



MINISTRY OF EDUCATION

COMPUTING

For Senior High Schools

TEACHER MANUAL

YEAR 1 - BOOK 2



NATIONAL COUNCIL FOR
CURRICULUM & ASSESSMENT
OF MINISTRY OF EDUCATION

MINISTRY OF EDUCATION



REPUBLIC OF GHANA

Computing

For Senior High Schools

Teacher Manual

Year One - Book Two



NATIONAL COUNCIL FOR
CURRICULUM & ASSESSMENT
OF MINISTRY OF EDUCATION

COMPUTING TEACHER MANUAL

Enquiries and comments on this manual should be addressed to:

The Director-General

National Council for Curriculum and Assessment (NaCCA)

Ministry of Education

P.O. Box CT PMB 77

Cantonments Accra

Telephone: 0302909071, 0302909862

Email: info@nacca.gov.gh

website: www.nacca.gov.gh



©2024 Ministry of Education

This publication is not for sale. All rights reserved. No part of this publication may be reproduced without prior written permission from the Ministry of Education, Ghana.



CONTENTS

INTRODUCTION	1
Learner-Centred Curriculum	1
Promoting Ghanaian Values	1
Integrating 21st Century Skills and Competencies	1
Balanced Approach to Assessment - not just Final External Examinations	1
An Inclusive and Responsive Curriculum	2
Social and Emotional Learning	2
Philosophy and vision for each subject	2
SUMMARY SCOPE AND SEQUENCE	3
SECTION 4: COMPUTATIONAL THINKING AND PROGRAMMING LOGIC	4
Strand: Computational Thinking (Programming Logic)	4
Sub Strand: App Development	4
Strand: Computational Thinking (Programming Logic)	4
Sub Strand: Algorithm and Data Structure	4
<i>Theme or Focal Areas:</i>	7
1. <i>Variables in computer programming.</i>	7
2. <i>Description of an algorithm including key characteristics and examples</i>	7
3. <i>Description of pseudocode with examples.</i>	7
4. <i>Description of a flowchart with examples.</i>	7
<i>Theme or Focal Areas: Identify and describe each of the following stages of the program development cycle: analysis, design, implementation, and testing.</i>	20
<i>Theme or Focal Areas:</i>	28
1. <i>Data types and examples</i>	28
2. <i>Data structures – an introduction and why data structures are important in programming</i>	28
3. <i>Classifications of data structures: linear and non-linear, static and dynamic</i>	28
<i>Theme or Focal Areas:</i>	38
1. <i>Describe arrays as a type of data structure</i>	38
2. <i>Explain the differences between one-dimensional and two-dimensional arrays</i>	38
3. <i>Advantages and disadvantages of arrays</i>	38
<i>Theme or Focal Areas</i>	47
1. <i>Linked lists – features, operations, examples, types, applications, advantages and disadvantages</i>	47
2. <i>Stacks – features, operations, examples, types, applications, advantages and disadvantages</i>	47



<i>Theme or Focal Areas</i>	55
1. <i>Queues – features of a linear queue, operations, examples, applications, advantages</i>	55
2. <i>Identify and describe the non-linear data structure, binary tree</i>	55
3. <i>Identify and describe the non-linear data structure, graph</i>	55
<i>Theme or Focal Areas</i>	64
1. <i>Programming basics using Python</i>	64
2. <i>How to translate a simple algorithm using single variables into actual code using a programming language</i>	64
<i>Theme or Focal Areas</i>	84
1. <i>Implementing and manipulating 1D arrays in Python</i>	84
2. <i>Built-in (array) list methods in python</i>	84
SECTION 5: COMPUTATIONAL THINKING AND WEB DEVELOPMENT	96
Strand: Computational Thinking (Programming Logic)	96
Sub Strand: Web Technologies and Databases	96
<i>Theme or Focal Areas</i>	98
1. <i>The difference between web design and web development</i>	98
2. <i>Web development – an overview, including an exemplar of a simple website</i>	98
3. <i>Components of a web page - headings, menus, links, text, images, and other relevant elements</i>	98
<i>Theme or Focal Areas</i>	112
1. <i>The role of a web designer</i>	112
2. <i>Web outline plan – What is it, and steps involved</i>	112
3. <i>Sitemaps – What is a sitemap? Exemplars of visual sitemaps</i>	112
<i>Theme or Focal Areas</i>	121
1. <i>Web page wireframes – what are they, their purpose, and how to create them</i>	121
2. <i>Website prototypes – what are they and how they are created and used</i>	121
3. <i>Advantages of using wireframes and prototypes</i>	121
ACKNOWLEDGEMENTS	131

INTRODUCTION

The National Council for Curriculum and Assessment (NaCCA) has developed a new Senior High School (SHS), Senior High Technical School (SHTS) and Science, Technology, Engineering and Mathematics (STEM) Curriculum. It aims to ensure that all learners achieve their potential by equipping them with 21st Century skills, competencies, character qualities and shared Ghanaian values. This will prepare learners to live a responsible adult life, further their education and enter the world of work.

This is the first time that Ghana has developed an SHS Curriculum which focuses on national values, attempting to educate a generation of Ghanaian youth who are proud of our country and can contribute effectively to its development.

This Book Two of the Teacher Manual for Computing covers all aspects of the content, pedagogy, teaching and learning resources and assessment required to effectively teach Year One of the new curriculum. It contains information for the second 12 weeks of Year One. Teachers are therefore to use this Teacher Manual to develop their weekly Learning Plans as required by Ghana Education Service.

Some of the key features of the new curriculum are set out below.

Learner-Centred Curriculum

The SHS, SHTS, and STEM curriculum places the learner at the center of teaching and learning by building on their existing life experiences, knowledge and understanding. Learners are actively involved in the knowledge-creation process, with the teacher acting as a facilitator. This involves using interactive and practical teaching and learning methods, as well as the learner's environment to make learning exciting and relatable. As an example, the new curriculum focuses on Ghanaian culture, Ghanaian history, and Ghanaian geography so that learners first understand their home and surroundings before extending their knowledge globally.

Promoting Ghanaian Values

Shared Ghanaian values have been integrated into the curriculum to ensure that all young people understand what it means to be a responsible Ghanaian citizen. These values include truth, integrity, diversity, equity, self-directed learning, self-confidence, adaptability and resourcefulness, leadership and responsible citizenship.

Integrating 21st Century Skills and Competencies

The SHS, SHTS, and STEM curriculum integrates 21st Century skills and competencies. These are:

- **Foundational Knowledge:** Literacy, Numeracy, Scientific Literacy, Information Communication and Digital Literacy, Financial Literacy and Entrepreneurship, Cultural Identity, Civic Literacy and Global Citizenship
- **Competencies:** Critical Thinking and Problem Solving, Innovation and Creativity, Collaboration and Communication
- **Character Qualities:** Discipline and Integrity, Self-Directed Learning, Self-Confidence, Adaptability and Resourcefulness, Leadership and Responsible Citizenship

Balanced Approach to Assessment - not just Final External Examinations

The SHS, SHTS, and STEM curriculum promotes a balanced approach to assessment. It encourages varied and differentiated assessments such as project work, practical demonstration, performance assessment, skills-based assessment, class exercises, portfolios as well as end-of-term examinations and final external assessment examinations. Two levels of assessment are used. These are:

- **Internal Assessment (30%)** – Comprises formative (portfolios, performance and project work) and summative (end-of-term examinations) which will be recorded in a school-based transcript.

- o External Assessment (70%) – Comprehensive summative assessment will be conducted by the West African Examinations Council (WAEC) through the WASSCE. The questions posed by WAEC will test critical thinking, communication and problem solving as well as knowledge, understanding and factual recall.

The split of external and internal assessment will remain at 70/30 as is currently the case. However, there will be far greater transparency and quality assurance of the 30% of marks which are school-based. This will be achieved through the introduction of a school-based transcript, setting out all marks which learners achieve from SHS 1 to SHS 3. This transcript will be presented to universities alongside the WASSCE certificate for tertiary admissions.

An Inclusive and Responsive Curriculum

The SHS, SHTS, and STEM curriculum ensures no learner is left behind, and this is achieved through the following:

- Addressing the needs of all learners, including those requiring additional support or with special needs. The SHS, SHTS, and STEM curriculum includes learners with disabilities by adapting teaching and learning materials into accessible formats through technology and other measures to meet the needs of learners with disabilities.
- Incorporating strategies and measures, such as differentiation and adaptative pedagogies ensuring equitable access to resources and opportunities for all learners.
- Challenging traditional gender, cultural, or social stereotypes and encouraging all learners to achieve their true potential.
- Making provision for the needs of gifted and talented learners in schools.

Social and Emotional Learning

Social and emotional learning skills have also been integrated into the curriculum to help learners to develop and acquire skills, attitudes, and knowledge essential for understanding and managing their emotions, building healthy relationships and making responsible decisions.

Philosophy and vision for each subject

Each subject now has its own philosophy and vision, which sets out why the subject is being taught and how it will contribute to national development. The Philosophy and Vision for Computing is:

Philosophy: The next generation of ethical creators and developers of technology can be empowered through observation, curiosity, exposure to related computing concepts and opportunities that leverage hands-on activities in a learner-centred environment leading to local and global relevance.

Vision: To prepare learners with 21st Century skills and competencies to ethically design, develop and apply computing systems to solve real-world problems.

SUMMARY SCOPE AND SEQUENCE

S/N	STRAND	SUB-STRAND	YEAR 1			YEAR 2			YEAR 3		
			CS	LO	LI	CS	LO	LI	CS	LO	LI
1.	Computer Architecture and Organisation	Data Storage and Manipulation	1	1	4	1	1	3	1	1	3
		Computer Hardware and Software	1	1	2	1	1	2	1	1	2
		Data Communication and Network Systems	1	1	3	1	1	3	1	1	3
2.	Computational Thinking (Programming Logic)	Algorithm and Data Structure	1	1	2	1	1	2	1	1	1
		App Development	1	1	2	1	1	2	1	1	2
		Web Technologies and Databases	1	1	2	1	1	3	1	1	3
Total			6	6	15	6	6	15	6	6	15

Overall Totals (SHS 1 – 3)

Content Standards	18
Learning Outcomes	18
Learning Indicators	46

SECTION 4: COMPUTATIONAL THINKING AND PROGRAMMING LOGIC

Strand: **Computational Thinking (Programming Logic)**

Sub Strand: App Development

Content Standard: Demonstrate knowledge and understanding of Computational Thinking (Algorithms and Programs)

Learning Outcome: *Apply Algorithms in various real-life and programming situations or problems*

Strand: **Computational Thinking (Programming Logic)**

Sub Strand: Algorithm and Data Structure

Content Standard: Demonstrate knowledge and understanding of Data Structures

Learning Outcome: *Apply knowledge of data structures to explain their real-life applications effectively*

INTRODUCTION AND SUMMARY OF SECTION

Computational thinking is using problem-solving techniques that imitate the process computer programmers go through when writing algorithms and computer programs. Programming logic is the structured thinking required to develop effective software solutions using the practical skills developed in a programming language. In this section, the learners will develop their computational skills by breaking down (decomposing) problem specifications and developing algorithms to solve these problems. The teacher will guide the learners how to use pseudocode and flowcharts to present their algorithms. Data structures are the fundamental constructs around which programs are built. The learners will have opportunities to develop programming logic skills by learning about data structures and algorithms, and through regular and consistent practice of creating algorithms and writing corresponding code in a suitable high-level programming language for use in a SHS. In this manual, all coding instruction is in Python. The data structures explored will be arrays, linked lists, stacks, linear queues, binary trees, and graphs. Of these, only algorithms and programs relating to 1-dimension arrays will be studied in depth. Adequate time will be required to study the basics of the programming language to be used. This will lay the foundation for the programming parts of the course and should not be rushed. A three-week block of lessons allows only enough time to include the bare fundamentals – variables, data types, simple sequence, and setting up and manipulating arrays without iteration of selection constructs. The importance of well-planned test data and adding internal commentary to their programs should be regularly emphasised to the learners. As well as a programming context, real-life life applications of data structures and algorithms will be examined.

The breakdown of the weekly learning indicators is as follows:

Week 13 : Explain and use algorithms to solve a real-life problem.

Identify the stages in the program development cycle: analysis, design, coding, testing

Part 1 – Variables in computer programming, Description of an algorithm including key characteristics and examples, Description of pseudocode with examples. Description of a flowchart with examples.

Week 14 : Explain and use algorithms to solve a real-life problem.

Identify the stages in the program development cycle: analysis, design, coding, testing

Part 2 - Identify and describe each of the following stages of the program development cycle: analysis, design, implementation, and testing.

Week 15: Explain in detail the concepts of Data Structures and their importance in organising and manipulating data efficiently.

Data types and examples, Data structures – an introduction and why data structures are important in programming, Classifications of data structures: linear and non-linear, static and dynamic.

Week 16: Differentiate between the types of Data Structures

Part 1 – Describe arrays as a type of data structure, Explain the differences between one-dimensional and two-dimensional arrays, Advantages and disadvantages of arrays

Week 17: Differentiate between the types of Data Structures

Part 2 - Linked lists – features, operations, examples, types, applications, advantages and disadvantages; Stacks – features, operations, examples, types, applications, advantages and disadvantages.

Week 18: Differentiate between the types of Data Structures

Part 3 - Queues – features of a linear queue, operations, examples, applications, advantages; Identify and describe the non-linear data structure, binary tree; Identify and describe the non-linear data structure, graph

Week 19 and 20

- 1. Implement algorithms into programs with the use of flowcharts, pseudocode and programming languages such as Python.**
- 2. Explain the steps in planning a program putting actions in the right order**

Part 1 - Programming basics using Python, How to translate a simple algorithm using single variables into actual code using a programming language

Week 21: Implement algorithms into programs with the use of flowcharts, pseudocode and programming languages such as Python.

Explain the steps in planning a program putting actions in the right order

Part 2 - Implementing and manipulating 1D arrays in Python, Built-in (array) list methods in python

SUMMARY OF PEDAGOGICAL EXEMPLARS

The pedagogical examples for this section are designed to guide teachers in enhancing learners' understanding of program design and development. This will include a study of some of the main data structures used in programming. Multiple instructional approaches that integrate direct teaching, visual aids, demonstrations of programs, and collaborative learning strategies are advised. Teachers are recommended to have a large bank of carefully selected practical tasks at hand. Developing and implementing algorithms is not easy for most learners new to program development. Lots of practice of developing algorithms of increasing complexity, and converting them to code (or vice versa) is essential for learners to progress well in this important part of the course. There are many exemplars of program design and practical tasks included in this manual. The main design notation used is

pseudocode with just a few examples of flowcharts. Commenting the algorithm within the code as a starting point is a strategy often use by teachers with newer programmers. This strategy is explained in the manual.

As usual, Teachers are expected to integrate GESI, SEL, and SEN in their classroom interactions to ensure that all learners, regardless of their personal and educational backgrounds, are provided with equitable opportunities to participate and excel in the lessons.

Teachers are expected to be familiar with their chosen IDE in advance of any practical programming lessons with the learners. If the learners are using iPads for coding, it is highly recommended that a set of keyboards is available to use with the class.

SUMMARY OF ASSESSMENT

This section applies the Depth of Knowledge (DOK) framework to evaluate learners' knowledge and skills acquisition as outlined in the focal areas. There are wide range of both formative and summative assessment questions and tasks that the teacher can build on. In weeks 19 to 21, learners should be instructed to use internal commentary to explain their code – this can be used as a measure of assessment.

WEEK 13

Learning Indicators:

1. *Explain and use Algorithms to solve a real-life problem.*
2. *Identify the stages in the program development cycle: Analysis, design, coding and testing.*

Theme or Focal Areas:

1. **Variables in computer programming.**
2. **Description of an algorithm including key characteristics and examples**
3. **Description of pseudocode with examples.**
4. **Description of a flowchart with examples.**

Key Concept Notes

Variables In Computing Programming

From earlier in the course, you will know that:

- A computer program is a set of instructions to complete a process such as to solve a given problem.
- A RAM chip on a computer's motherboard has storage locations. These locations hold data.

When we create a program we call the data held in these locations 'variables' if the data can be changed. In other words, in programming, a variable is **a value that can change**, depending on conditions or on information passed to the program. For example, if your program asks the name of the user to be entered, this input will vary depending on who is using the program.

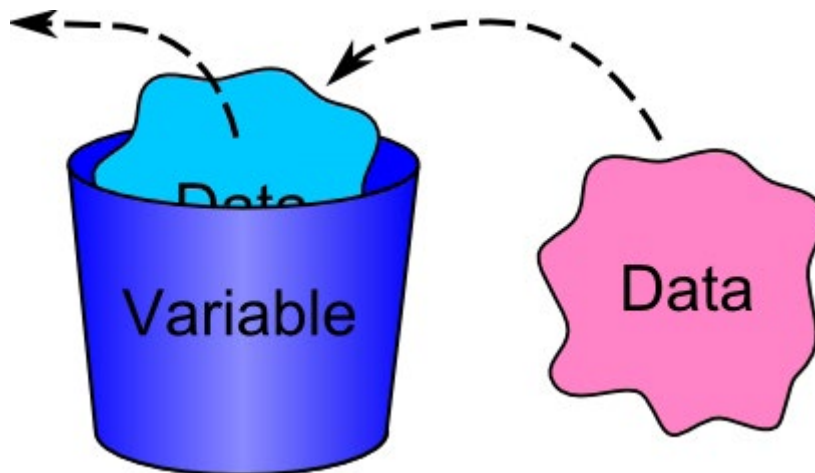


Figure 13:1 A variable is like a container. Its variable can change

Choosing suitable names for your variables is an important aspect in making your program code readable (i.e. easier to understand). Use variable names that describe their function whenever possible (e.g. *cost* rather than *number1*) and which follow a consistent theme throughout your code. There will be rules to follow and a number of naming conventions to choose from which may differ between programming languages. Most programming languages, including Python, will not permit variable names to start with a number nor a variable name with more than one word cannot have a space (use an underscore or camelCase). This means that *1name* or *first name* would not be permissible but

`name1` and `first_name` and `firstName` would be. CamelCase is a way to separate the words in a phrase by making the first letter of the second and subsequent word capitalised and not using spaces.

Algorithms

An algorithm is a step-by-step procedure or a set of well-defined instructions for solving a particular problem or performing a specific task. It is a fundamental concept in computer science, mathematics, and other fields that involve problem-solving and computation. Algorithms are a systematic way to describe how to achieve a specific objective, breaking down complex tasks into smaller, manageable steps that a computer (or a human) can follow to produce the desired outcome. An algorithm is a good way of planning a program before coding.

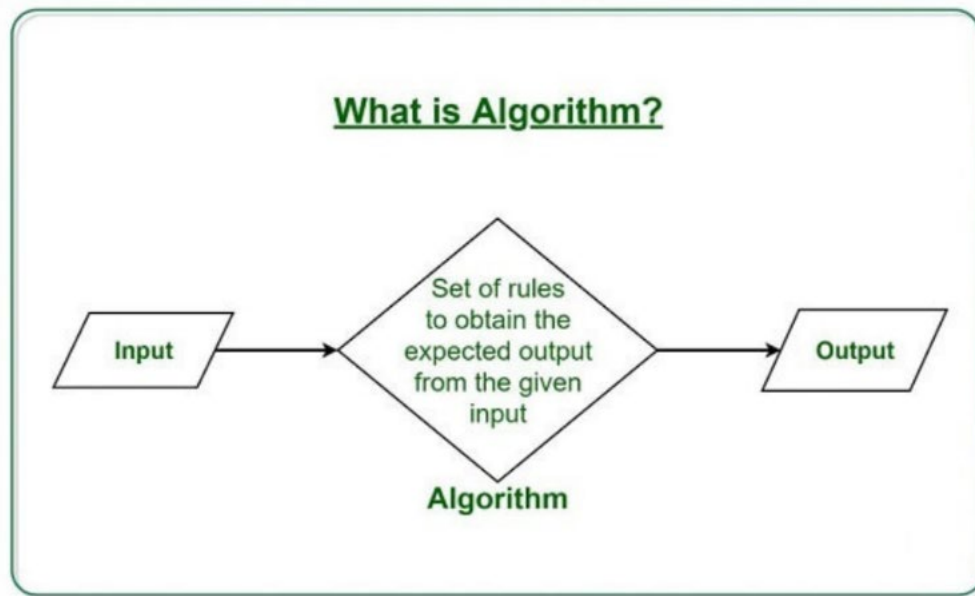


Figure 13:2 Algorithm Description

Key characteristics of algorithms

- **Input:** an algorithm takes zero or more inputs, which is the data or information the algorithm operates on, to produce the desired output. In a program, input could be entered by the user, read in from a file, or assigned within the program.
- **Output:** an algorithm produces an output, which is the result or solution to the problem based on the given inputs. In programming, this output is often displayed on the screen or stored in a file.
- **Definiteness:** each algorithm step must be precisely defined and unambiguous, leaving no room for interpretation or uncertainty.
- **Finiteness:** an algorithm must have a finite number of steps, meaning it should eventually terminate after a finite number of operations.
- **Effectiveness:** the steps of the algorithm should be simple and executable, meaning they can be performed by a computer or by a person with pen and paper.
- **Language independence:** algorithms must contain instructions that can be implemented in any suitable programming language, yet the output will be as expected.

Algorithms range from simple calculations to complex decision-making processes and they play a significant role in various domains. Algorithms ensure tasks are done systematically, saving time and resources, and reducing errors.

Examples of algorithms

1. Real-life Algorithms

An algorithm for a child getting ready for school could be:

1. Wake up
2. Brush teeth
3. Wash
4. Dress
5. Eat breakfast
6. Get school bag
7. Travel to school
8. Arrive at school
9. Enter school premises

Note that there could be many variations of this algorithm. For example, the order of the steps may be different for some children, such as step 3 (Wash) swapping with step 2 (Brush teeth), or the algorithm may be more detailed. A version could replace Step 1 Wake up with three steps: Turn off alarm clock, Get out of bed, Make bed.

An example of an algorithm for eating breakfast which involves iteration (repetition) is as follows:

1. Put porridge in a bowl
2. Add sugar and milk to the porridge
3. Stir to mix in sugar and milk
4. Spoon porridge into the mouth
5. Repeat step 4 until all porridge is eaten
6. Rinse bowl and spoon

2. Search Algorithms

A search algorithm defines a step-by-step method for locating specific elements in a data set. For example, a search algorithm could be used to find Ghana in a list of countries.

3. Sorting Algorithms

These are algorithms that puts elements of a list into an order, for example, to put the percentage test marks achieved by a class in ascending order.

4. Encryption Algorithms

These are algorithms that encode data to make it more secure when being stored or transmitted. For example, HTTPS websites that transmit credit card and bank account numbers encrypt this information to prevent identity theft and fraud. These websites will use encryption algorithms.

5. Mathematical Algorithms

These are algorithms that perform mathematical operations, such as finding the factorial of a number or calculating the greatest common divisor (GCD) of two numbers. Figure 13.3 is a simple algorithm written in pseudocode to add two given numbers and output the sum. Figure 13.4 shows an implementation of this algorithm in Python.

There are many other examples of algorithms, including those that can solve significant real-world problems such as optimising traffic flow, financial forecasting, healthcare diagnostics, and environmental monitoring.

Algorithms can be represented in different ways. Two ways are pseudocode and flowcharts.

PSEUDOCODE

Pseudocode is an algorithm (a list of instructions) written in English. These instructions need to be in the correct order to complete a process and the list is read from top to bottom. It is good practice to number the instructions/steps as shown in the real-life examples given on the previous page and in Figure 13:3.

Problem specification: Find the sum of 529 and 256 Pseudocode: 1. Assign x the value 529 2. Assign y the value 256 3. Assign z the sum of x and y. 4. Output the value of z
--

Figure 13:3 A simple mathematical algorithm

An implementation of this algorithm using the programming language, Python, as well as the output when the program is run, is given in Figure 13:3 below.

Pseudocode	Code/program	Output
1. Assign x the value 529 2. Assign y the value 256 3. Assign z the sum of x and y. 4. Output the value of z	<pre>x = 529 y = 256 z = x+y print(z)</pre>	785

Figure 13.4 :Algorithm with program and output

Another example of a simple algorithm written in pseudocode is given in Figure 13:5. A corresponding program in Python is also shown and the output when 5,4, and 2 are entered.

Problem specification:

A program is required that will prompt the user to enter three numbers and will output the product of these numbers.

Write the algorithm for this task in pseudocode.

Pseudocode:

1. Enter first number, number1
2. Enter second number, number2
3. Enter third number, number3
4. Let result = number1 x number2 x number3
5. Output result

Program (written in Python):

```
number1 = int(input("Enter first number: "))
number2 = int(input("Enter second number: "))
number3 = int(input("Enter third number: "))
result = number1 * number2 * number3
print(result)
```

Output:

```
Enter first number: 5
Enter second number: 4
Enter third number: 2
40
```

Note that the operator for multiplication in Python is *.

It is important to understand that pseudocode is not real code. Pseudocode is written in English and not written in a programming language. As mentioned earlier, pseudocode can be translated into any suitable programming language. Figure 13:6 shows the previous algorithm implemented in the Scratch programming language. Given the same inputs (5, 4, and 2), the same value should be output (40) whether the algorithm is implemented in Python, Scratch, or other programming language.

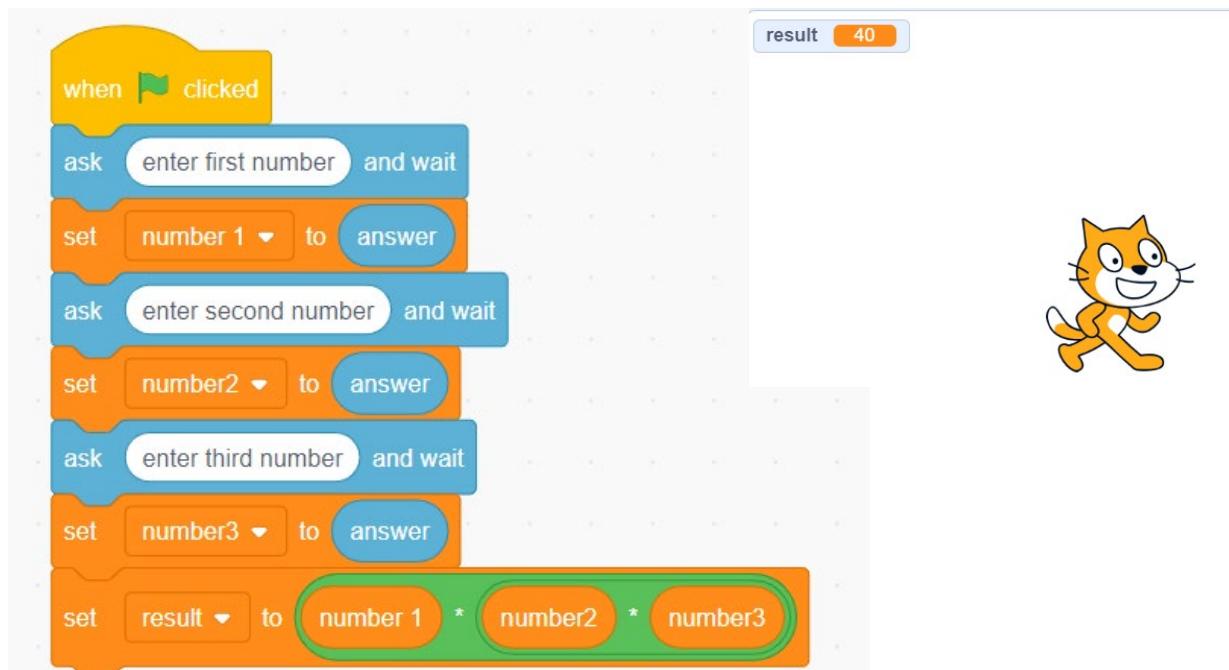


Figure 13.6: Scratch program with output

FLOWCHARTS

Flowcharts are graphical representations of algorithms that use shapes and arrows to illustrate the sequence of steps in an algorithm. Each shape represents a specific action or decision, and the arrows show the flow of the algorithm from one step to another. Flowcharts can play an important role in displaying information and assisting reasoning. Using flowcharts, learners can visually understand the logic and structure of an algorithm.

Basic symbols used in flowchart designs

1. **Terminator:** The terminator symbol represents the starting or ending point of the system.



Figure 13.7: Symbol for Start/Stop

2. **Input/Output:** A parallelogram denotes any input/output type function. Program instructions that take input from devices and display output on output devices are indicated with a parallelogram in a flowchart. It represents information entering or leaving the system. An input might be an order from a customer. Output could be the name of a product to be delivered.

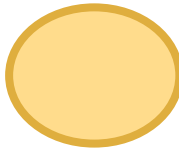


Figure 13.8: Symbol for Input/Output

3. **Processing:** A box represents arithmetic instructions. All arithmetic processes, such as adding, subtracting, multiplication and division, are indicated by action or process symbol. It has one incoming flowline and one outgoing flowline.



Figure 13.9: Symbol for Process

4. Decision: The diamond symbol represents a decision branching point. Diamonds indicate decision-based operations such as yes/no or true/false in a flowchart. It has one incoming flowline and two outgoing flowlines.	 Figure 13.10: <i>Symbol for Decision</i>
5. Flow lines: Flow lines indicate the exact sequence in which instructions are executed. The arrow represents the direction of the flow of control and relationships among different flowchart symbols.	 Figure 13.11: <i>Symbol for Flow</i>
6. Off-page connector: This symbol would contain a letter inside. It indicates that the flow continues on a matching symbol containing the same letter somewhere else on a different page.	 Figure 13.12: <i>Symbol for Off-page Connector</i>
7. On-page connector: It connects the flowchart on the same page. It has one incoming flowline.	 Figure 13.13: <i>Symbol for On-page Connector</i>

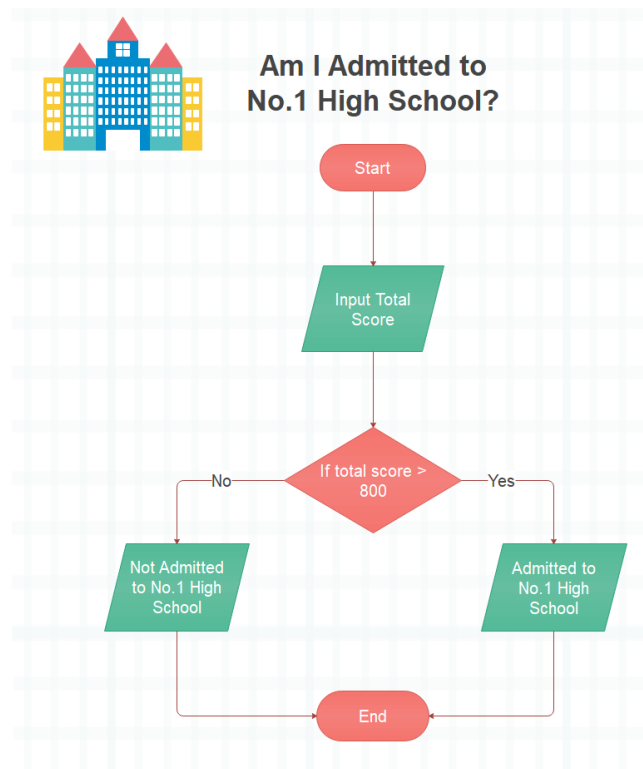


Figure 13.14

Problem specification:

Find the sum of 529 and 256

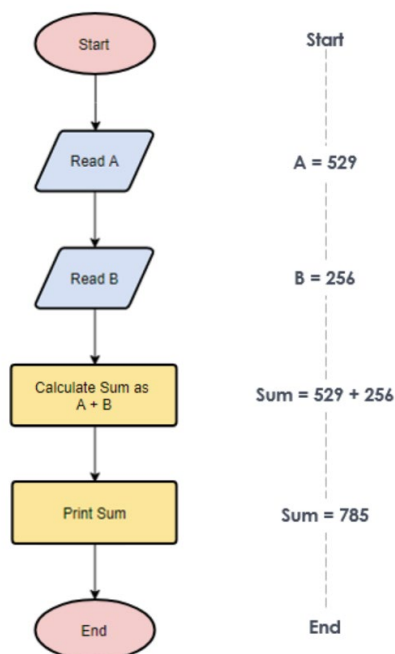
Flowchart:

Figure 13.15: Flowchart showing the summation of two numbers

Here is an example of a flowchart for a real-life process: The simple mathematical algorithm from earlier could also be represented using a flowchart. A possible version of this flowchart is shown in Figure 13:15.

Note that this algorithm of the summation task uses different variable names from the pseudocode version shown in Figure 13:3.

Learning Tasks

Here are some tasks to help learners understand the focal areas in week 13. Kindly take note of differentiated learning when applying these tasks.

Task 1

1. Learners are tasked to think-pair-share daily routines (not mentioned by the teacher) that could be described as algorithms.
2. Learners in their mix-ability groupings are assigned one or more of these real-life problems to be solved using an algorithm (e.g. creating a study schedule).
3. Each group presents their algorithm(s) to the class.
4. In an open class discussion, learners discuss with the teacher how the given algorithm(s) addresses the problem(s) and possible ways that the algorithm(s) can be improved.

Task 2

1. Working in small groups, the learners should develop pseudocode and a flowchart for each of the following problem specifications:
 - Write a program to multiply 37 and 70
 - Write a program to output the sum of four number entered by the user
2. Select one or two solutions for open class scrutiny and invite suggestions how these solutions can be corrected/improved.

Task 3

1. Give each group a set of strips of paper/card each containing a line of pseudocode, such as the lines of pseudocode to calculate the circumference of a circle with radius equal to 5cms.
2. The learners work together to put the lines in the correct order to solve a given problem.
3. This activity could be repeated for different problems.

Task 4

1. Present to the class a partially completed algorithm in pseudocode for finding the average of two numbers. Working together as one, the learners decide on how to fill in the gaps.
2. Each learners should then use this algorithm to create a corresponding flowchart.

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Teachers can begin the lesson with direct instruction, giving a clear and concise explanation of what is meant by algorithms, emphasising their role in problem-solving within various fields, such as computer science and mathematics.
2. Using suitable examples, the teacher should introduce pseudocode as one method of representing algorithms, starting with algorithms of everyday routines (such as frying an egg – see next page). Emphasise the importance of the accuracy of each step and the correct sequence order.

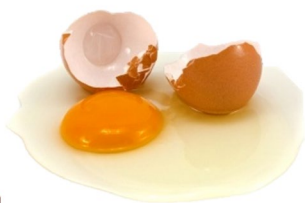
This should lead to the TPS Learner Task 1.

Frying an egg

Sequencing Instructions:

What is the correct order of these instructions?

1. Put pan on hob
2. Break egg
3. Get frying pan
4. Put oil in pan
5. Turn on hob
6. Hold egg over pan



Frying an egg

1. Get frying pan
2. Put pan on hob
3. Turn on hob
4. Put oil in pan
5. Hold egg over pan
6. Break egg



3. Moving on to examples of mathematical algorithms, the teacher should emphasise that a good way to start to write pseudocode for a given problem is to identify the inputs, processes, and outputs required.
4. Together, the whole class could write more algorithms in pseudocode, such as those given in Learner Task 2.
5. Writing code is something that will be covered in more detail later in the manual. At the very least, a demonstration by the teacher of how some of these algorithms can translate into Python code should be included at this stage. This will help learners understand the importance of using one statement per line in pseudocode. If learners have adequate programming experience, they could also be tasked with implementing some of the algorithms.
6. The importance of the order of the steps in an algorithm will be reinforced by completing the group activity, Learner Task 3.
7. The teacher should ensure that opportunities are given to learners to write some algorithms in pseudocode by themselves, perhaps starting off with partially completed pseudocode task as described in Learner Task 4. Lots of practice tasks should be available to learners at this point. A sound understanding of simple pseudocode should be demonstrated before introducing flowcharts.
8. The teacher should introduce flowcharts as a graphical method for representing algorithms, then explain the key symbols and their meanings, and present a number of examples of flowcharts of everyday routines. The teacher can guide the learners through repeating the TPS Learner Task 1 using flowcharts this time.
9. Learner Tasks 3, 4, 5, and 7 could then be completed using flowcharts rather than pseudocode.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the four focal areas.

DOK Level 1: Recall/Reproduction

1. The symbol shown in Figure 13:16 represents which of the following when shown as part of a flowchart?

**Figure 13.16**

- a. Process
 - b. Input/Output
 - c. Start/Stop
 - d. Loop
2. What is an algorithm?
 - a. A computer program
 - b. A graph
 - c. Step-by-step instructions used to solve a problem
 3. What algorithm could be used when making a birthday cake?
 - a. A map
 - b. A recipe
 - c. A set of ingredients
 4. Name two ways to represent an algorithm.
 5. How are the symbols in a flowchart connected?
 6. When you write an algorithm the order of the instructions is very important.
 - a. true
 - b. false
 7. What should be considered when designing an algorithm?
 - a. If the correct hardware is being used
 - b. If the correct software is being used
 - c. If there is more than one way of solving the problem
 8. State what is meant by a variable in programming.
 9. Flowcharts are read from
 - a. top to bottom
 - b. left to right
 - c. bottom to top
 10. State an example of a schooltime routine algorithm.
 11. Study the algorithm below and answer the questions that follow.
 - a. What is the name of this method of representing an algorithm?
 - b. How many different variables are shown in this algorithm?
 - c. How many inputs does this algorithm have?
 - d. How many outputs does this algorithm have?
 12. Complete the sentence:
 The purpose of an algorithm is to provide step-by-step _____ for problem-solving.

13. Making a sandwich is kitchen routine that can be represented as an algorithm. Identify another kitchen routine that could be classed as an example of an algorithm.
14. State where a program's variables are stored as the program is being run.
15. What is the name of the type of algorithms that would order elements in a list.
16. Which of the following variable names for the cost of an item would be the most appropriate?
 - a. item
 - b. 1cost
 - c. cost of item
 - d. cost

DOK Level 2: Skills and Concepts

1. Give one similarity and one difference between pseudocode and a flowchart.
2. Describe your daily morning routine before school in sequential steps.
3. Which of the following variable names for the cost of an item would be the most appropriate? Explain your reasoning.
 - a. item
 - b. 1cost
 - c. cost of item
 - d. cost
4. Give an example of a problem that would require a searching algorithm.
5. Create a flowchart for outputting the difference between two numbers input by the user.
6. Study the given flowchart (Figure 13:17) and answer the questions that follow.



Figure 13.17

- a. Circle the process symbol.
- b. What would have to be true for the looking for lost items to stop?
- c. How many input/output symbols are shown in the flowchart?
7. Write an algorithm for the directions to get from your home to school.
8. Draw a flowchart that outlines the steps to solve a simple school-related task, like signing the class attendance register.

DOK Level 3: Strategic Thinking

1. Explain the connection between a problem and an algorithm.
2. A problem specification is to write a program that will output the result of the first number divided by a second number. Both numbers should be entered by the user. The following algorithm has been written to plan for this program:
 1. Enter first number, x
 2. Enter second number, x
 3. Let $\text{answer} = x$ divided by x
 4. Output answer

Will this algorithm work? Justify your answer.

3. Critique the effectiveness of a flowchart in solving a given problem (problem and flowchart need to be given).
4. Every problem has only one algorithm. Is this a true statement? Give examples to support your answer.

DOK Level 4: Extended Thinking

1. Investigate how search engines use algorithms and write a report on your findings.
2. Interpret a more complex flowchart, such as the flowchart in Figure 13:18.

State the output from this flowchart. Show your working.

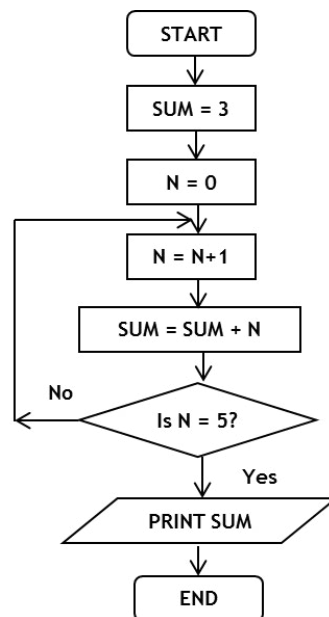


Figure 13.18

3. Research AI algorithms and create a slideshow on this topic. Add suitable multimedia elements (diagrams, videos, etc.) to your slides.
4. Integrate multiple simple flowcharts to develop a complex flowchart that solves a multi-faceted problem, such as planning a school event from initiation to completion.

WEEK 14

Learning Indicators:

1. Explain and use Algorithms to solve a real-life problem.
2. Identify the stages in the program development cycle: Analysis, design, coding and testing.

Theme or Focal Areas: Identify and describe each of the following stages of the program development cycle: analysis, design, implementation, and testing.

Note that for completeness, other stages of the program development cycle, such as maintenance are included in the Key Concept notes.

Key Concept Notes

Program Development Cycle/Program Development Life Cycle

(more commonly known as **Software Development Cycle/ Software Development Life Cycle**)

When creating new software/programs, it is common for a software development team to work through different 'phases'; these are collectively known as the software development cycle or software development life cycle (SDLC). The SDLC is a systematic process which is commonly used by software development teams to design, develop and test software. It consists of a series of phases that guides the development process. The aim of the SDLC is to produce software with the highest quality and lowest cost that meets the client's requirements in the shortest time possible. The client is the person or organisation for which the software is being developed.

While there are various SDLC models, and the following are the common phases/stages in a typical software development cycle: Analysis, Design, Coding (Implementation), Testing, and Maintenance (see Figure 14:1).

The whole process starts when a client asks a software development team to develop a program. The client will give problem definition/statement/specification to the team.

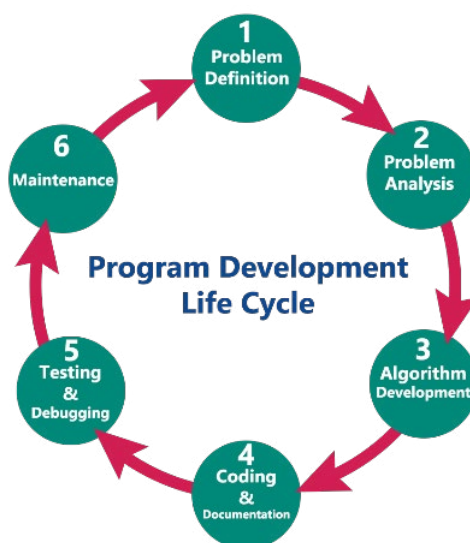


Figure 14.1

Problem Analysis

Analysis is the first phase of software/program development. At this stage, a representative from the development team (usually the systems analyst) will communicate with the client about the problem specification to establish the purpose and functional requirements of the program. The purpose of a program is often expressed as a description of what the program will be used for. The functional requirements will specify inputs, processes and outputs. It is also often necessary at the analysis stage to clearly outline any assumptions that have been made.

Example 1

This uses a problem specification introduced in Week 13: write a program to output the product of three numbers inputted by the user.

Purpose

A program should be created to allow a user to enter three numbers. The program should then multiply these three numbers together and output the result.

Functional Requirements

Inputs	Processes	Outputs
Three numbers	Multiple the three numbers	The product

Assumptions

- All numbers that are input will be valid numbers

Example 2

Problem specification: a program is needed to generate a list of usernames for a class of 20 pupils given their first names, surnames, and valid ages.

Purpose

A program is to be developed to create usernames for a class of twenty pupils. The program will ask a teacher to enter the first name, surname, and age of each pupil. The age entered must be between five and eighteen. The program should output a list of usernames.

Functional requirements

Inputs	Processes	Outputs
20 pupil first names	Validate ages	List of usernames
20 pupil surnames	Create usernames	
20 pupil ages		

Assumptions

- Only 20 sets of details will be entered.
- The output will link the name of each pupil to his/her username

Design

During this phase, the purpose statement, functional requirements, and any assumptions finalised at the analysis stage will be considered, and programmers will begin to develop a plan to create the software. An important part of this plan is the algorithm. Various methods can be used to represent the algorithm which include pseudocode and flowcharts; these design methodologies were studied in week 13. Other items usually produced at this stage are a list of the variables and data types that will be required, and a design for the user interface.

A user interface is the part of software that the user sees and interacts with. The user interface often includes features to allow the user to input data and includes areas where the user will see on-screen output. When designing user interfaces, it is common practice to use wireframes. This is a drawing of the screen outlines that the user of the program will interact with. Wireframes can be drawn by hand on paper or as shown in Figure 14:2, drawn using a computer.

Figure 14.2 Wireframe for an input screen

Implementation

This phase uses a programming language to write the code to implement the instructions/steps defined in the algorithm. As we enter the code, debugging may be required. Debugging means to correct errors in the code. An example of a syntax error (error in the spelling or grammar used when coding) is shown in Figure 14:3 where the command word *print* is incorrectly entered as *pront* so the program will not run correctly.

<pre>message1 = "Believe you can and you're halfway there." print (message1) print("\n") #prints a blank line in Python message2 = "Have a good day!" print (message2) message1 = "Believe you can and you're halfway there." pront (message1) print("\n") #prints a blank line in Python message2 = "Have a good day!" print (message2)</pre>	Believe you can and you're halfway there. Have a good day! Traceback (most recent call last): File "./prog.py", line 2, in <module> NameError: name 'pront' is not defined
---	---

Figure 14.3: A correct program and the program with a syntax error, each with corresponding outputs

Testing

To make sure a program actually solves the problem it is supposed to, you have to test it. This happens during the Test stage of the SDLC. Often this involves checking what the output from the program will be using various sets of test data, to check if it gives the desired output or not.

Testing should be systematic, that is, it should be planned, and the results of test runs recorded. Also, testing should be as comprehensive as possible. This may involve using different types of test data - normal, exceptional and extreme. For example, if a program is being developed to check how many

pupils in a class of ten passed a test where the pass result is 50% or higher, a possible comprehensive set of test data would be 56, 78, 47, 90, -82, 79, 58, 60, 50, 77. This set contains three types of data:

Normal data is as expected data that the program should accept as input (56, 78, 47, 90, 79, 58, and 77 in the above example). This data should output the desired output. The result of 90 in the above example should register a pass.

Extreme data is data on the boundary, such as 50 in the set given above. A % result of 0 or 100 would also be considered as boundary data in this program.

Exceptional data is out-of-range or invalid data, such as -82 in this example. A test result of 'forty' would also be considered exceptional data in this example.

Many large programs, often games, contain bugs, simply because it may not be possible to test every possible input/action that a future user might enter/make. The best way to use test data is to, if possible, work out what the output should be manually before you run the program. It is also recommended that someone other than the person who wrote the program to also test it. They may more easily spot mistakes that the coder may have missed.

Apart from syntax errors, other error that might be highlighted during the testing phase are run-time errors (also known as execution errors) and logic errors. An execution error occurs when a program is asked to do something that it cannot, resulting in a 'crash'. An example of a run time error is asking a program to divide by 0 or an attempt to perform operations on data of the wrong type. A logic error would be an error in the design of the program such as using the following incorrect formula to calculate the circumference of a circle with a radius of 5: *circumference* = $10 * 3.14 * 5$.

After testing and before the deployment of the software (i.e. making the software available for the client), installation instructions, user guides, and training materials are prepared to assist the client in using the software effectively. This is usually classified as a separate phase in the development cycle called Documentation. Another phase that is often added after Documentation is Evaluation where the program is reviewed against the initial problem specification

Maintenance

This phase occurs after the client starts using the solution (program). There are different kinds of program maintenance: fixing faults that turn up once it is being used regularly, improving the design to make it even better, or making changes for other situations (like making a version of the program that will work in another country or on a different operating system).

Learning Tasks

Here are some tasks to help learners understand the focal area in week 14. Kindly take note of differentiated learning when applying these tasks.

Task 1 - Pairs activity

1. Each pair of learners should rearrange cards that contains a stage of the SDLC in the correct order.
2. After the correct order is confirmed, each group should come up with a mnemonic (e.g. **A Dance In The Moonlight** for Analysis, Design, Implementation, Testing, and Maintenance)
3. Each pair should share their chosen mnemonic with the whole class.
4. A mnemonic for the SDLC for the class to use can be voted on.

Task 2 - Group Activity

1. Study a simple problem specification such as ‘Write a program to calculate and output the area of a circle using a radius value in centimetres entered by the user and taking pi as 3.14’.
2. Work together in your groups to complete as many of the steps in the SDLC as possible:
 - Identify the functional requirements (Analysis)

Inputs	Processes	Output

- Using a suitable design methodology, create an algorithm for a solution to this problem. (Design)
- If learners within the group have the necessary programming skills, write a program to match their algorithm. (Implementation)
- Create a set of data that could be used to test this program. This set should include normal data (e.g. 4, 8, 12.5) and exceptional data (e.g. -6). (Testing)
- The groups could present their work to the whole class and reflect on any difficulties they had when completing the task.

Task 3

Repeat Task 2 using a slightly more advanced problem specification; for example, ‘Write a program to calculate and output the area of the walls in a room using the values entered by the user for the length, breadth, and height of the room in metres’.

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Using appropriate visual aids, the teacher should explain what is meant by the Program Software Development Life Cycle. What happens at the Analysis, Design, Implementation, and Testing stages should be discussed. The Documentation, Evaluation, Maintenance stages can be included for completeness but are not required at this level (not mentioned in the Curriculum).
2. Working in pairs, the learners should complete Learner Task 1
3. After the teacher has guided the learners through a number of worked exemplars, the whole class should split into small groups to complete Learner Task 2. Having a bank of similar simple problem specifications for the groups to work on at this time is recommended. GESI, SEL, and SEN should be taken into consideration when making the groupings. The teacher should be on hand to check each stage of the development process as it is completed. The implementation of the algorithms is not expected at this time.

Requests for teacher support should decrease as the groups work through the similar problem specifications. Such specifications could include calculating the perimeter of a square or the volume of a cuboid.

Occasionally, bring the groups together to discuss and present their work.

4. Time should be factored for learners to individually complete their own program development documentation. This will allow the teacher to assess each learner’s understanding of the focal area.

5. Provide slightly more challenging specifications, such as Learner Task 3, for the more able learners, or group the learners carefully so that some group members can teach the other how to complete these tasks.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the focal area.

DOK Level 1: Recall/Reproduction

1. Name the first four phases/stages of the Program Development Cycle.
2. What is the second stage of the SDLC?
3. At what stage of the program development cycle would a wireframe of the user interface be created?
 - a. Analysis stage
 - b. Design stage
 - c. Testing stage
4. A client has requested a payroll program for her business that works on Windows computers. After she has used the program for several months, she requests a version of the program that works on Mac computers. Which stage of the SDLC is this?

Note that this is an optional question which can be included if the Maintenance stage is covered.

5. Name the type of test data when input is out-of-range.
6. Define the Program Development Cycle and explain its significance in software development.
7. What happens at the design phase of the SDLC?
 - a. The code is produced
 - b. Changes are made to the software has been created
 - c. Creating an algorithm and user interface
8. a. At what stage of the SDLC are the functional requirements created?
 b. Complete the following sentence: The functional requirements of a program will specify its inputs, _____, and _____.
9. What is the purpose of the testing stage when developing a program?
 - a. to ensure that the program has an attractive user interface
 - b. to ensure that a user becomes familiar with a program
 - c. to ensure that the program functions as intended
10. What is the name of the stage in software development that refers to the planning of a solution of a program?
11. Which design methodology for a program is as shown below?
 - a. Take in the price of one apple
 - b. Take in the number of apples
 - c. Calculate the final cost of the total number of apples

- d. Display the final cost of the apples

Level 2: Skills and Concepts

1. Explain the role of the client at the Analysis stage of SDLC.
2. Describe what is meant by extreme test data and give an example to support your answer.
3. What is the purpose of the Testing stage in the Program Development Cycle?
4. Below are some examples of program design. In each case the sequence of steps is incorrect. You must show the correct order of the steps. The correct sequence of the first example has been partially completed.

a.

1	Take in the price of one apple	1
2	Take in the number of apples	
3	Display the final cost of the apples	4
4	Calculate the final cost of the total number of apples	

b.

1	Calculate the price of an item including VAT	
2	Take in the price of an item	
3	Take in the current rate of VAT	
4	Display the final price of the item to the customer	

c.

1	Play the music track	
2	Decide if that point was on the 'play' button	
3	Get the position on the screen that was touched	

5. Describe what is meant by a logical error. State one example of pseudocode containing a logical error to support your answer.
6. Why does a program need to be tested?
7. A program is required to calculate the wall area of a room given its length, breadth and height. Name and describe one method of representing a possible algorithm that could be used when planning this program.
8. A program is required to find the average of four numbers. Complete the following table to show the functional requirements for this program:

INPUTS	PROCESSES	OUTPUTS

9. During testing, "Yes" is entered as the answer to:

"Enter another name? Y/N"

What type of test data is "Yes"? Explain your reasoning.

DOK Level 3: Strategic Thinking

1. Explain how the outcomes from the Analysis stage influence decisions made at the Design stage of the SDLC.
2. When executed, the program shown in Figure 14:4 generates a run-time error. Explain why this is so.

```
message = "Happy Birthday to you!"
print(message1
```

Figure 14.4: *Python program with run-time error*

3. A program is required to calculate the wall area of a room given its length, breadth and height. Using a design method that you are familiar with, write an algorithm for this program.
4. A program is to be created that outputs the corresponding season when a month is entered by the user. State two assumptions that might be made when identifying the functional requirements.
5. Some software development models are considered iterative. This means than an earlier stages in the development process can be revisited and changes made if required when new information becomes available. Describe how this may be true using the following stages:
 - a. Testing and Design
 - b. Design and Analysis

DOK Level 4: Extended Thinking

1. Investigate the following two software development models: Waterfall model and Agile model. Create a comparison table between these two models using the criteria: Client interaction, Teamwork, Documentation, Measurement of progress, and Testing.

Scan for three other questions and solutions:



WEEK 15

Learning Indicator: *Explain in detail the concepts of Data Structures and their importance in organising and manipulating data efficiently.*

Theme or Focal Areas:

1. **Data types and examples**
2. **Data structures – an introduction and why data structures are important in programming**
3. **Classifications of data structures: linear and non-linear, static and dynamic**

Key Concept Notes**Data Types**

In computer science, a data type is an attribute associated with a piece of data that tells a computer system how to interpret its value. We can define data as an elementary value or a collection of values. For example, an employee's name and ID may be the data related to an employee used by a program. A single unit of value is known as a data item. In the previous example, a data item could be Abena for an employee's name.

Examples of data types include integers, floating-point numbers, booleans, and strings. These types are often abbreviated to `int`, `float`, `bool`, and `str`. They are pre-defined data types that are built into most programming languages and have a fixed size and format.

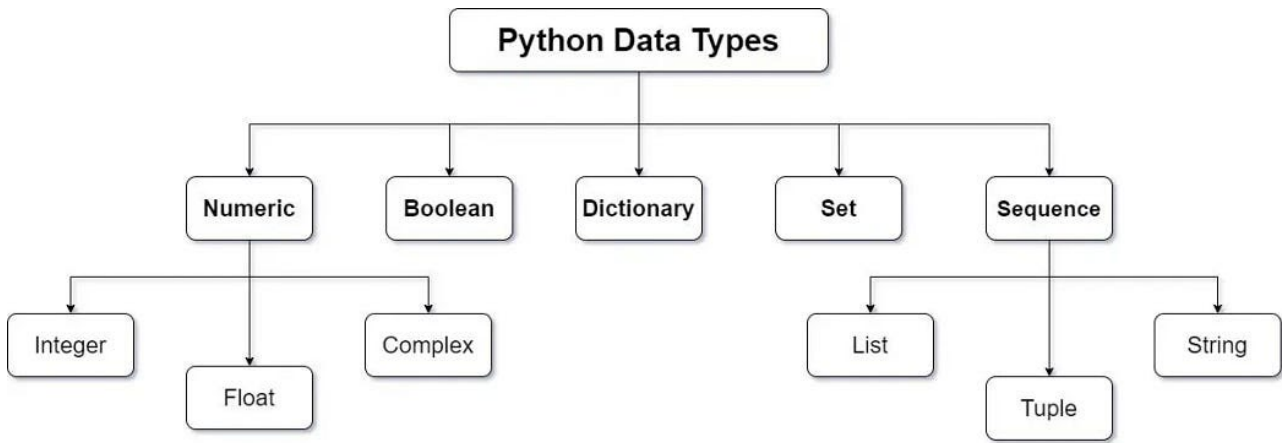
A variable is a named location in memory that stores a value of a specific data type. The data type defines which operations can safely be performed to create, transform and use the variable in another computation. A variable can have a short name (like `x` and `y`) or a more descriptive name (age, `carName`, `total_volume`).

Rules for naming Python variables

- Name must start with a letter or the underscore character.
- Name cannot start with a number
- Name can only contain alpha-numeric characters and underscores (A-z, 0-9, and `_`)
- Names are case-sensitive (age, Age and AGE are three different variables)

Data types in Python

The data types in Python are shown in Figure 15:1 and some of these will be exemplified in the manual. Integer and real values are specified by `int` and `float` respectively. A string variable be of type `str` and will store a sequence of characters while a Boolean variable will be of type `bool` and store either `True` or `False`.

**Figure 15.1**

In some programming language, a variable has to be declared, indicating its name and data type, before it can be used. In Python, you do not need to declare variables before using them. The data item is set to a type when you assign a value to a variable. Some examples are given in the table using a variable called 'x'.

Line of code	Data type
x = "Hello World"	str
x = 1961	int
x = 43.5	float
x = True	bool

If you want to set a variable to a specific data type, you can do so using type conversion functions as exemplified below using Python.

Line of code	Data type	Comment
x = str("Hello World")	str	
x = int(1961)	int	
x = float(43.5)	float	
x = bool(True)	bool	
x = str(1961)	str	
X = int(43.5)	int	

Mathematical operations will not work as expected– see Figure: 15.2 below

The decimal part will be truncated.

The Python function `type()` can be used to check data type. Studying the Python program and corresponding output in Figure 15.2 will aid understanding how basic type conversion and the `type()` function works in Python.

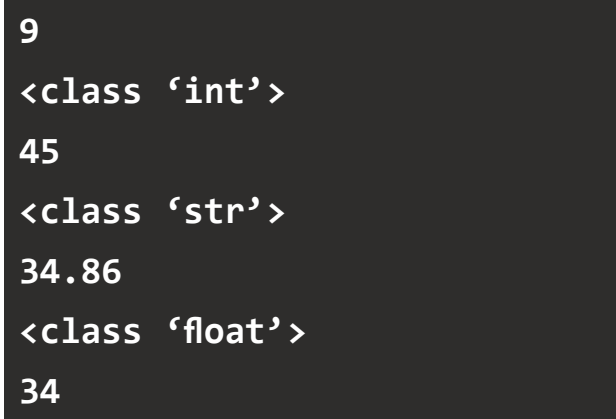
Python program	Output
<pre> x = 4 y = 5 print(x+Y) print(type(x)) x = str(4) Y = str(5) print(x+y) print(type(x)) z = 34.86 print(z) print(type(z)) print(int(z)) </pre>	 <pre> 9 <class 'int'> 45 <class 'str'> 34.86 <class 'float'> 34 </pre>

Figure 15.2

Data Structures

A single variable can be considered a data structure in the most fundamental sense. Data structures are more commonly understood to be more complex arrangements that can hold multiple values or collections of data. Data structures are a fundamental concept in computer science. They are used to organise and store collections of data in a way that facilitates efficient access and modification. Examples include arrays, linked lists, stacks, queues, trees, and graphs. These structures can hold multiple items of certain data type(s), and provide ways to efficiently access, manage, and manipulate these data items.

Data structures are essential for solving various computational problems and optimising the performance of algorithms. They are like the building blocks that allow programmers to efficiently store, retrieve, and manipulate data in computer programs.

Why should we learn data structures?

1. A data structure is an important concept in computer science.
2. Data structures allow us to organise and store data
3. Learning about data structures is required to become a programmer.
4. Data structures enable efficient storage and retrieval of data, reducing processing time and improving performance.
5. The choice of the most appropriate data structures will enable you to write more efficient code. (Note that two measures of efficient code are how fast it takes to run and how little memory is needed by the code.)
6. Data structures often hide the implementation details of data storage, allowing programmers to focus on the logical aspects of data manipulation.

The importance of data structures in organising and managing data efficiently will become more apparent in future weeks when exploring different data structures and their real-life applications.

Types of Data Structures

Data Structures are often classified into two main types:

1. Linear data structures
2. Non-linear data structures

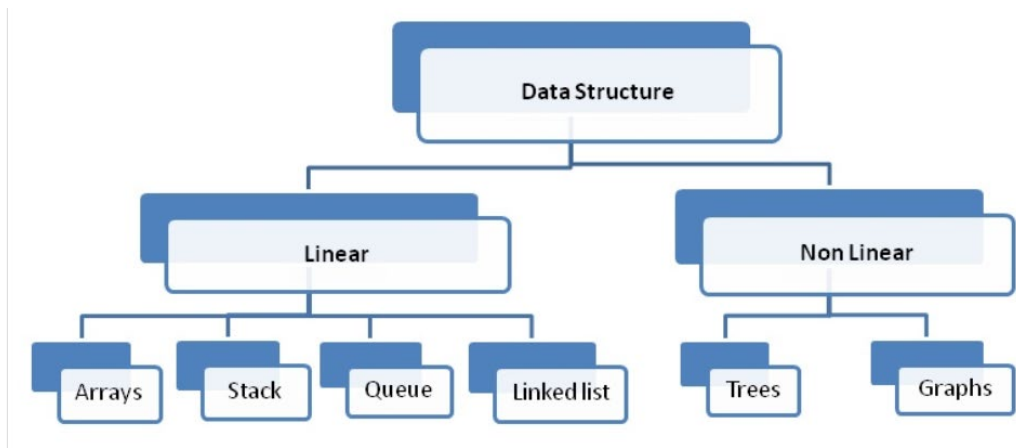


Figure 15.3: *Classifications of data structures*

Linear Data Structure

A data structure that preserves a linear connection among its data elements is known as a linear data structure. The arrangement of the data is made linearly, where each element consists of the successors and predecessors except for the first and the last data elements. The term *traversing* refers to the iterating over a collection of data. Data elements in a linear data structure are traversed one after the other and only one element can be directly reached while traversing. All the data items in linear data structure can be traversed in a single run. Examples of linear data structures include arrays, linked lists, stacks, and queues. We will study these data structures in future weeks.

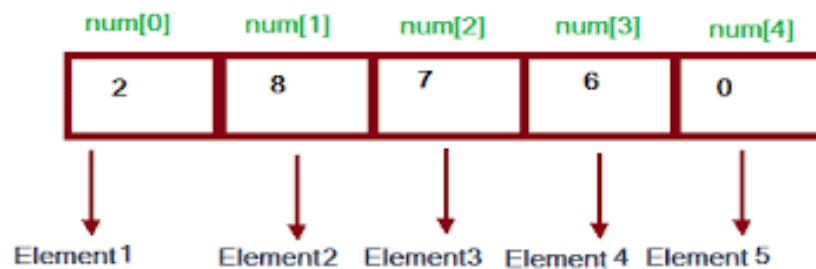


Figure 15.4: *An array named num(). An array is a linear data structure*

Non-Linear Data Structure

Non-linear data structures are those where the elements are not organised linearly, often storing data in a hierarchical or interconnected manner. All the data elements in a non-linear data structure cannot be traversed in single run. These data structures provide flexibility and efficiency for certain data organisation and manipulation tasks. Examples of non-linear data structures include trees and graphs.

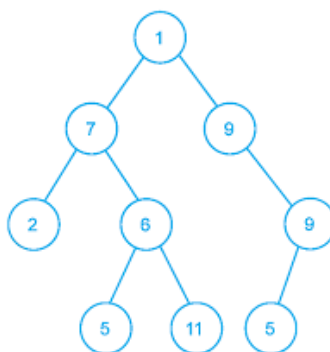


Figure 15.5: *A tree is a non-linear data structure*

Difference between a linear and a non-linear data structure

Linear Data Structure	Non-Linear Data Structure
Every item is related to its previous and next item.	Every item is attached with many other items.
Data is arranged in linear sequence.	Data is not arranged in sequence.
Data items can be traversed in a single run.	Data cannot be traversed in a single run.
Examples: Array, Stack, Queue, Linked List.	Examples: Tree, Graph.
Implementation is Easy.	Implementation is Difficult.

Static and Dynamic data structures

Based on memory allocation, data structures can also be classified into two types: static and dynamic.

Static data structures: these have a fixed size. The memory for these data structures is allocated at the compile time (i.e. when program code is converted into machine code), and the user cannot change their size after being compiled; however, the data stored in them can be altered. Examples of static data structures include arrays.

Dynamic data structures: these have a size that can change to accommodate different data requirements. The memory of these data structures is allocated at run time, and their size can vary during the code's execution. The user can change both the size of a dynamic data structure and the data elements stored in the data structure at run time. Examples of dynamic data structures include linked lists, queues, stacks, and trees.

Common data structures operations

Operations that can usually be performed on data structures are:

1. **Searching:** Looking for an element in a data structure.
2. **Sorting:** Putting the elements of a data structure either in ascending or descending order.
3. **Insertion:** Adding a new element to a data structure.
4. **Updating:** Replacing a data structure element with another element – see Figure 15:6.
5. **Deletion:** Removing the element from the data structure.

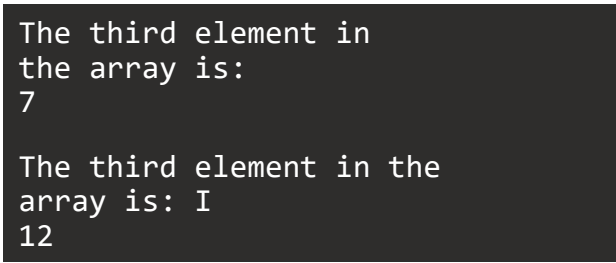
Python program	Output
<pre>num = [2,8,7,6,0] print('The third element in the array is:') print(num[2]) print() #print a blank line num[2] = 12 print('The third element in the array is:') print(num[2])</pre>	 <pre>The third element in the array is: 7 The third element in the array is: I 12</pre>

Figure 15.6: An example of a program where a data structure (an array) is updated.

An array can hold many values of the same data type, under a single name. In the Python program above (Figure 15:6), the num() array has the name 'num' and five elements of the *int* data type.

You can access the element in an array by referring to an index number. The first element in an array has the index 0, so the first element in the num() array can be referenced by num(0). Arrays will be studied in depth in Week 16.

Conclusion

Variables, data types, and data structures are three important interconnected concepts in programming. Variables are containers for data, and their data type define the kind of data they can hold. Data types define the characteristics of the data stored in variables. Data structures organise and manage variables of different data types, and these structures are widely used in various applications. There are a variety of data structures to choose from. Two possible classifications of these structures are linear and non-linear, and static and dynamic. Data structures enable the efficient storing, retrieving, and manipulating of data in computer programs.

Learning Tasks

Here are some tasks to help learners understand the three focal areas in week 15. Kindly take note of differentiated learning when applying these tasks.

Task 1 – Individual activity

Write down the data type of each item in list similar to the slide below.

Variable Types Activity	
Classify each of the following as <i>integer</i> , <i>string</i> , <i>Boolean</i> or <i>float</i> :	151
	Billy Bunter
	39.99
	GHANA
	0.25
	MR2
	Zebra
	-200
	True
	10 Yemen Street
-99.9	

Task 2 – Individual activity

Using the same list of data items as in Task 1, choose a suitable variable name to store the data item in Python.

Task 3 – Individual or Pairs activity

Study the following code or similar given by your teacher. Predict what the output will be and then run the code to check your predictions.

```
w = False
x = 1
y = 2.8
z = "hello"

print(type(w))
print(type(x))
print(type(y))
print(type(z))
```

The answers to Task 1 could also be checked this way.

Task 3 – Group activity

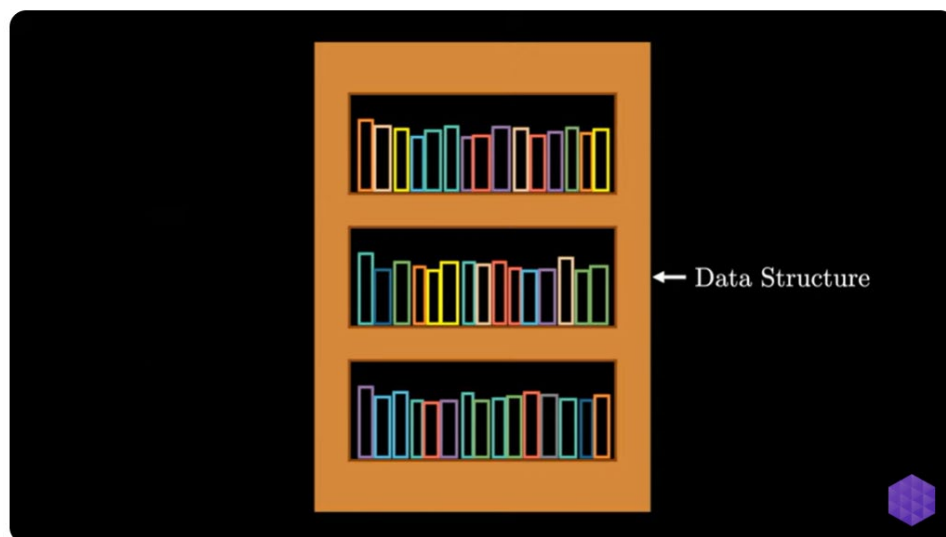
Work together to complete the following table to compare linear and non-linear data structures using the given criteria:

Criteria	Linear	Non-linear
Data arrangement		
Links between data items		
Traversing of data		
Implementation		
Examples		

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Using talk for learning, teachers can use a direct instruction approach to explain and guide learners to understand more about program variables, and introduce the concepts of data types and data structures. Explain the importance of these three concepts in programming and how they are interconnected.
2. Video pedagogy: there are many good videos for introducing data structures. The following YouTube video (approx. 4 minutes) uses the analogy of a library of books, and links data structures and algorithms. This video could lead to a discussion on what operations could be done on a library of books.



What Actually Is a Data Structure?

3. Teachers should guide the learners through the individual activities (Learner Tasks 1, 2, and 3). They should diagnose learners' responses and guide them to reflect on how their responses could be improved.
4. Teachers should integrate GESI, SEL and SEN in the groupings when organising Learner Task 4.



5. Scan for an image to use in a plenary session to end this week's lessons:
6. Learners should be encouraged to reflect on the focal areas over the past three weeks and how these areas are interconnected.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the three focal areas.

DOK Level 1: Recall/Reproduction

1. Data cannot be traversed in a single run in a linear data structure.
 - a. true
 - b. false
2. Identify the data type of the following values of the variable 'item':

<i>item</i>	Data type
24	
True	
78	
60.05	
Happy Birthday	
yes	
1.2	
ACCRA	

3. State the name of two data structures used in programming.
4. A data structure is a particular way of storing and organising data in a computer so that it can be used efficiently.
 - a. true
 - b. false
5. What is the name given to the type of data structure that changes in size as a program needs it by allocating and de-allocating memory?
6. A program uses a variable called *value*. Write the line of Python code that would output the data type of this variable.
7. Name two linear data structures used in programming.
8. Complete the sentence: Data structures are used to o _____ and store data.

9. Which of the following is not a data structure?
 - a. Linked list
 - b. Stack
 - c. Python
10. The type data structure shown in Figure 15:7 is

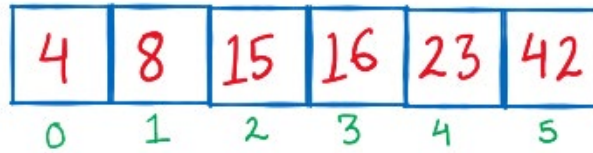


Figure 15.7

- a. Linear
- b. Non-linear

DOK Level 2: Skills and Concepts

1. Give one advantage of a linear data structure over a non-linear data structure.
2. Explain the difference between static and dynamic data structures, and give an example of each.
3. Describe how variables and data types are linked.

```
print("What is your name?")
firstname=input()
print("Hello,",firstname)
```

- a. How many integer variables are included in the above code?
- b. How many string variables are included in the above code?
4. Use the analogy of a box to explain what is meant by a computer variable.



5. Identify two features of efficient code.
6. Outline how the use of data structures can impact code efficiency.

DOK Level 3: Strategic Thinking

1. As a member of the computing club in your school, imagine you need to manage an ever-changing list of member names with frequent additions and deletions.
 - a. Which would be more suitable for this task: a static or a dynamic data structure? Justify your choice of answer.
 - b. Identify at least two operations that would be regularly performed on the chosen data structure.
 - c. Describe possible uses of these operations in this scenario.

DOK Level 4: Extended Thinking

1. Another classification of data structures is primitive and non-primitive. Research what is meant by this classification and write a short report on your findings.
2. Investigate how data structures are used by an operating system to manage memory, processes, and files.

Scan for other questions and solutions:



WEEK 16

Learning Indicator: *Differentiate between the types of Data Structures.*

Theme or Focal Areas:

1. Describe arrays as a type of data structure
2. Explain the differences between one-dimensional and two-dimensional arrays
3. Advantages and disadvantages of arrays

Key Concept Notes

Arrays

Definition and examples

Arrays are a collection of data elements of the same data type stored in contiguous (i.e. consecutive) memory locations. Arrays are used to store multiple values of the same data type in one single named variable. They offer fast access to elements through indexing.

The data elements of the array share the same variable name; however, each carries a different index number called a subscript. We can access any data element in the array using the array name and the subscript of the element. A key feature of the arrays to understand is that the data is stored in contiguous memory locations, making it possible for the users to traverse through the data elements of the array using their respective indexes.

Array variables provide an alternative to defining multiple variables to represent multiple values. For example, rather than having six different integer variables to store the values 2, 6, 11, 7, 18, and 4, a variable array as shown in Figure 16:1 can be used.

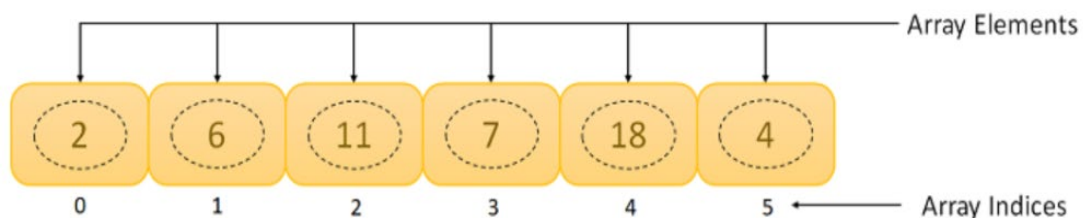


Figure 16.1 An array of integers

If this array was implemented in Python using the array variable name *numbers*, the code in Figure 16:2 could be used. *print* statements are included to show how array indexing works.

```
numbers = [2,6,11,7,18,4]
print(numbers)
print()
print(numbers[0])
print(numbers[5])
```

```
[2,6,11,7,18,4]
```

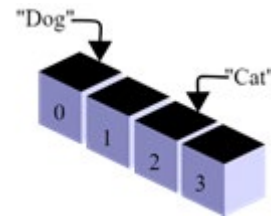
```
2
4
```

Figure 16.2: *Numbers program, version 1 - sets up an array of six integers and outputs the array and two elements*

An example of an array of string elements is given in Figure 16:3. Two of the array elements are empty strings. The second of the print statements contains text to label the output. The # symbol in Python Language is used as a comment. If you want to write something in program and do not want to execute it, then one way to do this is to precede it with the # symbol.

```
animals = ["Dog", "" "Cat", ""]
print(animals[2]) #this would output cat
print("The first animal stored in the array is",
animals[0])
```

```
Cat
The first animal stored in the array is Dog
```



Arrays are widely used in various algorithms to store lists of related information. Examples include a shopping list (type = string), a list of the names of students in a class (type = string), a list of the number of goals scored by a football team over 12 games (type = integer), a list of the answers to a survey question returned by 20 people indicating if they agree with a debate motion (type = boolean), and a list of the amounts collected by a charity in a collection box at five different churches (type = float).

When referring to an array variable, the name often includes the number of possible data items the array can hold in parentheses. The array variable score(9) would allow ten data items to be stored. The array variables, *numbers* and *animals* in Figure 16:2 and Figure 16:3, could be referred to as *numbers(5)* and *animals(3)* respectively.

Why use arrays?

Arrays are one of the simplest and most widely used data structures. Because arrays are so commonly used in programming, they are often the first data structure computing students study when learning programming.

Using an array can simplify a program by storing related data under one name. This means that a program can be written to search through an array of data and sort an array much more quickly than having to write a new line of code for every variable. This reduces the complexity and length of the program which makes it easier to find and debug errors.

Figure 16:4 shows an alternative program to the program shown in Figure 16:2 that will produce the same output. Notice that this version of the program has more lines of code and six variables rather than one making it less efficient.

```
number1 = 2
number2 = 6
number3 = 11
number4 = 7
number5 = 18
number6 = 4
print(number1, number2, number3, number4, number5, number6)
print(number1)
print(number6)
```

```
2,6,11,7,18,4
2
4
```

Figure 16.4: *Numbers program, version 2 – uses six different variables to store six integers*

Arrays and memory (more advanced optional content)

To understand how arrays work on a low level in memory, you can think of an array as a group of apartments (elements) side by side in an apartment complex (the array). Each apartment has an associated address number (memory address) at its front door. For instance, if the first apartment has an address number of 1000, then the second apartment will have an address number of 1001, and so on. Following this logic, we know that the fifth apartment will have an address number of 1004 and the tenth apartment will have an address number of 1009. A similar phenomenon happens in arrays. Since the elements of an array are stored in contiguous memory locations, if the memory address of the first element of an integer array is 1024, then the memory address of the fifth element ($=5$) will be $1024 + 4 \times 4$ (the size of an integer element is 4 bytes) = 1040. This means that as long as we keep track of the memory location of the first element of the array, we can find any element in the array with a quick calculation (see Figure 16:5 below).

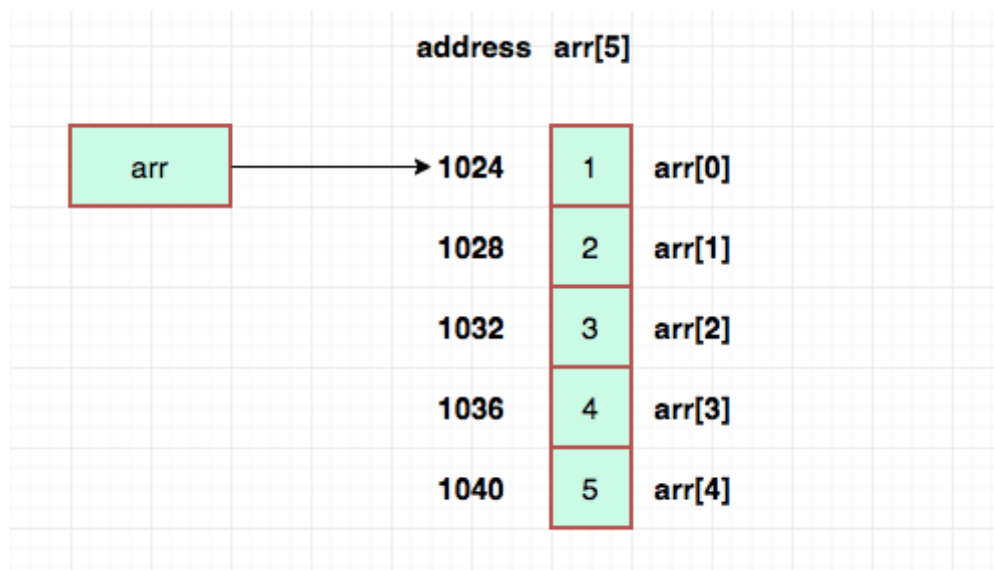


Figure 16.5: An integer array variable called *arr* with five elements.

Comparing One-Dimensional and Two-Dimensional Arrays

Two types of arrays are one-dimensional arrays and multi-dimensional arrays. Arrays can have any number of dimensions and a multi-dimensional array is basically an array of arrays. In this manual we will just cover the most common multi-dimensional array, a two-dimensional/2D array.

1. **One-dimensional array:** This is the type of array that we have studied so far. This structure has only one row of data elements which are stored in an ascending storage locations. Unless otherwise stated, when the term ‘array’ is used, it will refer to a one-dimensional array.

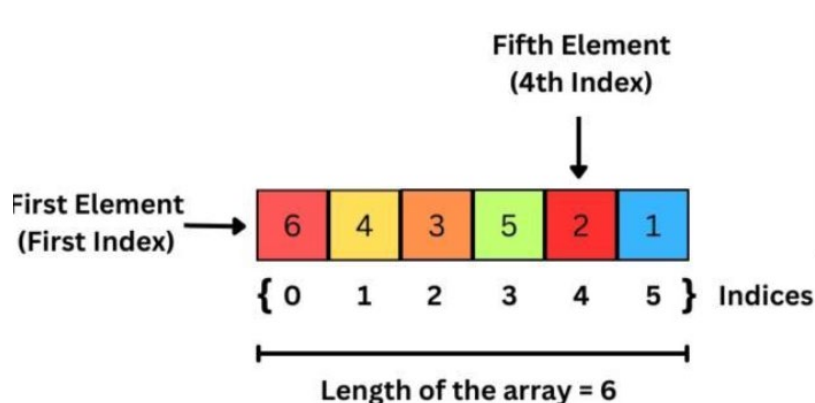


Figure 16.6: A one-dimensional array of integers

Parallel arrays use two or more arrays to represent a collection of data where each corresponding array index is a matching field for a given record. For example, if there are two arrays, one for names and one for ages, the array elements, `names[5]` and `ages[5]` would be the name and age of the sixth person - Baba is the sixth person and she is aged 12, as shown in Figure 16:6. This is one way to use arrays to store data of different types.

names	Alberta	Angela	Kofi	Oko	Yorkoo	Baba
ages	12	13	12	14	13	12

Figure 16.6: A set of two parallel arrays

2. **Two-dimensional array:** This is an array that consists of multiple rows and columns. The example of a two-dimensional array shown in Figure 16:7 consists of three rows and four columns. Each element is accessed by two indices (r,c), one for row and one for column. The value of the array element at

		Column Index			
		0	1	2	3
Row Index	0	1	2	3	5
	1	7	11	13	17
	2	19	23	29	31

Figure 16.7

- (0, 2) is 3
- (1,1) is 11
- (2, 0) is 19

A 2D array is also known as a matrix (a table of rows and columns).

Two-dimensional arrays are an alternative to parallel arrays, having the advantage of quicker processing but the disadvantage that all elements must be of the same type.

Suppose a school wanted to store the marks registered by five students (Daniel, Kojo, Beatrice, Daniel, and Emmanuel) for the following four subjects: Computing, Science, Mathematics, and English. This requires storing the marks in a tabular form comprising of four rows and five columns.

Daniel	Kojo	Beatrice	Daniel	Emmanuel
Computing				
Science				
Mathematics				
English				
10	55	90	70	60
50	60	75	65	95
95	75	55	85	45
35	95	65	55	75

Figure 16.8

Implementation of a 2D array in Python (more advanced optional content)

There are different ways of implementing a 2D arrays in Python. One way of implementing the array shown in Figure 16:8 is shown in Figure 16:9. The program includes code to update one element in the array. The output from this program is shown in Figure 16:10.

```
marks = [ [10,55,90,70,60], [50,60,75,65,95],
          [95,75,55,85,45], [35,95,65,55,75]

print(marks[1][2]) #outputs the element in the second
row and third column

marks[1][2] = 92 #changes the value of the element in
the second row and third column

print marks[1][2])
print (marks) #outputs the whole array
```

Figure 16.9: Python program using a 2D array to represent data shown earlier

```
75
92
[[10,55,90,70,60], [50,60,75,65,95], [95,75,55,85,45], [35,95,65,55,75]]
```

Figure 16.10: Output from the program given in Figure 16.9

Applications of 2D arrays

These applications include:

1. **Tabular Data Representation:**
 - See examples in Figures 16:7 and 16:8.
 - Representing tabular data, such as spreadsheets or database tables.
2. **Game Boards and Maps:**
 - Game boards, whether for chess (8X8 matrix), or other games, can be modeled using 2D arrays.
 - Maps in video games (top-down or side views) are often represented as 2D arrays, where each cell corresponds to a specific location.
3. **Computer Graphics and Image Processing:**
 - Each pixel's colour in a bitmapped graphic can be stored in a 2D array
 - Image processing algorithms often manipulate pixel values using 2D arrays.

Advantages and Disadvantages of Arrays

The advantages of arrays include:

1. **Efficient access:** You can retrieve an element by its index quickly.
2. **Memory efficiency:** Arrays allocate memory in a contiguous block, which minimises memory overhead
3. **Ordered collection:** Arrays maintain the order of elements, which is crucial when dealing with sequences, lists, or data sets where element order is meaningful.
4. **Simplicity:** Arrays are straightforward to use and implement in most programming languages

The disadvantages of arrays include:

1. **Fixed size:** Arrays generally have a fixed size that must be specified during declaration. This limitation can be problematic when the size of data is dynamic or unknown.

2. **Same type data:** Arrays typically store elements of the same data type. If you need to store mixed data types, other data structures will be required.
3. **Inefficient insertions and deletions:** Adding or removing elements from the middle or beginning of an array can be inefficient, as it often requires shifting other elements to accommodate the change.
4. **Memory allocation:** Arrays pre-allocate memory based on their declared size. This can lead to wasted memory if the array is larger than needed or insufficient memory if it's smaller than required.

Learning Tasks

Here are some tasks to help learners understand the three focal areas in week 16. Kindly take note of differentiated learning when applying these tasks.

Task 1

With an allocated time period, suggest lists of items of the same type that a computer program may need to store. One possible list is a list of times taken (in minutes) by a ten runners in a race,

Task 2

Using some of the lists identified in Task 1, suggest a suitable data type and array variable name for use in a Python program. The type and suitable name for the example given in Task 1 would be *float* and *race_times* (*race_times(9)*)

Task 3

Repeat Task 1 and Task 2 for tabular data rather than a list. An example is a table giving the daily rainfall readings (in mm) for a town over a four week period.

Task 4

Issue a list of problem specifications which the learners have to label 1D or 2D depending on which type of array that could be used to solve the problem. Two examples are:

- Modelling a game of tic tac toe
- Processing a set of test marks registered by a class to find out the number of passes

Task 5

Using a set of cards, each with a number representing a data element in a 1D array. Arrange the cards in the line and then talk through how the following operations could be performed:

- Searching for a particular element/number that is in the array/line.
- Searching for a number that is not in the array
- Sorting the array in descending order
- Updating one of the elements to a different value

Task 6

Using the same list of cards as in Task 5, explain how the following could be performed:

- Inserting a new element/number in the middle of the array
- Inserting a new number at the end of the array

- Deleting the first number in the array
- Making a duplicate of the whole array

Task 7

Write out at least three advantages and three disadvantages of the array data structure (weaker learners could be given lists of features such as the data elements must be the same type, which they label either an advantage or disadvantage.)

Task 8

Create a slideshow summarising the focal areas for this week's lessons. Include a description of how these focal areas link to previous focal areas on data types and algorithms.

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Using talk for learning, teachers can use a direct instruction approach to explain and guide learners to understand more about arrays and what they are used for in programming. A sound understanding of 1D arrays should be demonstrated before introducing 2D arrays.
2. Practical demonstrations:
 - When describing a 1D array use a set of playing cards or a set of boxes with data elements (on card) inserted. Talk through how the array could be searched and sorted (note similarity to Learner Task 4). A line of learners, each holding a different playing card from the same suit would work better for some learners.
 - 2D arrays can be a little confusing for beginners. An empty egg carton to demonstrate how a 2D array is traversed may help when teaching this data structure.



In a two-dimensional array, each row represents a distinct array of elements. As pictured above, the carton (array) holds two rows of eggs (elements), so it contains two distinct arrays, specifically [0, 1, 2, 3, 4, 5] and [6, 7, 8, 9, 10, 11]. In class, the learners could place different coloured (or numbered) sweets into different spaces of an egg carton to simulate a piece of array-accessing code step by step. The learners could eat the sweets afterwards!

- When the learners have grasped the basics of what arrays are using real-life examples, the teacher should demonstrate how they can be implemented in Python (or other chosen programming language). Use the programs in Figures 16:2 and 16:4 to illustrate the reason for using arrays. An explanation of what is meant by efficient code should be given at a suitable level for SHS Year 1. Note that learners will be given the opportunity to implement algorithms using arrays in later weeks but, if appropriate, some practical programming tasks could also be included this week.
3. Brainstorming activities as indicated in Task 1 and Task 3 could be used when introducing 1D and 2D arrays, and later in the week to identify the type of data elements and suitable array names.

4. The teacher should guide the learners through the Learner Tasks and encourage them to reflect on their responses and how and if they could be improved. Learner Tasks 4, 7, and 8) could be done individually or in groups.
5. Teachers should integrate GESI, SEL and SEN in the groupings when organising Learner Task 5. Provide slightly more challenging tasks, such as Learner Task 6 for the more able learners, or group the learners carefully so that some group members can teach the other how to complete the task.
6. On the last lesson for this week, some learners could present their summary slideshows (Learner Task 8).

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the three focal areas.

DOK Level 1: Recall/Reproduction

1. What is an array?
 - a. A list of variables
 - b. A list of strings
 - c. A series of memory locations, each with the same name, that hold related data
2. What is the value of element *amounts*(2) in the array *amounts*: (6, 10, 7, 12, 9, 2)?
3. Each element in an array will have the same data type.
 - a. true
 - b. false
4. How many elements would the array *shopping_list*(12) hold?
5. What would be the most appropriate data type for an array of shopping items?
6. When creating the array *scores*(9), what does the 9 in brackets represent?
 - a. The number of elements the array can hold
 - b. The largest number of digits an element can hold
 - c. The largest number of characters an element can hold
7. State the name of the type of array that can be used by a program to represent a chessboard.
8. In the array *colours*(“Purple”, “Blue”, “Red”, “Green”, “Yellow”), how would you reference the element that has the element “Purple”?
9. Name the type of array that consists of multiple rows and columns.
10. A reason why arrays are used?
 - a. Arrays store more data than if we were using a single variables
 - b. Arrays store more data than if we were using strings
 - c. Arrays store multiple data values under one name, reducing the program complexity

DOK Level 2: Skills and Concepts

1. Describe what is meant by an array in programming.
2. Describe how data is stored in a two-dimensional array.

3. Compare and contrast a one-dimensional array to a two-dimensional array.
4. Create a slideshow which demonstrates some applications of arrays. Include images and video links.

DOK Level 3: Strategic Thinking

1. A data structure is needed to store the monthly temperatures over a three month period as shown in Figure

Jan	15	15	16	19	23	28	30	31	28	24	20	17
Feb	33	35	32	21	26	30	15	15	15	18	19	14
Mar	18	12	11	11	14	21	23	30	28	13	17	19

Average monthly temperatures for Jan, Feb and March

Figure 16.11

- a. Recommend a data structure to use and give a reason for your choice.
- b. Suggest an alternative grid layout that could be used to store this data.
2. As a Year 1 Computing learner, you are tasked to store data for the sitting plan for your class. Which type of array would you choose to use and why?
3. Evaluate the ease of performing the following operations on an array:
 - a. Accessing a data element
 - b. Modifying an element
 - c. Adding a new element
 - e. Removing an element
 - f. Sorting the array
 - d. Searching for a specific data item in the array

DOK Level 4: Extended Thinking

1. Research jagged arrays and create a short summary of your findings. Include examples of jagged arrays in your summary.

Scan for some solutions:



WEEK 17

Learning Indicator: *Differentiate between the types of Data Structures*

Theme or Focal Areas

1. **Linked lists – features, operations, examples, types, applications, advantages and disadvantages**
2. **Stacks – features, operations, examples, types, applications, advantages and disadvantages**

Key Concept Notes

Linked Lists

A linked list is a linear data structure whose data items have a link to the next data item in the list, allowing it to expand and contract dynamically. An entry/element in a linked list is generally called a node will consist of data and a pointer to the next item. The pointer field/part will be the address of the subsequent node(s) in the list.

The linked list in Figure 17:1 has three nodes. The linked list in Figure 17:2 has six nodes.

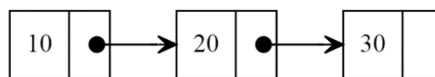
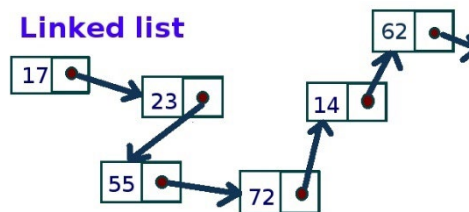


Figure 17.1



data format

Figure 17.2

The pointer of the last node of the linked list consists of a *null* pointer, as it points to nothing, and the entry point into a linked list is called the *head* of the list. – see Figure 17:3. The head is not a separate node, but the pointer to the first node. If the list is empty then the head is a null reference.

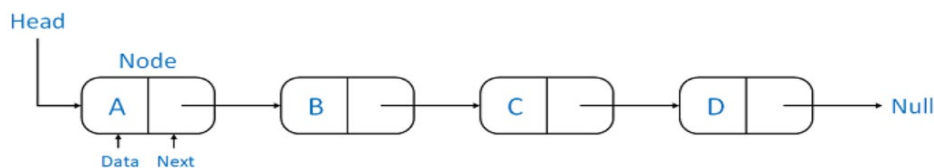


Figure 17.3

Unlike Arrays, the user can dynamically expand and contract the size of a linked list as per the requirements.

How are nodes added and deleted

In the example in Figure 17:3, deleting item B from the list would be achieved by changing the link from item A to point to Item C. As a result, the memory used by item B would then be marked as free space.

Inserting an item X between C and D would be achieved by changing the link of the pointer from C to point to the memory location of X and then linking the pointer from X to the memory location of D.

Similarly adding an item to the end of the list can be done by redirecting the link from D to the new item and linking the new item to Null. Adding an item to the beginning of the list can be done by redirecting the Head of the list to the new item and pointing its link to the memory location of item A.

Types of linked lists

Two types of linked lists are singly linked lists and doubly linked lists:

1. **Singly Linked List:** The examples shown earlier are single linked lists. These are uni-directional as they can only point to the next node in the list but not to the previous,
2. **Doubly Linked List:** A doubly linked list consists of a data field and two pointer fields. The first pointer field contains an address of the previous node, whereas the second pointer field contains a reference to the next node. – see Figure 17:4. Thus, traversing a doubly linked list can happen in both forward and backward directions.

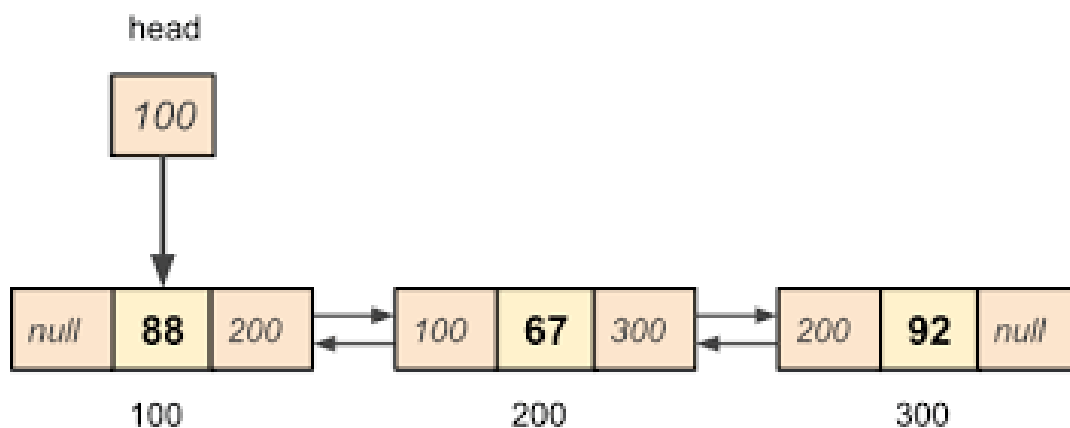


Figure 17.4: An example of a doubly linked list with pointers giving the addresses of the linked nodes

An advantage of doubly linked list allows elements to be traversed two ways. Singly linked list allows traversal of the elements only in one way. Disadvantages of doubly linked list include greater memory requirements (two pointers rather than one pointer per node) and more code is required for implementation.

Some Applications of Linked Lists

1. To implement other data structures such as stacks, queues, binary trees, and graphs of predefined size.
2. Task scheduling by a computer's operating systems, where each process waiting to be executed is represented as a node in the list.
3. Previous and next page in a web browser – the previous and next URL searched in a web browser can be accessed by pressing the back and next buttons since they are linked as a linked list.
4. Music Player – songs in the music player app are linked to the previous and next songs.
5. GPS navigation systems - linked lists can be used to store and manage a list of locations and routes, allowing users to easily navigate to their desired destination.
6. Manipulation of polynomials by storing constants in the node of the linked list

Advantages and Disadvantages of Linked Lists

Advantages include:

- A linked list is a dynamic data structure, which means that its size is not fixed, and can grow and shrink during execution to fit the data set.
- A linked list is very flexible as the order of items in a linked list can be changed without actually moving any data around, just the links between them change.
- A linked list is more memory efficient than an array because it only needs to be as large as the number of items to be stored, not as large as the total possible number of items to be stored.

Disadvantages include:

- With a linked list structure, it is not possible to identify a specific item directly using its index in the way that you can when storing data in an array. To identify the position of an item in a linked list you have to walk through the list from beginning to end.

Stacks

A Stack is a linear data structure that follows the LIFO (Last In, First Out) principle that allows operations like insertion and deletion from the top of the stack. Stacks can be implemented with the help of contiguous memory, an array, and non-contiguous memory, such as a linked list.

Real-life examples of stacks are piles of books, a deck of cards, stack of plates, and many more. Figure 17:5 represents a real-life example of a stack where the operations are performed only from the top, such as inserting and removing new books.

The stack is an area of memory which is bounded by 'Bottom of Stack' and 'End of Stack' pointers. The 'Stack Pointer' (SP) points to the current top location occupied in the stack



Figure 17.5: A stack of books

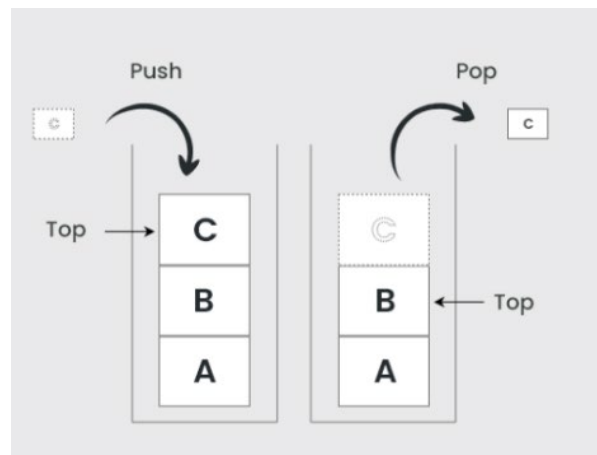


Figure 17.6: Stack operations

The primary operations in the stack

Push: is the operation to insert a new element in the stack.

Pop: is the operation to remove or delete elements from the stack.

See Figure 17:6 and Figure 17:7 for examples of how these two operations work.

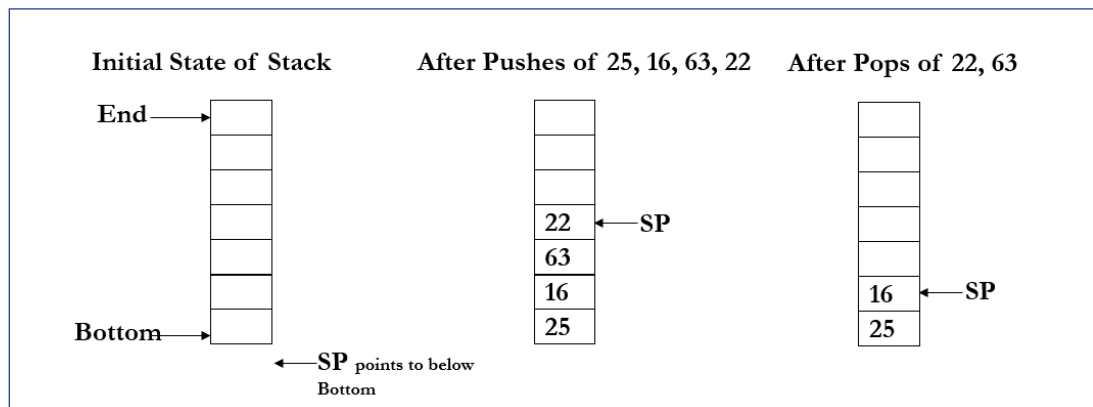


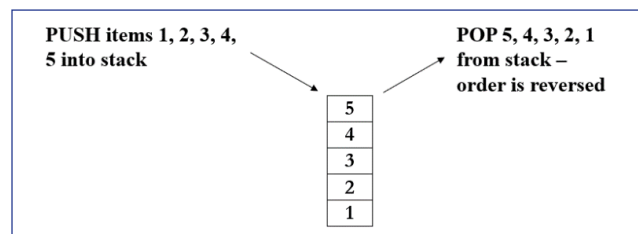
Figure 17.7: An example of stack push and pop operations

When data is added to the stack, it must not go beyond the End pointer. If it does, a *Stack Overflow* occurs, and the program terminates

When data is removed from the stack, it must not be removed below the Bottom pointer. If it does, a *Stack Underflow* occurs and the program terminates

Some applications of stacks

1. To reverse the order of a list of items – see Figure 17:8.
2. For the Undo and Redo functions in an edit.
3. For reverse polish notation (RPN) is used by many calculators (without an = key). You enter the calculations as two data items followed by an operator – see Figure 17:9.
4. To manage memory allocation in some operating systems and programming languages.
5. To keep track of the return addresses of function calls in a program, allowing the program to return to the correct location after a function has finished executing.



Calculate $(6+2) \times (5-1)$

This is entered in the calculator as **6 2 + 5 1 - x**

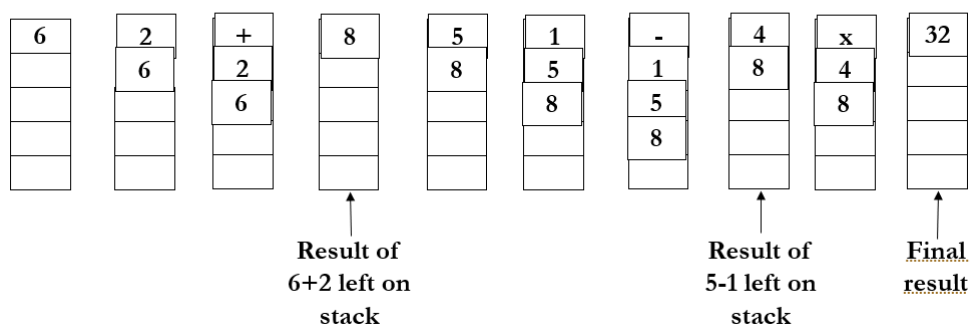


Figure 17.9: An example of RPN using a stack

Advantages and Disadvantages of Stacks

Advantages include:

- **Simplicity:** Stacks are a simple and easy-to-understand data structure, making them suitable for a wide range of applications.
- **Last-in, First-out (LIFO):** Stacks follow the LIFO principle, ensuring that the last element added to the stack is the first one removed. This behavior is useful in many scenarios, such as function calls in programs.
- **Limited memory usage:** Stacks only need to store the elements that have been pushed onto them, making them more memory-efficient compared to some other data structures.

Disadvantages include:

- **Limited access:** Elements in a stack can only be accessed from the top, making it difficult to retrieve or modify elements in the middle of the stack.
- **Potential for overflow:** If more elements are pushed onto a stack than it can hold, an overflow error will occur, resulting in a loss of data.
- **Not suitable for random access:** Stacks do not allow for random access to elements, making them unsuitable for applications where elements need to be accessed in a specific order.
- **Limited capacity:** Stacks have a fixed capacity, which can be a limitation if the number of elements that need to be stored is unknown or highly variable.

Learning Tasks

Here are some tasks to help learners understand the two focal areas in week 17. Kindly take note of differentiated learning when applying these tasks.

Task 1

Create a PowerPoint slideshow that clearly explains the following:

- What is meant by a singly linked list with examples and possible real-life applications.
- What is meant by a doubly linked list with examples and possible real-life applications.
- The relative advantages and disadvantages of these two types of linked lists.

Use suitable diagrams, images and video links in your slideshow.

Present your slideshow to the class.

Task 2

Choose one of the following linear data structures and create a poster to illustrate how it looks and operates.

- Arrays
- Linked Lists
- Stacks

Task 3

Identify at least one difference between the following pairs of data structures:

- 1D array and 2D array
- Singly linked list and doubly linked list

- A 1D array and a singly linked list
- A 1D array and a stack
- A singly linked list and a stack

Task 4

Identify the most appropriate data structure from those studied so far for a given list of tasks. Two possible tasks are:

- To reverse a list of numbers
- To program the Back and Forward button on a web browser
- A seating plan for a classroom

Task 5

List at least two advantages and two disadvantages of the following data structures:

- Linked lists
- Stacks

Task 6

Discuss how the choice of a data structure affects performance for operations like insertion or deletion of data elements.

Task 7

Research how linked lists and stacks can be implemented in a programming language that you are familiar with.

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Direct instruction: the teacher should guide learners in the study of linked lists and stacks using visual aids such as
 - Projected diagrams and images
 - Playing cards or a set of cards with printed data items arranged to suit each of the data structures

Note that programs to implement these two data structures are beyond the scope of the Year 1 SHS Curriculum.

Use these visual aids to explain how operations such as inserting and deleting work on these structures.

2. Learner activities: Individual activities such as Learner Task 2 and group activities such as the other Learner Tasks. Note that Task 7 would only be suitable for the very proficient programmers. The teacher should ensure each group has a balance of learners (GESI, SEN, SEL, research proficiency) and guide the learners to reflect on their responses and how they can be improved.
3. Open class discussion after the slideshow presentations (Task 1) on how the choice of linked list affects performance for operations like insertion and deletion. The teacher should guide learners to consider factors like time complexity, memory usage, and ease of implementation when choosing a data structure.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the two focal areas.

DOK Level 1: Recall/Reproduction

1. What is a node used for in a linked list?
 - a. To check for the end of the list
 - b. To store the data and the link to the next item
 - c. Not used in a linked list
2. Name two basic operations that can be performed on a stack.
3. In a doubly linked list, each node contains a pointer to the next node and the node after that one.
 - a. true
 - b. false
4. Complete the sentence: In a stack, if a user tries to remove an element from empty stack it is called stack _____.
5. One way to implement a stack in a program is using a linked list.
 - a. true
 - b. false
6. A stack is a LIFO data structure. What does LIFO stand for?
7. A linked list is different from an array because
 - a. An array cannot be sorted, but a linked list can be
 - b. An array is fixed in size, but a linked list is dynamically sizable.
 - c. A linked list can handle more types of information than an array.
8. Delete where a choice of word or phrase is given for accurate statements:
 - Compared to arrays, linked data structures allow more/less flexibility in organising data and in allocating space for it.
 - Compared to arrays, the insertion and deletion operations linked data structures are easier/more *difficult* to implement.
9. What is the name of the operation that puts a data item on a stack?
10. A stack is a data structure that implements movement of data in which format?
 - a. First in first out
 - b. Last in first out
 - c. Random access
11. Study the stack in Figure 17:10 and answer the questions that follow:
 - a. Which data element will the Stack Pointer (SP) point to?
 - b. Draw the stack in Figure 17:10 after the following operations have been completed: POP, POP, POP

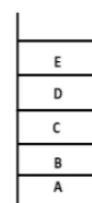


Figure 17.10

12. Name a data structure in which elements can be inserted and removed at the same end.

DOK Level 2: Skills and Concepts

1. Describe the structure of a doubly linked list. Use a diagram to support your description.
2. Describe how a singly linked list can be navigated.
3. Describe what is meant by a stack using real-life examples.
4. Explain how a data item in a linked list can be deleted.
5. Given the output stream A, B,C,D,E,F. Write down the sequence of operations (Push and Pop) on a stack which would produce the sequence C,B,D,E,F,A.
13. a. Use a diagram to explain clearly how a stack could be used to reverse the list of numbers:18, 56, 3, 78, and 11.
b. Describe one other possible application of a stack.
14. Explain why insertions and deletions of data items are faster in a linked list compared to an array.
15. Create a slideshow which demonstrates some applications of both linked lists and stacks. Include images and video links.
16. Use diagrams to explain how a stack could be used to evaluate $(5+3) * 7$ using RPN.

DOK Level 3: Strategic Thinking

1. Recommend a data structure to use for implementing an undo feature in a text editor. Give a reason for your choice of data structure.
2. Explain why the memory overhead of a linked list is higher than an array.
3. Explain why it is more efficient to insert and delete data elements in a linked list compared to a stack.
4. Which type of the linked list would be more efficient if you frequently insert new nodes at the beginning of the list? Justify your answer.

DOK Level 4: Extended Thinking

1. Investigate what is meant by a circular type of linked list. Write a short summary of your findings. Include possible applications of a circular linked list.
2. Research what is meant by the time and space complexities of data structures. Use this information to analyse the time and space complexities of the following:
 - Stack push and pop operations
 - Inserting a data item at the beginning of a singly linked list
 - Inserting a data item at position N in a singly linked list
 - Accessing an item in an array

Scan for some solutions:



WEEK 18

Learning Indicator: *Differentiate between the types of Data Structures*

Theme or Focal Areas

1. **Queues – features of a linear queue, operations, examples, applications, advantages**
2. **Identify and describe the non-linear data structure, binary tree**
3. **Identify and describe the non-linear data structure, graph**

Note that an in-depth study of Tree and Graph data structures is beyond the scope of this course at this level.

Key Concept Notes

Queues

A queue is a linear data structure that looks like a stack, but differs in how elements are added and removed. The adding of an element in a queue is done at one end, and the removal is done at another or opposite end. Thus, we can conclude that the queue data structure follows the FIFO (First In, First Out) principle to manipulate the data elements.

Some real-life examples of queues are a line at a voting booth or a ticket counter - see Figure 18:1. The first person who joins the queue at the voting centre gets to vote first, as opposed to the last person who joins the queue. Similarly, at a movie ticket counter, the customer who comes first is always served first. The customer arriving last will be served last.

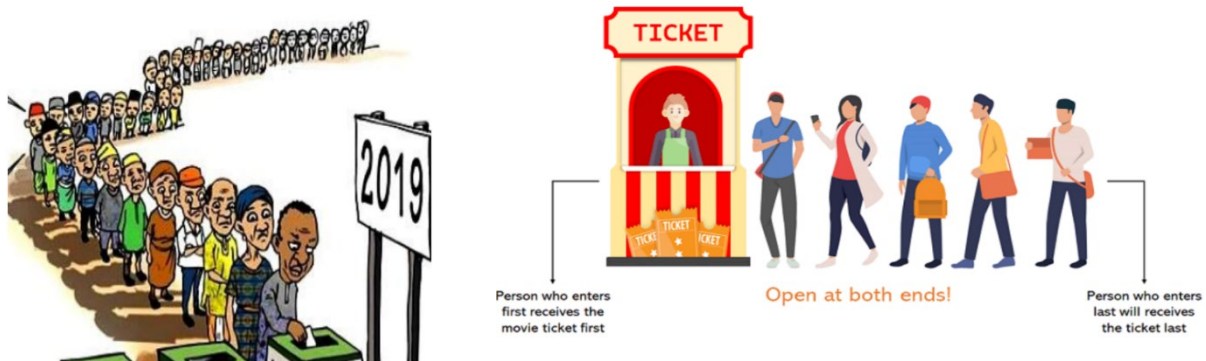


Figure 18.1: *Real-life examples of queues*

A queue data structure can be linear or circular. This course will only focus on a linear queue.

Primary operations of the Queue

1. **Enqueue:** *Enqueue* is the insertion or addition of some data elements to the queue. The element insertion is always done with the help of the rear or end pointer.
2. **Dequeue:** Deleting or removing data elements from the queue is termed *dequeue*. Deleting the element is always done with the help of the front or head pointer.

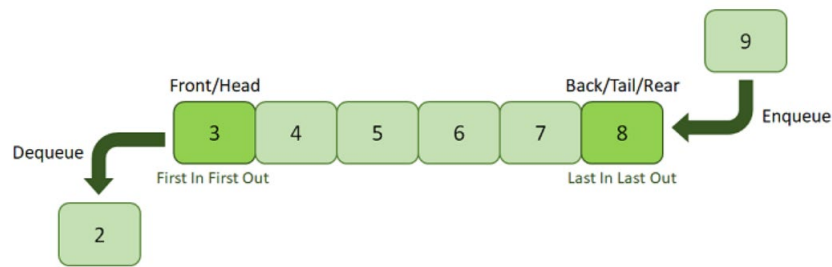


Figure 18.2: *An examples of a queue*

A linear queue can be illustrated as a horizontal collection of data (see Figure 18:2) or a vertical collection (see Figure 18:3).

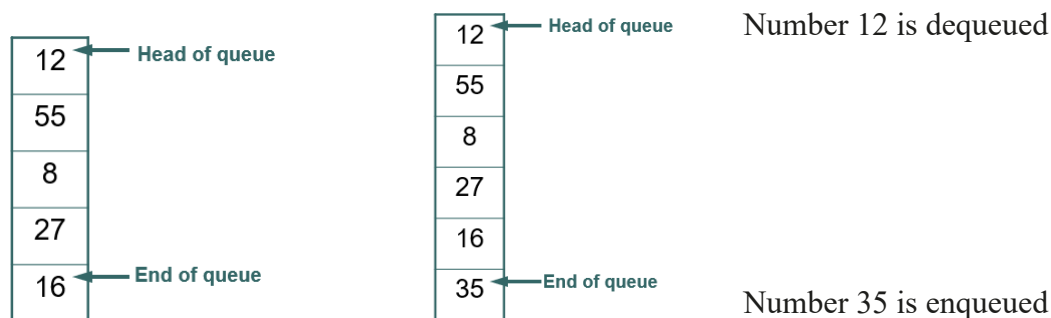


Figure 18.3: *Another example of a queue, and the adding and removing of elements*

An important aspect to realise here is that the data itself does not move but merely the pointers to the head and end of the queue. The 'Head Pointer' and 'End Pointer' are two dynamic pointers. So, unlike a stack, a queue does not have a fixed end or start point. A queue will be constrained within an area of memory bounded by two pointers (Head/Front and End/Back/Tail/Rear).

Queues can be implemented using arrays or linked lists.

Some Applications of Queues:

1. *Printer queues:* printers employ queues to manage print jobs. Jobs are added to the queue upon submission, and the printer processes them sequentially.
2. *Task scheduling:* queues are used to schedule tasks based on priority or the order in which they were received. For example, a task management system might use a queue to ensure that high-priority tasks are executed promptly.
3. *Operating systems:* operating systems often rely on queues to manage processes and resources such as CPU time.
4. *Traffic management:* Transportation systems (e.g., airport control or road networks) use queues to manage traffic flow, helping regulate the movement of vehicles or passengers.
5. *Network Protocols:* TCP and other network protocols use queues to manage transmitted packets. Queues ensure correct packet delivery order and appropriate transmission rates.
6. *Computer memory:* Certain types of computer memory use a queue data structure to hold and process instructions. For example, in a computer's cache memory, the fetch-decode-execute cycle of an instruction follows a queue. The first instruction fetched is the first one to be decoded and executed, while new instructions fetched are added to the rear.

The advantages and disadvantages of the queue data structure are not included at this level. It is worth noting, however, that a main advantage of queues is order preservation. Queues follow the First in

First Out (FIFO) order. This means that the element inserted first will be the first one to be removed. This property is useful in scenarios where maintaining order matters, such as scheduling tasks or processing requests.

Non-Linear Data Structures

Non-linear data structures are those that store data in a hierarchical or networked order, where each data element can have multiple predecessors and successors. You can access the elements by following different paths or relationships.

Binary Trees

A tree is a non-linear hierarchical data structure that organises and stores data in a way that allows for efficient navigation and searching. It consists of nodes with data connected by edges, forming a branching structure.

A binary tree is a tree where each node has at most two child nodes. For example, in Figure 18:4, the node 6 has two child nodes, 5 and 11, and the node 9 has one child node, 4.

One use of tree data structures is by search engines to organise and index web pages.

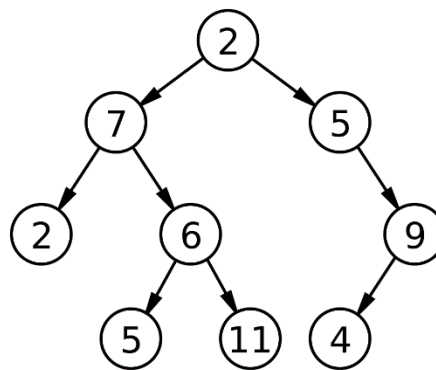


Figure 18.4: *Binary Tree*

Graphs

A graph is another example of a non-linear data structure. It is a collection of nodes with data elements/objects and are connected to other nodes. These nodes can also be referred to as vertices.

One use of a graph structure is to model relationships between users of social media platforms like Facebook and X. The users are represented as nodes, and friendships or connections between them are represented as edges. – see Figure 18:5

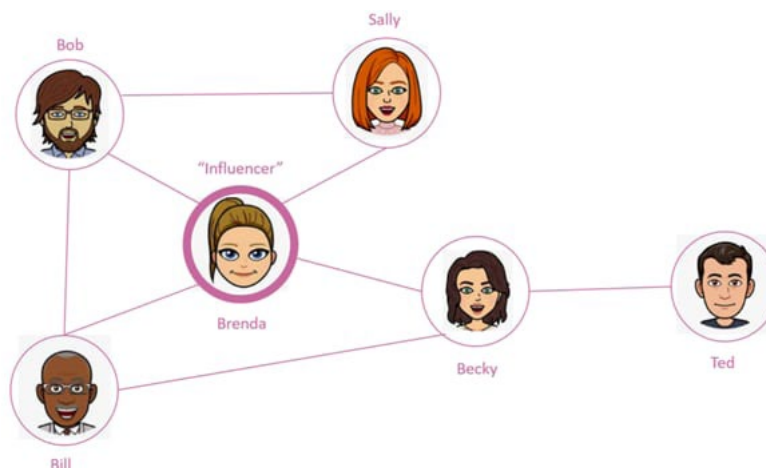


Figure 18.5: *Social Media Graph*

Comparing Linear Data Structures and Non-Linear Structures

When choosing between linear and non-linear data structures, it is important to consider factors such as memory usage, access time, insertion and deletion, and search and sort. Linear data structures typically use less memory than non-linear data structures, but have slower access time often due to their fixed size and structure. Non-linear data structures may have faster access time but require more memory to store pointers or references. Additionally, linear data structures tend to have slower insertion and deletion compared to non-linear data structures, as well as simpler search and sort algorithms. On the other hand, non-linear data structures may have faster insertion and deletion due to their flexible structure, as well as more complicated search and sort algorithms due to their multiple paths.

Learning Tasks

Here are some tasks to help learners understand the three focal areas in week 18. Kindly take note of differentiated learning when applying these tasks.

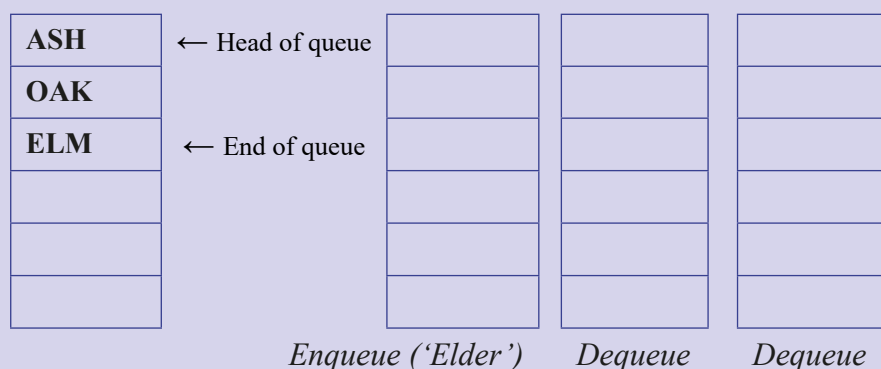
Task 1

List as many real-life examples of a queue as you can (in a given time period).

Then choose one of these examples and explain why it is a queue and not a stack.

Task 2

Work thorough a number of questions in a worksheet similar to the question below. Add a 'Head of queue' pointer and a 'End of queue pointer' to each drawing of the queue after the given operation.



Task 3

Create a PowerPoint slideshow that clearly explains the following:

- What is meant by a queue with examples and possible real-life applications.
- A description of how the *enqueue* and *dequeue* operations work on a queue of integers (using Powerpoint animation tools, if possible)
- At least two computer applications of queues

Present your slideshow to the class.

Task 4

Use the given diagrams and descriptions under the most appropriate heading (1D array, 2D array, singly linked list, doubly linked list, stacks, queues, trees, and graphs).

Some diagrams can be found in assessment question 12. Some descriptions that could be included are:

- A hierarchical structure
- All data elements must be of the same type
- To access an element can be accessed, both a row and a column reference are required
- A FIFO data structure

More than one copy of some descriptions will be required.

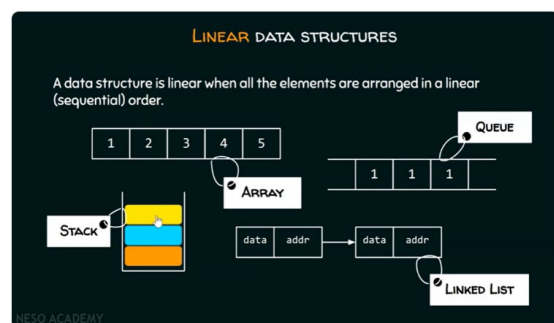
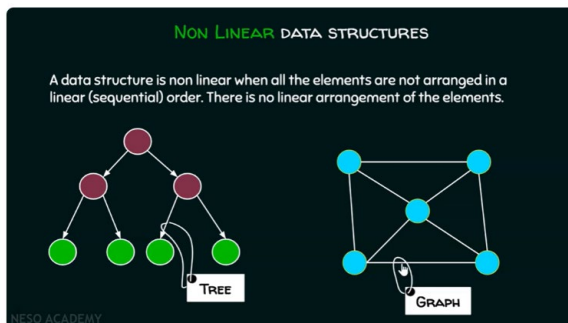
Task 5

Create a concept map on the linear and non-linear data structures that you have studied to date in this course.

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Brainstorm real-life examples of queues. This should lead to a whole class discussion on the features of queues.
2. Using talk for learning, teachers can use a direct instruction approach to work through worked examples of linear queues similar to the example illustrated in Figure 18:3. The animation tools in PowerPoint could be used to illustrate the enqueue and dequeue operations. Diagrams should be used to illustrate binary trees and graphs.
3. Practical demonstrations:
 - The teacher should send a number of documents to the local printer and then show the printer queue on the data projector.
 - Alternatively, if a network printer is available, get a number of learners to send a document to this printer, and then project the network printer queue.
4. Roleplaying: Give a card with a data item (e.g. their names) to an ordered line of learners. These learners should then follow a set of enqueue and dequeue instructions.
5. Video-assisted learning (VAL): There are many videos that could be incorporated into this week's lessons. Some could be used to summarise the data structures covered in this section. One example is the following YouTube video (approx. 4 minutes):



6. The teacher should organise the learners into groups to complete Learner Tasks 1 to 4 at appropriate times, and should guide the learner to reflect on the accuracy and completeness of their responses.
7. Class discussions: The benefits of each data structure and the differences between the data structures can be reinforced through further discussion under the guidance of the teacher.

8. Quiz: Task the groups to create a quiz to test learners on the data structures that you they have studied.
9. Concept map (Task 5): This should be attempted individually before the learners in pairs/small groups collaborate to create a group concept map on data structures.
10. Plenary session: Selected concept maps could be used during a plenary of the content of weeks 15 to 18.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the three focal areas.

DOK Level 1: Recall/Reproduction

1. A queue follows the
 - a. FIFO (First In First Out) principle
 - b. LIFO (Last In First Out) principle
 - c. Random access principle
2. Name two basic operations that can be performed on a queue.
3. What is the name of a data structure that operates like a line where elements are added at one end and removed from the other end?
4. If the elements "A", "B", "C" and "D" are placed in a queue and are removed one at a time, in what order will they be removed?
 - a. ABCD
 - b. DCBA
 - c. DCAB
 - d. ABDC
5. State the name of two non-linear data structures.
6. Name three examples of a linear data structure.
7. Non-linear data structures are more complex and flexible, but they may require more space and time to process and traverse.
 - a. true
 - b. false
8. A queue can be implemented in a programming language using
 - a. an array
 - b. a linked list
 - c. both an array and a linked list

9. Figure 18:6 illustrates a small part of the structure of the Windows file system.

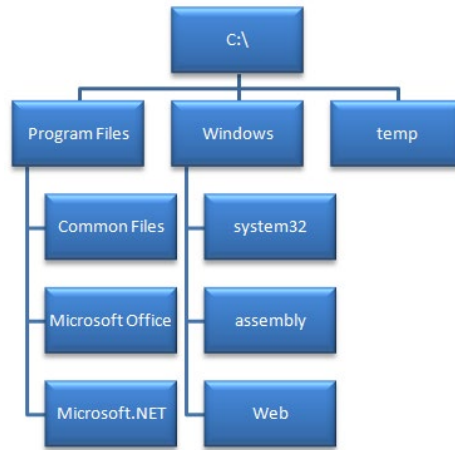


Figure 18.6

This is an example of a

- a. binary tree
 - b. non-binary tree
 - c. graph
10. a. State the name of the data structure shown in Figure 18:7.
- b. Mark ('X') a node belonging to this data structure

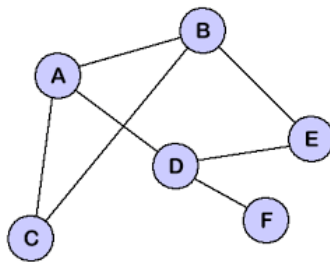


Figure 18.7

11. State one difference between a stack and a queue.
- a. One is linear and one is non-linear
 - b. Stacks are FIFO (first in, first out) data structures. Queues are LIFO (last in, first out) data structures.
 - b. Stacks are LIFO (last in, first out) data structures. Queues are FIFO (first in, first out) data structures.
12. List one data structure that uses contiguous memory allocation and one that uses non-contiguous memory allocation.
13. Complete the sentence: When choosing a data structure, one consideration is how you want to store and a _____ your data.

14. Label the data structures illustrated in Figure 18:8. and tick which structure(s) are non-linear.

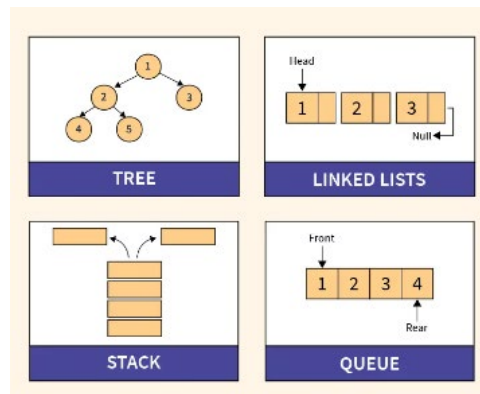


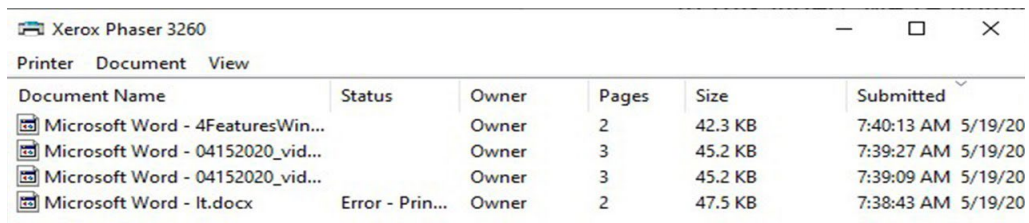
Figure 18.8

DOK Level 2: Skills and Concepts

- Describe the structure of a linear queue and the two fundamental operations that can be made on the queue elements. Use a diagram to support your description.
- Create a slideshow that outlines the main features of each of the following data structures, using a single slide per data structure. Include suitable diagrams and images.
 - 1D array
 - Singly linked list
 - Stack
 - 2D arrays
 - Doubly linked list
 - (Linear) Queue
- What is the main difference in how a stack and a queue access their elements?
- Describe what is meant by a queue using two real-life examples.
- Describe two differences between a queue and a linked list.
- Explain why you would choose to use a queue rather than an array.
- Draw diagrams to illustrate how data items are arranged in the following data structures:
 - Binary tree
 - Graph

DOK Level 3: Strategic Thinking

- Explain why a queue is considered to be a valuable tool for tasks that require sequential processing.
- Identify what application is illustrated in Figure 18:9 and explain why a queue would be a suitable data structure for this application.



Document Name	Status	Owner	Pages	Size	Submitted
Microsoft Word - 4FeaturesWin...		Owner	2	42.3 KB	7:40:13 AM 5/19/20
Microsoft Word - 04152020_vid...		Owner	3	45.2 KB	7:39:27 AM 5/19/20
Microsoft Word - 04152020_vid...		Owner	3	45.2 KB	7:39:09 AM 5/19/20
Microsoft Word - It.docx	Error - Prin...	Owner	2	47.5 KB	7:38:43 AM 5/19/20

Figure 18.9

3. Describe at least three different criteria that could be used to compare data structures.
4. Recommend a data structure for each of the following applications. In each instance give at least one reason for your choice of structure:
 - CPU task scheduling,
 - Reversing a list of stored characters
 - Storing a song playlist to play previous and next song
5. Explain why a programmer would choose a contiguous memory data structure to implement a queue rather than a non- contiguous memory structure.

DOK Level 4: Extended Thinking

1. Investigate what is meant by a circular queue. Use the application of a computer-controlled traffic light system to explain how a circular queue works.
2. Find out more about tree and graph data structures. Create a presentation about these two non-linear data structures.
3. Research what is meant by the Big O notation, and describe how it links to data structures and algorithms.
4. Find out what is meant by a priority queue and write a short report on your findings.
5. Expand your knowledge and understanding of non-linear data structures by researching the following:
 - Heaps
 - Hash tables

WEEKS 19 and 20

Learning Indicators:

1. *Implement Algorithms into programs with the use of flowcharts, pseudocode and programming languages such as Python, etc.*
2. *Explain the steps in planning a program putting actions in the right order*

Theme or Focal Areas

1. **Programming basics using Python**
2. **How to translate a simple algorithm using single variables into actual code using a programming language**

Teachers should dedicate time to read and familiarise themselves with the programming language they wish to use for this course. These notes use and recommend the Python programming language. It is an easy-to-learn language, with a simple syntax, popular in schools but also used in industry. It is a text-based programming language which is specified in Year 2 of the Curriculum.

Key Concept Notes

Programming Basics Using Python

An introduction to Python

There are many programming languages, including Python, Java, C#, and Swift. Python is a popular and versatile high-level programming language known for its readability and ease of use. Python is considered readable because it is high-level so syntax resembles the English language, making it easier to learn. It is versatile because it can be used for a variety of purposes.

What can Python do?

Python is used for software development, web development (server-side), mathematics, and lots more. It is an incredibly powerful programming language – there's a reason why companies like Google, Dropbox, Spotify, and Netflix use it. For example, the Dropbox desktop client is written entirely in Python, which speaks to its cross-platform compatibility.

Steps for getting started with Python

After the introduction, it is common to demonstrate and have learners experiment with the Python console (also known as the Python shell) to try out some basic Python code. Explain that the Python shell is a good place to experiment with small code snippets. It is then time to introduce the learners to writing Python programs:

1. **Select an Integrated Development Environment (IDE) for writing and running your Python code**
 - An IDE (integrated development environment) is software that combines all the functions needed for program development in one place. Without an IDE, developers would need to use both a text editor to enter code and a separate program called a compiler to make the program understandable to the computer.
 - Popular choices of IDE's for Python include IDLE, Spyder, PyCharm, Visual Studio Code, and Jupyter Notebook.
 - At SHS Year 1 level, the basic Python IDE called IDLE will suffice. Download the latest version of IDLE that is compatible with your computer's operating system at <https://www.python.org/downloads/>

- To create and test Python programs on an iPad, there seems to be a good number of Python editor apps available from the App Store, including Python Editor (free) - see, Juno (free), Pythonista, and Pyto.

If using an iPad or smartphone, it is strongly recommended that learners get a keyboard, and possibly also connect a Bluetooth mouse.



Figure 19.1

- Replit (www.replit.com) is a free online service that gives you a way to develop and run Python programs inside a browser.

2. Set up a local or cloud folder

Teachers will need to instruct learners how and where to set up a folder to store their Python programs. When saved, Python files will have the **.py** extension.

3. Start coding!

- print () function is an in-built function in Python.

To print a blank line:

`print ()` or `print("\n")`

A first Python command

```
>>> print('Hello, world!')
Hello,
world!
```

See Figure 19:2 on how to label output in Python.

```
message = "Python is fun"
print(message)
message = "Python is fun"
print("My message to you is ", message)
player = "Michael Jordan"
print(player, " earned the nickname 'Air Jordan'.")
```

```
Python is fun
My message to you is
Python is fun
Michael Jordan
earned the nickname
'Air Jordan'.
```

Figure 19.2

- Assignment operator

The assignment operator in Python is the “=” symbol. For example,

age = 15

reply = False

cost = 34.56

city = “Kumasi”

console = ‘PlayStation’

Note that single-quoted strings and double-quoted strings should both work in most Python editors.

- Arithmetic operators

Maths	Python
=	==
≠	!=
<	<
>	>
≤	<=
≥	>=

Double equals sign

The equality comparison is defined in Python with a *double* equals sign, “==”. The sign is doubled to distinguish comparison from assignment.

- input ()_function allows user input.

Study the code in Figure 19:3 to understand how this function works. This code was written in Replit (at www.replit.com).

```
print("Enter your name:")
fname = input()
print("Hello, " + f name) # use Of +
print("Have a good day, ", fname) #use of comma
print("\n"*3) #prntns 3 blank lines
sname = input("Enter your surname: ")
print("That's a nice name" )
```

```
Enter your name:
Mark
Hello, Mark
Have a good day, Mark
```

```
Enter your surname:
That's a nice name
Zuckerberg
```

Figure 19.3

The above code exemplars should enable you to now code simple sequence programs, that is, programs that consist of simple input, process, and output steps. The remainder of these notes for weeks 19 and 20 will focus on the implementation of algorithms involving simple sequence using the Python programming language.

A computer program using the input-process-output model receives inputs from a user or other source, does some computations on the inputs, and returns the results of the computations. Before the program is run, it needs to be written, and before it is written, it needs to be designed. Part of the design process, as we have seen already, is to create an algorithm to solve the given problem. Outputs

are the goal of the problem solution. Inputs are the information you get to use to solve the problem. Processes are the steps need to get from the input information to the output results.

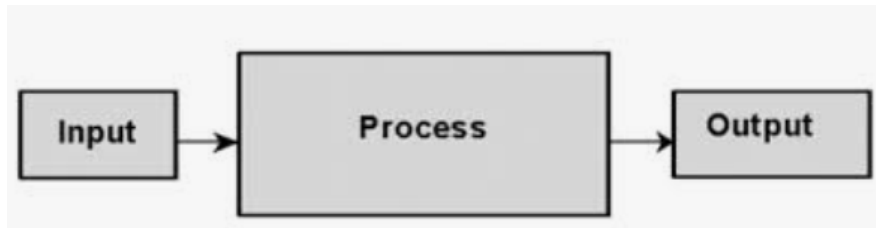


Figure 19.3

Algorithms Implementation

Algorithm implementation refers to the process of translating an algorithm's logical steps and instructions into actual code that can be executed by a computer. This process involves writing code in a programming language to achieve the desired functionality described by the algorithm.

Here is a step-by-step guide to implementing an algorithm in Python.

1. **Understanding the algorithm:** Ensure you have a clear understanding of the algorithm you are implementing. Note that this design may be modular or non-modular. For learners new to coding, the general advice is to first begin with non-modular programs. Exemplars of each are given in this manual.

Review the algorithm's description, inputs, processes, outputs, including any required data structures.

A dry run of the algorithm is advised. A dry run involves a programmer working through the algorithm manually using pen and paper to trace the value of variables to see that they are used and updated as expected.

2. **Translating the algorithm steps to code:**

- Break down the algorithm's steps into smaller segments, if required.
- Write Python code that corresponds to each step, using appropriate language constructs. Programming language constructs include simple sequence, selection, loops, and functions. If the algorithm requires data structures such as arrays and stacks, implement them in your code.

Note that in a three-week slot, it would be possible for learners new to programming to get up to speed with all the main constructs. This manual will only focus on simple sequence programs (i.e. input, process, output programs) using discrete variables and variable arrays. There are practical programming elements in the Curriculum for Year 2 and Year 3 where other important constructs such as selection and loops/iteration can be covered.

- Handling inputs and outputs: Implement code to handle inputs required by the algorithm. You can use the assignment operator (`=`), the `input()` function for inputs, and the `print()` for display outputs. Data can also be input and output from an external file in Python using special file-handling functions.
3. **Creating a new Python file:** Open your chosen IDE and create a new Python file. Enter your code. Make your program more readable by using meaningful variable names, adding whitespace, and adding some comments to your code (using the `#` symbol). Save your program with a suitable filename in the folder for your Python files.
 4. **Testing and debugging:** Test your code with various test data/cases to ensure it is producing the correct results. Advice on designing comprehensive test data sets was covered in week 14. The IDE will have in-built debugging tools to help you identify and fix any errors.

5. **Optimisation and refinement:** Review your code for opportunities to optimise performance, improve readability, and remove unnecessary code.
6. **Code review and feedback:** If applicable, share your code with peers or mentors for code review. Feedback can help you enhance your implementation.
7. **Integration (if applicable):** If your algorithm is part of a larger application or project, integrate your Python code as needed.
8. **Deployment (if applicable):** If your implementation is part of a software product, ensure it is correctly deployed and functions as intended in a production environment.

Implementation Exemplars

Example 1 – a simple maths algorithm

We have already seen this algorithm, both in pseudocode and as a flowchart in week 14.

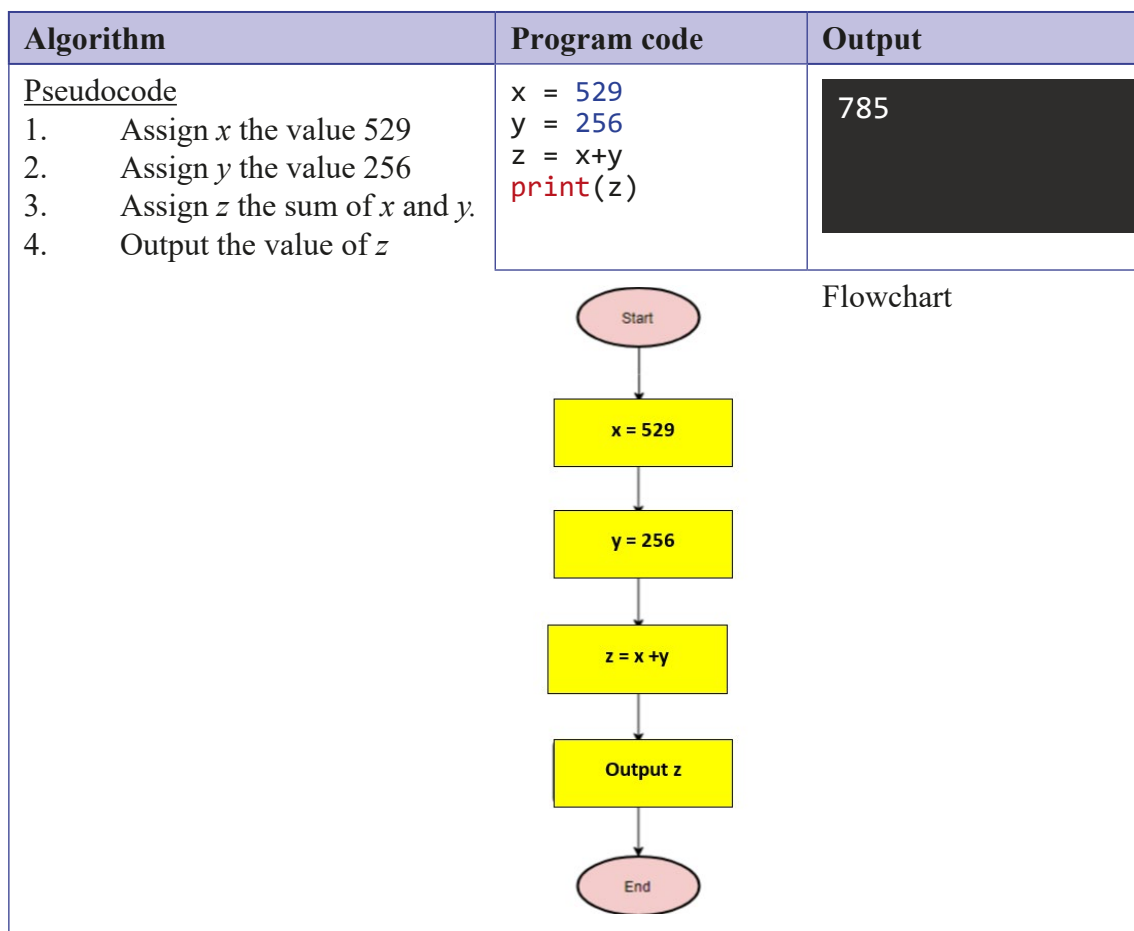


Figure 19.5: Algorithm in two formats with program and output

Example 2 – a swap algorithm

1. Introduce the Concept of Swapping

The swap algorithm is a simple and standard algorithm used to exchange the values of two variables. It allows us to interchange the contents of two variables, effectively swapping their values. The swap algorithm is commonly employed in programming when there is a need to reassign values between two variables without losing their original data.

The general idea of the swap algorithm can be explained in simple steps:

- Store the value of the first variable in a temporary variable.
- Assign the value of the second variable to the first variable.

- Assign the value stored in the temporary variable to the second variable.

By following these steps, the values of the two variables are effectively exchanged or swapped. The temporary variable serves as a placeholder during the swapping process to prevent the loss of data.

2. *Manual Swapping*

Begin with a simple manual swapping exercise. Ask the learners to imagine they have two variables, A and B, and write down the values of these variables on a piece of paper.

Guide them through the steps of swapping the values of A and B without using any programming. Encourage them to think step by step and explain the process, including why a temporary variable is needed.

3. *Pseudocode for Swap Algorithm*

Remember, pseudocode is a way to represent an algorithm in a language-independent manner.

A suggested first step would be to write an algorithm for a non-modular program that swaps two variables. Figure 19:6 gives an algorithm that swaps the values of two variables, A and B.

Algorithm Swap – Version 1

1. A = 49
2. B = 61
3. Output A, B and label as original values
4. Temp = A
5. A = B
6. B = Temp
7. Output A, B and label as values after swap

Coding (non-modular):

```
#Swap program (non-modular), version 1
a = 49
b = 61
print ("The original values of a and b are", a, " and ", b)
#swap the values in a and b
temp = a
b = a
a = temp
print ("The new values Of a and b are", a, " and ", b)
```

Output:

```
The original values of a and b are 49 and 61
The new values of a and b are 61 and 49
```

Figure 19.6

Note that the variable names used in the code can differ from those in the algorithm. In this example, the names are the same but the case is different. The convention in Python is to use lowercase for variables.

Figure 19:7 shows a slightly different version of this algorithm where the user enters the two values to be swapped. The corresponding code and the output from a test run are also shown.

Algorithm Swap - Version 2

1. Enter first value
2. Enter second value
3. Output these two values and label as original
4. Let temp = first value
5. Let first value = second value
6. Let second value = temp
7. Output first value and second value, and label as values after swap

Coding (non-modular):

```
#Swap program (non-modular), version 1
a = input("Enter a value for a ")
b = input("Enter a value for b ")
print ("The original values of a and b were", a, " and ", b)
#swap the values in a and b
temp = a
a = b
b = temp
print ("The new values Of a and b are", a, " and ", b)
```

Output:

```
Enter a value for a 20
Enter a value for b 68
The original values of a and b were 20  and  68
The new values of a and b are 68  and  20
```

Figure 19.7

If the learners are familiar with subroutines in Python the code for a swap function could be demonstrated or they could experiment with it, as is shown in Figure 19:8, or similar. The corresponding code and output when the program are executed are also shown. In this Python example, we define a swap function that passes two variables, *A* and *B*, and returns the swapped values. The three steps of the swap algorithm are implemented within the function. After calling the swap function with *a* and *b* as arguments, the values of *a* and *b* are swapped, as demonstrated in the output.

Algorithm for swap function:

Swap (A, B)

1. Temp = A
2. A = B
3. B = Temp

Coding (Modular):

```
def swap(a, b):
    temp = a
    a = b
    b = temp
    return a, b

#Main program
firstNumber = 49
secondNumber = 61
print ("The original values of firstNumber and secondNumber are",
firstNumber, " and " secondNumber)

firstNumber, secondNumber = swap(firstNumber, secondNumber)
print ("The new values of firstNumber and secondNumber are", firstNumber,
" and ", secondNumber)
```

Output:

```
The original values of firstNumber and secondNumber are 49 and 61
The new values of firstNumber and secondNumber are 61 and 49
```

Figure 19.8

The swap algorithm is a fundamental concept in programming and is widely used in various applications, such as sorting algorithms, data manipulation, and variable-swapping tasks.

Example 3 – using if statement

This is an example from the BBC Bitesize website for learners who have some previous programming experience and are familiar with the selection construct (if/else statement)

Problem: A cinema is offering discount tickets to anyone who is under 15. Find out if someone is eligible for a discount ticket.

Decomposing this problem, gives:

- Find out how old the person is
- If the person is younger than 15 then say, “You are eligible for a discount ticket.”
- Otherwise, say “You are not eligible for a discount ticket.”

To convert the flowchart or pseudocode into a program, look at each individual step, and write an equivalent instruction. Sometimes the steps will not match exactly, but they will be fairly close.

Algorithm (written in pseudocode)	Algorithm (in a flowchart)
OUTPUT “How old are you?” INPUT User inputs their age STORE the user’s input in the age variable IF age < 15 THEN OUTPUT “You are eligible for a discount.” ELSE OUTPUT “You are not eligible for a discount.”	<pre> graph TD Start([Start]) --> Input[/Input 'How old are you?'/] Input --> Decision{Is age < 15?} Decision -- YES --> Output1[/Output 'You are eligible for a discount'/] Decision -- NO --> Output2[/Output 'You are not eligible for a discount'/] Output1 --> Stop([Stop]) Output2 --> Stop </pre>
Coding: <pre> age = int(input("How old are you?")) if age < 15: # checking if correct age for a discount ticket print("You are eligible for a discount.") else: print("You are not eligible for a discount.") </pre> Run 1 output: <pre> How old are you?17 You are not eligible for a discount. </pre> Run 2 output: <pre> How old are you?17 You are not eligible for a discount. </pre>	

Note that lines 1, 2, and 3 of pseudocode have been combined into one input() statement.

Learning Tasks

Here are some tasks to help learners understand the three focal areas in weeks 19 and 20. Kindly take note of differentiated learning when applying these tasks.

Task 1

Convert a set of single lines of pseudocode to code. An example is:

- Output ‘Happy Birthday’ three times, separating each Happy Birthday message with a blank line.

Task 2

Convert a single flowchart step to code using a programming language that you are familiar with. An example is:

```
Set rate to 11.75
```

Task 3

Use a set of given algorithms, complete the partially-completed corresponding programs. An example using Python code is:

Pseudocode to calculate the area of a triangle

1. Enter length of base in cms
2. Enter perpendicular height in cms
3. Area of triangle = (length of base x perpendicular height)/2
4. Output area of triangle in square cms

Code to complete

```
#Program to calculate and output the area of a triangle
1 ("Area of triangle")
print()

#Get dimensions of triangle in cms
base = 2 ("Enter your length of the base in cms:")
height 3 ("Enter your length of the base in cms:")
=
#Calculate area
area = 4

5 Output area in square cms
print("The area of the triangle in square cms is" 6)
```

Task 4

Correct the errors in the following lines of code. These errors could be syntax or logical. Two examples are:

- size = input("Enter your shoe size)
- average = (number1 + number2)/3

Task 5

Correct the errors in the following Python programs. One example (with 3 syntax errors) is:

```
print "Enter your favourite colour: "
colour = inputt()
print("Hello there")
print( colour "is a nice colour")
```

Enter your corrected code in Python to check it works.

Task 6

For each example, rearrange the lines of code to match the given pseudocode. One example (with comment lines included) is given below. Note that the operator for exponentiation in Python is **, so print(10 ** 3) would output 1000.

Pseudocode

1. Let x equal 2

2. Let p equal 5
3. Calculate the value of x^p
4. Output the value of x^p

Code

Line of Code	Correct order
p = 5	
#Output answer	
print ("The answer is ", answer)	
#get values and do calculation	
x = 2	
answer = x**p	

Enter the re-arranged code and run to check that it gives the expected answer.

Task 7

For each example, rearrange the lines of pseudocode to match the given code. One example is:

Code	Lines of pseudocode	Order
x = 2 p = 5	Let p equal 5	
answer = x*p	Output the value of xp	
print("The answer is ", answer)	Calculate the value of xp	
	Let x equal 2	

Task 8

Add comments to the following sets of programs to show your understanding. An example of a simple program to comment is:

```
# Define the value of pi
pi = 

# Define the radius of the sphere
r = 

# Calculate the volume of the sphere using the formula
v = 

# Print the calculated volume of the sphere
print  2
```

Task 9

Design and code a program that will swap two string variables.

Task 10

- Study the given problem specification below and the corresponding algorithms.
- Do a dry run of the algorithm to calculate the expected output (using your chosen user inputs, where applicable)
- Implement your algorithm using a programming language that you are familiar with.

Break down the steps further where required before implementation. Add comments to explain your code.

- Test your program, make any corrections and improvements before saving the final version.

Problem Specification

Find the total price a pair of trousers and a jacket in a sale. The original prices of the two items should be entered by the user. 10% discount should be applied on the trousers and 5% on the jacket. A detailed receipt should be output together including the total cost.

Algorithm

1. Enter cost of the pair of trousers in GHC
2. Enter cost of the jacket in GHC
3. Calculate discount on trousers @10%
4. Calculate discount on jacket@5%
5. Calculate total cost
6. Output receipt showing names of items, original costs, discount applied, final costs, and total cost.

Task 11

- Write an algorithm for each of the following sets problem specifications.
- Use pseudocode to illustrate your solution.
- Plan what data you will use to test your program. Do a dry run of your algorithm and record the outcomes.
- Implement the algorithm using your chosen programming language. Add suitable commentary.
- Test your program with your test data,

A possible problem specification is:

A program is required to calculate the total floor area from a floor plan given in Figure 19:9. The dimensions of both rooms should be entered by the user.

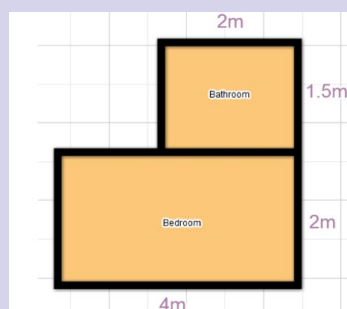


Figure 19:9

Task 12

Same steps as Task 10 with a more complicated floor plan such as that shown in Figure 19:10.



Figure 19:10

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher. Teachers should have established their chosen IDE to use in lessons and be well versed in their chosen high-level language in advance of these lessons.. There are many excellent textbooks and online resources on learning to program in Python. A useful resource to share and use with learners new to Python is <https://www.w3schools.com/python/>.

1. Teachers can begin the lesson with a brief refresher session on what is meant by the SDLC and explain to the learner that the main focus of the lessons over the two weeks will be on the implementation stage/phase of this cycle.

```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 03:13:28)
[Clong 6.0 (clong-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> x = 5
>>> print(x)
5
>>>
```

Figure 19.11

Demonstrate Python basics. A recommended sequence of instruction steps is:

- What is Python and what is it used for?
- How to launch Python IDE
- Python console/shell for testing line/lines of code - see Figure 19:11
- print() function
- Recap of variables in Python and naming convention, and data types
- Assignment statement and type conversion
- Creating a new Python program, entering code, saving, and running
- Opening an existing Python program
- Arithmetic operators
- input() function
- Adding comment lines

Learner should have a sound understanding of these individual basic steps before proceeding to focal area 2 (How to translate a simple algorithm using single variables into actual code using a programming language).

2. Recap the types of programming errors (syntax, run-time, and logic) with examples.
3. Recap the two design notations introduced previously in this section: pseudocode and flowcharts. Slides/visuals from week 13 could be used here.
4. Using simple sequence examples only (input-process-output), demonstrate translating from an algorithm to code following a similar order of activities to the Learner Tasks. Have a bank of examples for each activity.
 - a) Translate one line of pseudocode to code
 - b) Translate one flowchart symbol to code
 - c) Use a given algorithm to fill the gaps in partially completed code
 - d) Convert from an algorithm to code and vice versa
 - e) Add comments to a given program to explain the code
 - f) Identify suitable data to test a given program
 - g) Do a dry run of a given algorithm
 - h) Do a dry run using a program listing
 - i) Write a possible algorithm in pseudocode using a given problem specification, convert it to code, and then test your program using a pre-planned set of test data.

Encourage good practice when writing pseudocode, such as numbering the steps, and breaking down the steps as much as possible, thinking about the test data at design time and doing a dry run of the algorithm before implementation.

Similarly for implementing the algorithm, encourage good practice such as using meaningful variable names, adding commentary, and adding whitespace to make the code more readable, as well as labelling output.

5. Use manual demonstrations and real-life references, wherever possible. An example is given in the notes for using three pieces of card and paper for a manual swap of two variables before using the computer.
6. Learners should work in groups of no larger than three when completing the Learner Tasks. Teachers should ensure that all group members are actively participating and contributing. Also ensure that learners complete some of examples from each task individually. This will allow the teacher to check who is managing to convert from an algorithm to and from code successfully. Repetition of similar tasks can help understanding where a learner is experiencing difficulties.
7. Emphasise the importance of internal commentary. Studying the comments they have added (such as in Task 8) can be an easy way for teachers to gauge a learner's understanding of given code. Of course, verbal commentary on the code should also be acceptable.

Advice on adding commentary:

- It is better to have no comment at all than one that is difficult to understand or inaccurate. Comments should be updated regularly to reflect changes in your code.
- Too many comments can be distracting and reduce code quality.
- Comments are not needed where the code's purpose is obvious. (This advice may have been ignored in some of the simple program examples).

A common approach used by teachers with learners who are new to programming or who have difficulty with translating from algorithm to code is to:

- Start a new program in the editor
 - Copy and paste the algorithm into the text editor
 - Comment out each line of the algorithm, for example,


```
#1. rate = 17.50
#2. noHours = 40
#3 calculate pay = rate x noHours
....
```
 - Enter the appropriate code *under* each comment
 - Remove the numbers from the comments
 - Delete any unnecessary comments. For example, the first two comments above would be deleted as the code should be self-explanatory. The third comment could be amended to `#calculate pay`
8. Learners should have the freedom to amend their program design and program code as they go. Going from algorithm to code to testing is an iterative process. For example, often the learners will need to amend their algorithm once they have started implementation, or as a result of an error highlighted during testing.
 9. Strips of card with written lines of code and pseudocode could be used for Tasks 6 and 7 respectively rather than a table.
 10. Learners need lots of practice at writing algorithms and code, and converting from one to the other. Teachers should have a large bank of simple sequence tasks, including some more complex tasks such as Learner Task 12.
 11. Give learners opportunities to verbally explain their algorithm or code to the teacher, other learners, or the whole class. Encourage discussion on how the code could be more efficient, if applicable.
 12. Learners will be introduced to other high-level programming constructs and modular programming in Years 2 and 3. However, they should not be deterred from exploring these constructs themselves if they demonstrate a strong aptitude for programming or have some prior programming skills. Teachers, therefore, should have at hand extensions to the above tasks or more advanced problem specifications (such as Example program 3).

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

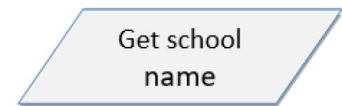
The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the two focal areas.

Learners should state which programming language they are using when answering coding questions.

DOK Level 1: Recall/Reproduction

1. What does the statement 'print' do in a Python program?
 - a. Compares what the user has input to a stored value
 - b. Outputs the specified message on the screen
 - c. Prints a hard copy of the pseudocode to a printer

2. At what stage of the program development cycle is the algorithm translated into code?
3. Complete the sentence:
A program is an i_____ of code to instruct a computer on how to execute an algorithm.
4. a. Complete the following line of code to calculate the area of a square where *side* is the length of one side:
area = side _____
b. State a possible data type for the variable *side*.
5. State the name of three programming languages that could be used to implement an algorithm.
6. Translate the following lines of pseudocode to code using a programming language you are familiar with.
 - a. Set *amount* to 48
 - b. Make *answer* equal the square of the value stored in *number*
 - c. Output the value of *final_price*
 - d. Output the value of *final_price* with a suitable label
 - e. Let *triple* equal the value stored in the variable called *cost* times three
7. Implement the given step from a flowchart using a programming language that you are familiar with.
8. The following lines of Python code generate an error(s) when they are run. Re-write the incorrect lines to remove the errors. Test your corrected code in the Python shell.
- 9.



	Incorrect code	Corrected code
1.	<code>print(Happy Birthday to you)</code>	
2.	<code>age := 13</code> <code>pront(age)</code>	
3.	<code>x = 9</code> <code>y = 7</code> <code>x + y = sum</code>	

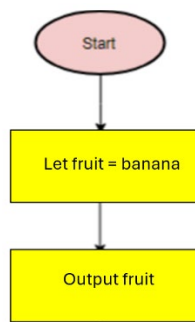


Figure 19.12

Implement into code the part of a flowchart shown in Figure 19:12.

10 `answer = True`

The line of code shown above is

- an equal statement
 - an assignment statement
 - an input statement
11. Once the program code has been written, the algorithm cannot be changed.
- true
 - false
12. Convert the following lines from an algorithm into a programming language you are familiar with:
- Output the sum of 5 and 8
 - Assign a number between 1 and 10 to a variable called *noOrdered* and output the two times the value of this variable.
 - Assign two integers input by the user to variables called *no_1* and *no_2*, and output the result of *no_1* divided by *no_2*

DOK Level 2: Skills and Concepts

- Describe how an algorithm for a problem specification and a program can be linked.
- Joseph has been asked to design a program to calculate the potential profit in a soft drink business. The program will store the costs involved in producing and selling one litre of each drink.

The following calculations will be used to output the profit made for each litre of drink:

manufacturing cost = water cost + flavouring cost + labour cost

profit = selling price – manufacturing cost

- State the number of variables Joseph would require in his program.
 - State a possible name and data type for each of these variables.
 - Using these variable names, write the code for the two calculations.
3. Draw a labelled diagram to show how the swap algorithm works.
4. The following algorithm calculates the average of two numbers
- Enter first value
 - Enter second value
 - Let $\text{sum} = \text{first value} + \text{second value}$
 - Let $\text{average} = \text{sum} / 2$
 - Output the average value with a suitable label.
- Implement the above algorithm in a language that you are familiar with.
- Add commentary and test your program.
- Adapt the above algorithm so that it finds the average of three numbers entered by the user.
 - Write code to match the algorithm produced for part b. Enter this code into your program editor. Add commentary and test.
5. Study the Python program below and answer the questions that follow.

```
celcius = float(input("Enter temperature in celcius: ")) #int also possible here
fahrenheit = (celcius * 9/5 + 32
```

```
fahrenheit = round(fahrenheit, 2) #this could be included in previous line
print (str(celcius) + " is equal to " + str(fahrenheit) + " in fahrenheit")
```

- Describe in your own words what you believe each line of this program does.
- What will the output be from the above program when the following values are input for temperature in Celsius? Write down the results of your calculations.
 - 100
 - 0
 - 23
 - 31.5
- Write a corresponding algorithm in pseudocode for this program.
- Describe a graphical notation that could be used to represent this algorithm and give one advantage that it has over pseudocode.
- Copy this code into your program editor and use it to check your answers for part b.

DOK Level 3: Strategic Thinking

- Write an algorithm that will swap three variables called number1, number2, and number3 with the variable ahead in a circular way. This means number2 would take number1's value, number3 would take number2's value, and number1 would take number3's value.
- Design a program to convert a temperature input by the user from °F to °C and implement this design. Add commentary to your program and amend your algorithm to ensure that it matches your final program. Use the formula:

$^{\circ}\text{F} = (^{\circ}\text{C} \times 9 / 5) + 32$
- Design and code a foreign exchange calculator program for GHC to US \$. The user should enter the amount to be converted and the exchange rate. The result of the conversion should be output.
- Design and code a program that will calculate the floor area shown in Figure 19:13. The dimensions of each room should be assigned within the program.

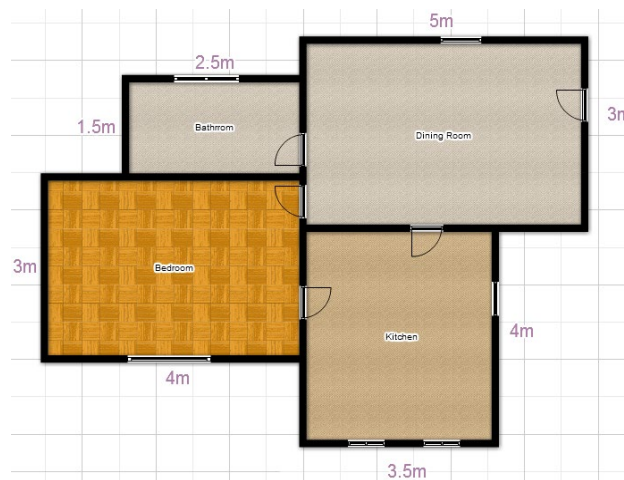


Figure 19.13

- Design and code a program that will calculate the area of the shape below as efficiently as possible. The user should input the necessary dimensions from the given diagram. The area in *square cms* should be output.

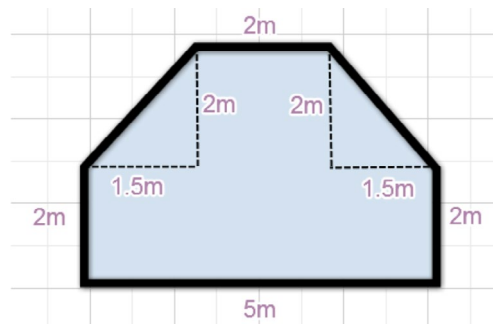


Figure 19.14

DOK Level 4: Extended Thinking (going beyond simple sequence and non-modular programs)

1. Use <https://www.w3schools.com/python/> to find out more about the following constructs in Python:
 - Selection
 - Iteration/loops (unconditional and conditional)

Create some programs using these constructs.

2. Design and code a Python program which will accept as input three valid 5-letter words (any case). The program should output if each of the words is a palindrome or not and the total number of words entered that are palindromic. For example, part of the output when *radar*, *madam* and *hello* are input would be 'The number of palindromes entered was 2'.

The word *computing* should be rejected as it does not have exactly 5 letters. You will need the `len()` function for this tasks.

3. Using efficient code, design and code a program that will do the following:
 - Input and store 6 values between 1 and 10 inclusive
 - Each number input must be validated
 - Process and output both the sum and average of the 6 numbers
 - The average should be output correct to 2dp

Test your program as fully as possible, including the input validation code.

4. Investigate how Python is used in data science and analytics. Create a PowerPoint slide summarising your findings.
5. In cryptography, a Caesar Cipher is one of the simplest and most widely known encryption techniques. In this cipher, each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. The method is named after Julius Caesar, who used it in his private correspondence.

Your task is to create a modular Python program which will accept the input of a word (any case) and a shift key (assume a right shift) for the Caesar Cipher. The encrypted message should be output. Save as *caesar_1.py*.

Extensions:

caesar_2.py: Encrypt a valid four-word message. Spaces should not be encrypted.

caesar_3.py: This version should allow the user to select from a right or left shift key

With a (right) shift key of 3, A would be replaced by D, B would become E, and so on. Z would be replaced by C - see Figure 19:15. 'cAb' when encrypted would be 'fDe'.

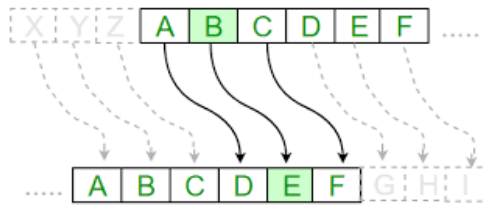


Figure 19.15

6. Create two versions of a program that meets the specification below. One of these programs (*decorator_1.py*) should be non-modular. The other (*decorator_2.py*) should be modular. An algorithm for each version should be written in pseudocode.

A data flow diagram (often drawn to help analyse a problem) is included and should help get you started on the program design.

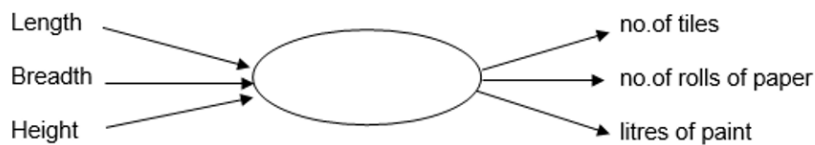
Design, write and test a program to

- prompt the user to enter the length, breadth and height of a rectangular room
- calculate the floor area and wall area
- calculate
 - a. the number of carpet tiles for the floor
 - b. the number of rolls of wallpaper for the walls
 - c. the number of litres of paint for the ceiling
- display the results of these calculations

Notes:

- a. carpet tiles are 1m square
- b. 1 roll of wallpaper covers approximately 5 m²
- c. A litre of paint covers approximately 4 m², and two coats will be needed

Data Flow Diagram



7. Research one other popular high-level programming languages other than Python. Create a table in Word to summarise a comparison between this language and Python using suitable criteria.

WEEK 21

Learning Indicators:

1. *Implement Algorithms into programs with the use of flowcharts, pseudocode and programming languages such as Python, etc.*
2. *Explain the steps in planning a program putting actions in the right order*

Theme or Focal Areas

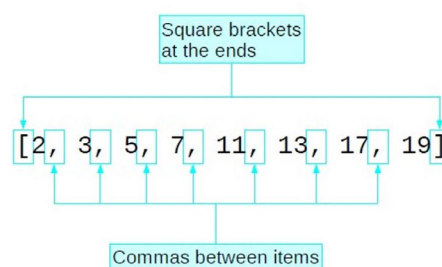
1. **Implementing and manipulating 1D arrays in Python**
2. **Built-in (array) list methods in python**

Arrays In Python

As discussed earlier in this manual, one of the most fundamental data structures in any language is the array which is an ordered collection of items of a single type. Python does not have a native array data structure, so the list data structure is used in Python to implement arrays.

A list is simply a sequence of values stored in a specific order with each value identified by its position in that order.

How Python presents lists



Python counts from **zero**

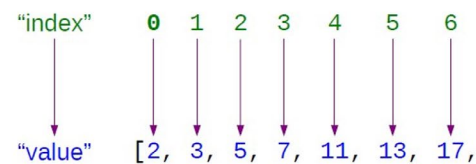


Figure 21.1

Python lists can contain duplicate values.

This manual from now on will generally use the term ‘array’ in the examples instead of list.

We have looked at how to reference a particular element in an array earlier in the manual – we use the index of the element.

For example, if the array in Figure 21:1 was called *numbers*, then the code `print(numbers[1])` would output 5.

To be able to code actions to fully manipulate arrays in a Python program, an understanding of iteration and selection constructs will be required. These will be covered in Year 2 and 3 of the course. To allow for implementation of algorithms that use arrays at this stage, we will use in-built code called functions and methods. This will make the algorithms very straightforward and most of the coding will be done for you.

Array length

To determine how many items an array has, use the **len() function** – see Figure 21:2.

```
fruitList = ["apple", "banana", "cherry", "apple"]
print(len(fruitList))
```

4

Figure 21.2: A program using `len()` and output

A brief description of some common list methods in Python

1. `clear()`: Removes all elements from the list.
2. `copy()`: Creates a copy of the list.
3. `count()`: Returns the number of occurrences of a specified value in the list.
4. `index()`: Returns the index of the first occurrence of a specified value.
5. `reverse()`: Reverses the order of elements in the list.
6. `sort()`: Sorts the list in ascending order (or based on a custom sorting key).

Examples of how the Sort method works is shown in Figure 21:3

```
>>> numbers = [4, 7, 5, 1]

>>> numbers.sort()
<----- The function does not
          return the sorted list.

>>> print(numbers)
[1, 4, 5, 7] <----- It sorts the
                  list itself.

Numerical order.
```

```
>>> greek = ['alpha', 'beta', 'gamma', 'delta']

>>> greek.sort()

>>> print(greek)
['alpha', 'beta', 'delta', 'gamma']

Alphabetical order
of the words.
```

Figure 21.3: Checking out how `Sort ()` method works in the Python console/shell

Algorithms Implementation (Continue From Weeks 19 And 20)

Example 4 – reversing an array

Problem: Design and write a program that set up an array of three daily activities of your choice (you choose the elements, no duplicates). Use in-built Python code to reverse the order of the elements in the array. Each element in the reversed array should be output on a separate line.

Algorithm

1. Set up an array with the names of 5 daily activities.
2. Use an in-built method in the programming language to reverse the order of the elements in the array.

3. Output each element in the re-ordered array, one element per line.

Coding and Output

```
#set up array of 5 daily activities
activityList = ["sleeping", "eating",
               "talking", "playing", "studying" ]

#Reverse the items in the array
activityList.reverse()

#print out the array in reverse order
print("The activities in reverse order")
print(activityList[0])
print(activityList[1])
print(activityList[2])
print(activityList[3])
print(activityList[4])
```

```
The activities in reverse order
studying
playing
talking
eating
sleeping
```

Figure 21.4

Encourage the learners to think of a more efficient algorithm to give the same output.

Example 5 – counting occurrences in an array

Problem: Design and write a program that set up an array of the names of seven types of computers (duplicates allowed). Use in-built Python code to find how many of two types of computer in the array.

Use the following elements in this order: tablet, desktop, tablet, smartphone, supercomputer, tablet

Algorithm

1. Set up an array with the given elements
2. Set up two types of computers to check for
3. Use an in-built method in the programming language to find the number of occurrences of the two items in the array
4. Output the number of occurrences of each item to be checked in the array

Coding and Output

```
computerList=["tablet", "desktop", "tablet",
             "smartphone", "supercomputer", "tablet"]

#enter items to check occurrences of
toCheck1 = "tablet"
toCheck2 = "mainframe"

print("Number of " + toCheck1)
print(computerList.count(toCheck1))
print("Number of " + toCheck2)
print(computerList.count(toCheck2))
```

```
Number of tablet
3
Number of mainframe
0
```

Figure 21.5

Example 6 – copying an array

Introduce the concept of array copying: start by explaining the concept of array copying. Emphasise that copying an array means creating and copying the elements from the original array to a new one.

Problem: Copy the contents of an array to another array with a different name.

Algorithm

1. Assign the names of five types of snakes to an array called *snakesList*
2. Use an in-built method in the programming language to make a copy of this array with the name *copySnakeList*.
3. Output the original array and the copied array.

Code and Output

```
#set up array of 5 types of snakes
snakesList = ["python", "cobra",
             "mamba", "viper", "adder"]

#Make a copy of this array
copySnakeList = snakesList.copy()

#print out both arrays
print ("SnakesList")
print (snakesList)
print ("Copy of SnakesList")
print (copySnakeList)
```

```
SnakeList
['python', 'cobra', 'mamba', 'viper', 'adder']
Copy of SnakeList
['python', 'cobra', 'mamba', 'viper', 'adder']
```

Figure 21.6

There are other algorithms using more advanced constructs for copying an array which learners will study in the course in later years. Some of the more experienced learners may be able to design and write these programs at this stage. For example, a possible algorithm for a copy array function is:

1. array (source, destination, size):
2. for i from 0 to size-1:
3. destination[i] = source[i]

The copy() method can be used to swap two arrays using the swap algorithm met earlier.

Note the indentation of the steps

Example 7 - swap array algorithm**Algorithm**

1. A = first array
2. B = second array
3. Output A, B and label as original arrays
4. Temp = a copy of A
5. A = a copy of B
6. B = a copy of temp
7. Output A, B and label as arrays after swap

Code and output

```
#initiate arrays a and b, and print
a = [1,2,3,4,5]
b = [6,7,8,9,10]

print("The original arrays")
print("array a is ", a)
print("array b is ", b)
print("\n") #print a blank line

#swap array a with array b, and print
temp = a.copy()
a = b.copy()
b = temp.copy()

print ("The swapped arrays")
print ("array a is ", a)
print ("array b is ", b)
```

Output:

```
The original arrays
array a is [1,2,3,4,5]
array b is [6,7,8,9,10]
```

```
temp = list(a)
a = list(b)
b = list(temp)
```

Above code would also work.

Figure 21.7

Learning Tasks

Here are some tasks to help learners understand the two focal areas in week 21. Kindly take note of differentiated learning when applying these tasks.

Task 1 - Revision

Complete the table to give a suitable description, name and data type for a set of arrays:

Array description	Array name	Data type
The first 7 prime numbers		
The divisors of the number 12		
Monthly income for 2024		
	<i>favourite Songs()</i>	
Top 10 songs in the charts this month		
Answers to five True/False questions		
Answers to five multiple-choice questions		
		string
		float
		bool

Task 2

Convert a set of single lines of pseudocode to code. Make your code as efficient as you possible can.

An example is:

- Output a greeting to each of Victor's best friends, separating each greeting with a blank line, for example, *Hello, Lumo* (Assume that an array with the names of Victor's 5 best friends already exists and the array is called *friends*)

Task 3

Convert a single flowchart step to code using a programming language that you are familiar with. Make your code as efficient as possible.

An example is:

Set votes to
1, 3, 3, 5, 1, 2

Task 4

Use a set of given algorithms to complete the partially-completed corresponding programs. An example using Python code is:

Pseudocode to output the first element alphabetically stored in the array of the days of the week

Note that this algorithm re-orders the array; otherwise constructs beyond the Year 1 course will be required

- Set up *days*, an array of the names of the weekdays, Monday to Sunday
- Reorder the array in alphabetical order
- Output the first element in the sorted array

4. Output the last element in the sorted arrayCode to complete

#Set up days() array

#Order the array into alphabetical order

#Output the first and last element

Task 4

Correct the errors in the following programs and classify each error (syntax, runtime or logic).

One example of a Python program with errors is:

```
cars = ["Ford", "Volvo", "BMW", "Porsche"]
print(cars
print ("The first car is ", cars[0])
print ("The last car is ", cars[4])
```

Task 5

Design and write programs to match a set of problem specifications. Use pseudocode to illustrate your design. Make amendments, if necessary, to ensure that your final code and pseudocode match. Add comments to your code and test.

Two possible problem specification are:

1. A program is required to do the following:
 - Store the seven colours of the rainbow in an array
 - Make a copy of this array
 - Print out both arrays
2. A program is required to do the following:
 - Store a set of results of a poll of 6 family members' choices of birthday cake to buy for grandmother's birthday.
 - The six choices of cake are input by the user. The choices are: chocolate, lemon, and vanilla.
 - Output how many votes were cast for each type of cake.

Task 6

Implement an array swap algorithm

More complex tasks

Task 7

Design and write a program that sets up three parallel arrays: one for the names of 8 students, one for Computing test marks, and one Mathematics test marks. The student name, Computing mark, and Mathematic mark should be output for the position in the array entered by the user. For example, if the user entered 1, then the first student in names array would be printed as well as their two marks.

Enter and test your code. Add comments to explain your code.

Task 8

Design, code and test a program that performs the same function as Task 7 but uses a 2D array to store the test marks for Computing and Mathematics rather than two parallel arrays.

Task 9

Imagine that you are a teacher of a SHS Year 1 Computing class. Write a guide advising your students how to do the following:

- Write an algorithm in pseudocode
- Create a flowchart to illustrate an algorithm
- Implement a simple sequence algorithm using Python code
- Debug and test Python code

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher

1. Start with a *question and answer session* on arrays to see what learners remember about the array data structure.
2. *Demonstrate* using the Python shell and the data projector how to code setting up an array with known values. Use examples that the learners can relate to, such as an array of subjects taught in their school, an array of everyone in the class who has a name beginning with 'K'. Recap how to access a particular element in an array.
3. Divide the learners into small *groups*. They should work together to complete Learner Task 1. Each group should share their completed table with the whole class.
4. Using a *slideshow with code snippets*, the teacher should go through the six built-in list methods given in the notes. Note that `clear()` and `index()` methods were not used in the exemplars.
5. Allow the learners time to use the Python shell to *experiment with the code* relating to arrays that they have learned so far.

It is now time to explore how algorithms using sets of items of the same type can be implemented in Python.

6. Using the data projector and Examples 4 to 7 (or similar) from these notes, the teacher should explain how each line of pseudocode translates to Python code.
7. The importance of adding internal commentary to code should be reinforced.
8. The learners should be given the opportunity to experiment with these exemplar programs.
9. Discuss possible improvements, extensions to these programs, any alternative algorithms to match the given specifications.
10. Examples which implement an algorithm in flowchart format should be included here.

11. Learners should work in groups of no larger than three when completing the Learner Tasks 2 – 6. Teachers should ensure that all group members are actively participating and contributing. Also ensure that learners complete some of the activities individually. This will allow the teacher to check who is managing to convert from an algorithm to and from code successfully. Repetition of similar tasks can help understanding where a learner is experiencing difficulties. Encourage Learners to refer to paper guides or online guides such as W3Schools when they encounter issues they cannot solve before asking the teacher.
12. Learner Tasks 7 and 8 using parallel arrays and 2D arrays respectively are suitable tasks for learners who have demonstrated a high proficiency in programming arrays.
13. Learners should be given time to write a draft of a guide for Task 9 before coming together in their groups to produce a final guide. The final guides should be presented to the class.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the two focal areas.

DOK Level 1: Recall/Reproduction

1. What is an array?
 - a. A list of numbers
 - b. A list of strings
 - c. A series of memory locations, each with the same name, that hold related data
1. Complete the sentence: Arrays store multiple data values under one n_____, reducing the complexity of our program.
2. Every element in an array has the same data type.
 - a. true
 - b. false
3. How many elements would the array `temperatures(10)` hold?
4. What is a suitable data type for an array that stores the average daily temperatures recorded over a 5-day period?
5. In the array `primes (2, 3, 5, 7, 11)`, what would `print(primes[3])` output?
 - a. 5
 - b. 7
 - c. [2, 3, 5]
6. When creating the array `scores(16)`, what does the 16 in brackets represent?
7. An array `fruits ("pineapple", "banana", "mango")` exists in a Python program. Complete the line of code that would change "pineapple" to "papaya":


```
fruits[ ] = "papaya"
```

DOK Level 2: Skills and Concepts

1. a. Create an array called *seasons* in Python that, when output, will give:

```
['Spring', 'Summer', 'Autumn', 'Winter']
```

- b. Write the line of code that must be added to update the first element to 'SPRING'.
 - c. Write a line of code that will print the third element in the array.
 - d. Write one line of code that will sort the seasons array elements into alphabetical order
 - e. Test these lines in the Python shell to check your answers.
2. Implement the following array in Python:
 1. set1 = first array
 2. set2 = second array
 3. Output set1, set2 and label as original arrays
 4. tempSet = set1
 5. set2 = tempSet
 6. Output set1, set2 and label as arrays after swap.
 3. Using arrays for a list of related data is much more efficient than using separate variables. Give one reason why this statement is true.
 4. Complete the missing code in the given screenshot.

```
# Create a list of prime numbers
```

```
# reverse the order of list elements
```


5. Create a program using the completed version of this code.
Run the program to check that it works as expected.
6. Write down the output from the program shown in Figure 21:8.

```
#My first program usign Python lists
progLangs= ["PHP", "Python", "Java"]
print(progLangs)
print(progLangs[1])
print(progLangs[0])
progLang[0] = "JavaScript"
print(progLangs[0])
```

Figure 21.8**DOK Level 3: Strategic Thinking**

1. Design and write a program that will create an array of your five favourite foods and then sort this array. Both the unsorted and sorted array should be output with suitable headings.

2. Design a program that will
 - create an array called subjects consisting of three school subjects input by the user
 - swap the first and last subject element
 - output two versions of the array – original and after swap
3. Design and write a program that set up an array of the names of ten items (e.g. ten Ghanaian football teams) with repeats of some items. Ask the user to enter a team to search for (e.g. Asante Kotoko). Use in-built Python code to find how many occurrences of the search item are in the array. Output the results of the search.

DOK Level 4: Extended Thinking

1. Investigate each of the following data structures in Python:

- List
- Tuple
- Set
- Dictionary

Describe the main features of each of the listed data structures and the differences between them.

2. Research how arrays (lists) can be used in Python to implement the following data structures:

- A stack
- A linear queue
- A linked list

Use your research findings to implement one of the listed data structures using an array.

3. You are given a task to optimise an existing piece of software that frequently copies large arrays. Using what you know about the `swap` function and array copying, propose a strategy that could potentially enhance the software's performance. Justify your strategy with logical reasoning.

Section 4 Review

Section 4 has spanned nine weeks, from week 13 to week 21. It has concentrated on deepening the learners' understanding of aspects of computational thinking, in particular, the decomposition of problem specifications and the development and understanding of algorithms to solve problems. Teachers were tasked with guiding learners through a range of focal areas that relate to software development. A variety of algorithms were exemplified, mainly using pseudocode but some using flowcharts. The software development life cycle (SDLC) was examined with the greatest focus being on the design and implementation stages. The fundamentals of the programming language, Python, were studied with lots of examples of simple sequence programs. The main features of data structures ranging from simple 1D arrays to more complex non-linear structures such as trees and graphs were explored. Examples were given of how these structures can be used to implement algorithms in the program development cycle as well as in real-life settings. Only the implementation of 1D arrays in Python was covered at this stage of the course and without the use of loops.

The pedagogical exemplars in the section guided teachers in employing various instructional strategies, which included demonstrations of code and dry runs of algorithms. Many of the lessons, in particular, the lessons in weeks 19 to 21 were practical lessons, where the learners were required to use a computer and have access to an IDE for the programming language they were using.

The assessments in the section were strategically designed using the Depth of Knowledge (DOK) framework to gauge learners' mastery of the focal areas. Some of these assessments, both formative and summative, were practical and required the use of a computer. Wherever possible, every learner should have had the opportunity to do some individual coding.

In summary, computational thinking provides the mindset and problem-solving skills, while programming logic is the practical implementation of those skills in writing code. Both are essential for effective software development and problem-solving in the digital age. This section encouraged learners to reflect clearly on problems to be solved and to develop strategies and skills that are required to develop efficient solutions.

Teaching and Learning Resources

The teacher can integrate some proposed teachings and learning resource lists into their lesson.

- Paper tape
- Photos and diagrams
- Videos
- Smartphones
- Desktop/Laptop computers (preferably)
- Tablets with keyboard
- Open Educational Resources (Including YouTube, MOOCs-Udemy/Coursera, Khan Academy, TESSA)
- Python editor (examples listed in manual)
- www.replit.com
- www.w3schools.com

Reading

1. Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). *Data Structures and Algorithms in Python*. Wiley.
2. Lee, K. D., & Hubbard, S. (2015). *Data Structures and Algorithms with Python*. Springer. DOI: 10.1007/978-3-319-13072-9

Reference

1. Visual Paradigm. (n.d.). Flowchart Tutorial. Retrieved from <https://www.visual-paradigm.com/tutorials/flowchart-tutorial/>
2. MQL5. (2019, February 1). Hedge and correlation strategy. Retrieved from <https://www.mql5.com/en/blogs/post/713281>
3. Sedgewick, R., Wayne K. (2014). *Algorithms*. Pearson Education
4. Morin, P. (2013). *Open Data Structures: An Introduction* (Vol. 9). Athabasca University Press.

SECTION 5: COMPUTATIONAL THINKING AND WEB DEVELOPMENT

Strand: Computational Thinking (Programming Logic)

Sub Strand: Web Technologies and Databases

Content Standard: Demonstrate knowledge and understanding of Web Development

Learning Outcome: *Explain the basic principles of web design and identify the modern technologies used in web design.*

INTRODUCTION AND SUMMARY OF SECTION

This section spans three weeks, from week 22 to week 24. Over these three weeks, learners will develop knowledge, understanding, and practical problem-solving skills in web design, through a range of practical and investigative tasks. The focus of instruction must be on website design as dictated by the learner outcome; in particular, on web outline plans (including sitemaps, wireframes, and prototypes). However, a brief overview of the whole website development process appropriate for the Year 1 level should be included. This will emphasise the importance of the design stage in the creation and success of a website, and will help the learners understand why it is an iterative process. Web development tools to produce web pages is covered in the Year 2 and Year 3 of the Curriculum but an exemplar simple website that uses HTML, CSS, and JS is also included at this point. Demonstrating this website, or getting the learners to use it should focus the learners on possible components of a web page which is one of the focal areas. It may also increase engagement in this area of study and perhaps encourage some learners to experiment themselves with web development tools. There are many excellent online tutorial sites on HTML, CSS, and JS. These include www.w3schools.com and www.codecademy.com.

The breakdown of the weekly learning indicators is as follows:

Week 22: Identify and describe the key components of a Web page, including headings, menus, links, text, images, and other relevant elements.

Web development – an overview, including an exemplar of a simple website

Components of a web page - headings, menus, links, text, images, and other relevant elements

Week 23: Draw and explain 2 web outline plans showing the wireframes.

The role of a web designer

Web outline plan – What is it, and steps involved

Sitemaps – What is a sitemap? Exemplars of visual sitemaps

Week 24: Draw and explain 2 web outline plans showing the wireframes.

Web page wireframes – what are they, their purpose, and how to create them

Website prototypes – what are they and how they are created and used

Advantages of using wireframes and prototypes

SUMMARY OF PEDAGOGICAL EXEMPLARS

The pedagogical examples for this section are designed to guide teachers in enhancing learners' knowledge and understanding of web design using a variety of teaching and learning strategies. These strategies include direct teaching, video for learning, and group work. Teachers should have a bank of visual sitemap and wireframe exemplars in a variety of contexts available to demonstrate in lessons – there are a number in this manual that could be used. The Learner Tasks include instructions to draw sitemaps and wireframes. Again, having a bank of tasks similar to these is recommended. Learners should gain experience of drawing these diagrams using both pen/paper and a computer. Examining existing websites, their components, layout, and design features has an important role to play in these lessons. This could start as teacher led using projected websites, but learners individually or working in groups should also be given opportunities to do this. The exemplar website included in week 23 would fit better in SHS Years 2 and 3, but it is recommended here for reasons explained in the introduction.

Teachers are again encouraged to integrate GESI, SEL, and SEN in their lessons and classroom interactions to ensure that all learners, regardless of their personal and educational backgrounds, are provided with equitable opportunities to participate and excel in the lessons.

SUMMARY OF ASSESSMENT

This section applies the Depth of Knowledge (DOK) framework to evaluate learners' understanding and skills acquisition in web design. Teachers should build on the bank of formative and summative tasks contained in this manual. These tasks should be appropriate for the SHS Year 1 level and measure learners' comprehension of the focal areas. They can also be used by teachers to note areas where learners need further assistance or practice.

WEEK 22

Learning Indicator: *Identify and describe the key components of a Web page, including headings, menus, links, text, images, and other relevant elements.*

Theme or Focal Areas

1. The difference between web design and web development
2. Web development – an overview, including an exemplar of a simple website
3. Components of a web page - headings, menus, links, text, images, and other relevant elements

Key Concept Notes

The Difference Between Web Design and Web Development

Web design primarily concerns creating a website's look and feel whereas web development centres on the functionality of a website. Web developers bring the web designer's ideas to life by writing code. Web development, in particular, HTML, will explore in more detail in SHS Years 2 and 3. An overview is included here as well as an exemplar of a simple one-page website to allow a fuller understanding of how websites are created. Web design is explored in more detail in weeks 23 and 24.

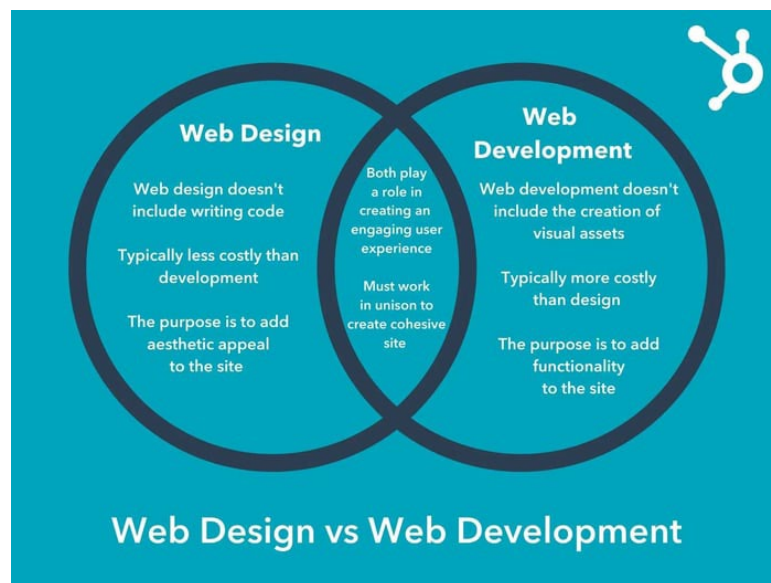


Figure 22.1

Web Development – An Overview

Web development is the process of building and maintaining websites and web applications. Various components and technologies are involved in this process, and the goal of web development is to meet the functional requirements of the client (who is paying the cost of the development). This generally means accurate, informative, reliable, interactive and user-friendly online experiences for the end-users.

When developing a website, there are two aspects – front-end development and back-end development:

Front-end Development

Front-end development focuses on the user interface and user experience of a website. Technologies commonly used in front-end development include HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript:

- **HTML (Hypertext Markup Language)** is the standard language for creating web pages. It provides the structure and content of a webpage. HTML uses bracketed codes called ‘tags’(e.g. <body>, <p>,) to tell the browser how to display the content.
- **CSS (Cascading Style Sheets)** is the language used to style the visual presentation of a document written in HTML. It allows for the customisation of colours, layouts, and fonts, among other visual aspects. The CSS code can be within the HTML document or, as shown in the exemplar website given later in these notes, it is in a separate file which has a link within the HTML document. This separation of the content and structure of a web page from its visual design, makes it much easier to apply consistent styling across multiple pages.
- **JS (JavaScript)** is a type of programming language called a scripting language. JavaScript provides interactivity and dynamic functionality to web pages. It allows for client-side scripting, these enabling actions such as form validation. JS is used for creating animations, interactive maps, providing real-time updates, and other features that enhance user experience.

HTML is like the skeleton of your website. CSS gives all the nice styled covering to the website. JS provides basic functionality to the website to make it come alive -see Figure 21:2.

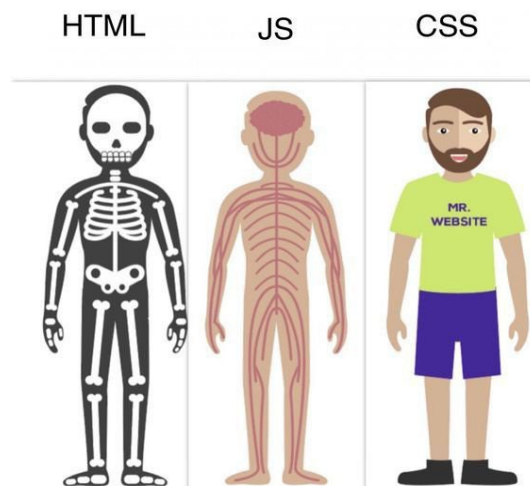


Figure 22.2

These three components are essential tools for creating modern, responsive, and interactive websites. A responsive web design ensures that the website will adapt to different screen sizes and devices, providing an optimal user experience regardless the device they are using.

Back-end Development:

Back-end development deals with server-side processes, database management, and application logic. It is where the data is stored; without it, there would be no front end. The back end of the web combines the server with a database to store the data, host the website, and an application for running it. Common back-end programming languages include Python, Ruby, Java, and PHP.

Web Hosting and Deployment

After development (which includes testing and adding security measures), websites need to be hosted on a web server to be accessible to users over the internet. Web hosting services provide server space and resources to store and run websites.

Deployment involves uploading and configuring the website on a web server for public access.

Users accessing the website

Each website has a unique URL (short for Uniform Resource Locator), commonly called its web address. The URL tells your browser where to go on the internet. When you type a URL into the browser's address bar and press Enter on your keyboard, the browser will load the page associated with that URL. A browser is software which is used to show web pages. See Figure 22:3 for examples of popular browsers.



Figure 22.3 Some popular browsers

An Exemplar website

This section goes beyond the scope of the Year 1 course to look at how HTML, CSS and JS fit together. How to use HTML to create simple web pages is covered in Year 2. This content is therefore optional at this point. However, an awareness of how a user interface is developed will help a learner better understand web design. In the real world, a web designer needs to be well-versed in HTML, CSS, and JS, even though it is a web developer who will be using this code.

This is a simple one-page website that illustrates how HTML, CSS and JS work together.

When this website is launched, JS code generates an input box asking the user to enter the name of their llama. Once this name has been entered, a pop-up window greeting to the owner of the llama appears. The user should click on OK in this window to make the web page appear. The structure and content of the page (heading, paragraph text, hyperlink, and image) come from the HTML code, the colour of the text and alignment from the CSS code –see Figures 22:4 and 22:5.

Note that this example uses external CSS, i.e. the code for the styling is in a separate file (called styles.css here). CSS can also be internal, that is it is within the HTML file. The various files for this website and how they are structured are shown in Figure 22:5. The convention is to use 'index' as the default filename for the home page of a website. Arranging elements horizontally on a web page is quite straightforward. Vertical arrangement of media, or a combination of vertical and horizontal is a bit more complex.

 The image shows two screenshots of a web form. The top screenshot shows the form with the heading "This page says", the text "Hello! What is the name of your llama?", and an empty text input field. Below the input field are two buttons: "OK" (blue) and "Cancel" (grey). The bottom screenshot shows the same form, but the input field now contains the text "Isaac".

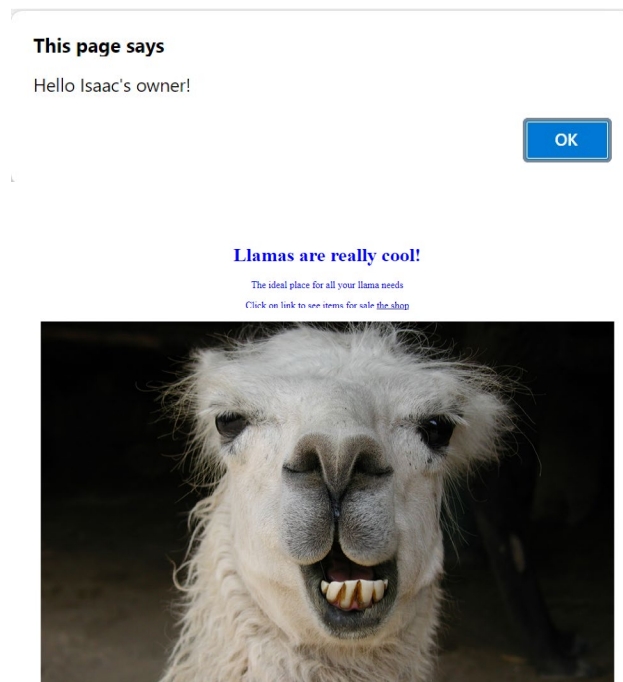


Figure 22.4: Browser output for the exemplar website

File Structure	File	Content/Code
css IMAGES JS index.html	css style.css	<pre>body { color: rgb(00,00, 255); text-align: center; }</pre>
	llama.jpg	
	JS script.js	<pre>var username = prompt("Hello! What is the name of your llma?"); alert("Hello " + username + "'s owner!")</pre>
	<pre>file <!doctype html> <html> <head> <link rel="stylesheet" href="css/style.css"> <script src="js/script.js"></script> <title>Llamas Page</title> </head> <body> <h1>Llamas are really cool!</h1> <p>The ideal place for all your llama needs</p> <a>Click on link to see items for sale the shop

 </body> </html></pre>	

Figure 22.5: Code behind the browser output

If an electronic copy of this manual is not available:

- Replicas of the .html, .css, and .js shown above could be quickly created use a text editor such as Notepad.
- A .jpg image of a llama could be sourced in Google Images – same filename unless changed in code
- Note the file structure should match unless file pathways are changed in the code
- Vary the content to suit the target learner audience
- Learners could also complete this practical activity

Components Of Web Pages

Components of a web page are often referred to as web elements. Examples include:

1. **Headings:** Organising web pages by headings helps users get a sense of the page's organisation and structure. Different sized headings provide a hierarchical organisation. Visually, headings are presented as larger and more distinct than surrounding text. Making texts larger helps users of the web pages navigate through sections, and scan for relevant information.
2. **Menus:** The top-level navigation of a website is called a menu or a navigation bar. On larger websites, there may be instances where multiple links will get stored under one main link, creating a dropdown menu. Menus allow users to access different sections or pages within a website, providing a consistent and intuitive way for users to explore and find the necessary information. A common presentation of a menu is as horizontal or vertical bars/lists of links. In the horizontal menu in Figure 22:6, the underlined HOME indicates that the page from the website currently being displayed is the HOME page

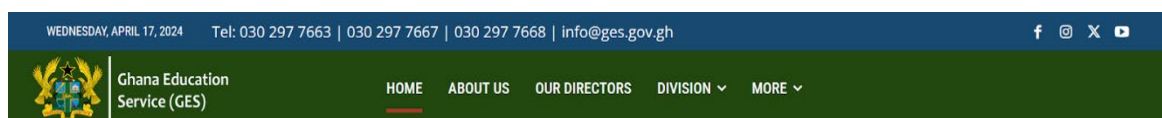


Figure 22.6: An extract from a website showing the header with a horizontal menu

3. **Links:** Links, also known as hyperlinks, are clickable elements that connect web pages. They allow users to navigate between different pages within a website or sections within the same page (internal hyperlink), or to an external website (external hyperlink). Hyperlinks can be text-based or represented by images. Links are a fundamental concept behind the World Wide Web which makes navigation between and within web pages easier. In Figure 22:6, the four options (HOME, ABOUT US, OUR DIRECTORS, DIVISION, and MORE) are examples of hypertext (text containing hyperlinks) which, when clicked, navigates the user to other pages on the website. The default format of an unclicked hyperlink is blue and underlined, but this can be easily changed by the developer. The cursor changes from an arrow to a hand when the mouse pointer goes over a hyperlink – see Figure 22:7.

The quick brown
fox jumps over
the lazy dog.

Figure 22.7

4. **Text:** Text is a primary means of conveying information on a web page. It can be used for headings, paragraphs, lists, and other textual content. Proper formatting, such as using appropriate font sizes, styles, and colours, helps improve readability and visual hierarchy. Textual content plays a significant role in communicating information, providing context, and guiding users through the website.
5. **Images:** Images enhance the visual appeal and engagement of a web page. They can convey information, illustrate concepts, or evoke emotions. Images should be properly optimised by the web developers. Image optimisation is about reducing the file size of the images as much as possible without sacrificing quality. The goal of optimisation is to make load times for the web page as low as possible. Alt text, also known as alternative text, is the use of text to describe the function or appearance of non-text elements like icons, charts, and images – see Figure 22:8. This text appears in place of an image on a webpage if the image fails to load on a user’s screen. This text also helps screen-reading tools describe images to visually impaired readers and can allow search engines to better crawl and rank a website. See Figure 22:5 to see how this is coded in HTML. The text ‘A cool llama’ will be displayed in place of an image of the llama if an image file `llam.jpg` cannot load.

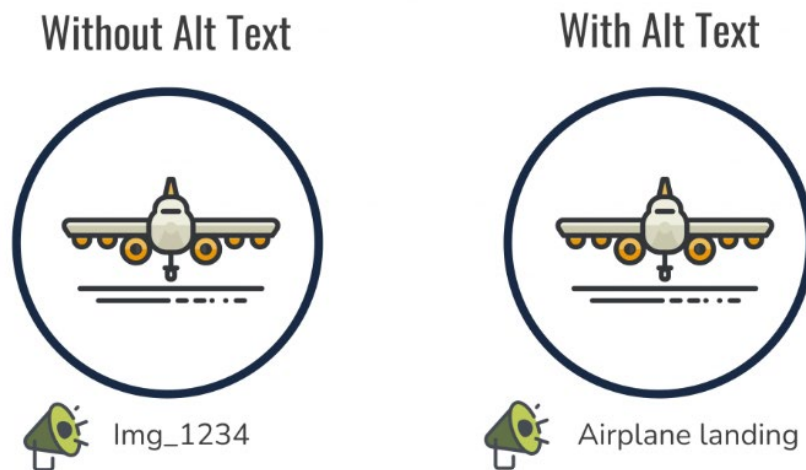


Figure 22.8

6. Other Relevant Elements

Various other elements contribute to a web page’s functionality and interactivity, including:

- **Headers/Footers:** The header and footer are typically the top and bottom sections, respectively, that remain the same- no matter which webpage a visitor navigates to on the website. These elements of a website typically define its style and design and contain links to other pages, the website’s logo or title and contact information. See Figure 22:6 and Figure 22:9 for the Header and Footer respectively of the website for the Ghanaian Education Service (in May 2024).

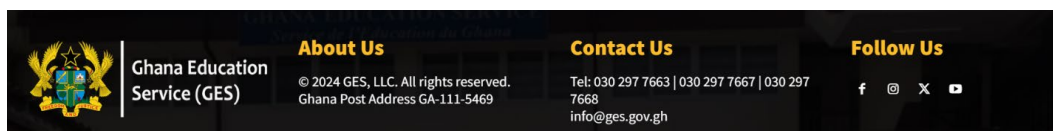


Figure 22.9: An example of a website footer

- **Forms:** Forms enable users to interact with the website by entering information, making selections, or uploading files. This data can then be submitted for processing. See Figure 22:10 for an example of a web form.

The image shows a web form with a yellow background. It is divided into several sections:

- Purchases**: Contains a 'Product Name' dropdown menu and a 'Quantity' text input field.
- Personal Information**: Contains 'First Name' and 'Last Name' text input fields.
- Credit Card Details**: Contains a 'Credit Card Type' section with radio buttons for 'Visa' and 'Mastercard', a 'Credit Card Number' text input field, and an 'Expiry Date' text input field.
- Billing Information**: Contains a 'Postal Address' text input field and a 'Home Address' section with a checkbox labeled 'Same as above'.

Figure 22.10: *An example of a web form*

- **Buttons:** Buttons provide interactive elements that trigger actions when clicked, such as submitting a form, navigating to a different page, or initiating a specific function.
- **Audio/video:** Audio and video elements can be incorporated into web pages to deliver engaging content and enhance user experience.
- **Web Widget:** Widgets elements are small, standalone programs that provide specific functionalities on a website. Examples include:
 - o Calendar website widget: displays a calendar view of events or appointments.
 - o Social media widget: displays updates, feeds, or recent posts from social media accounts. A social icons widget is a quick way to add icons to your website with links to your profiles on different social networks – see Figure 22:11.
 - o Website search widget: makes it easy for website visitors to find what they are looking for – very useful for content-heavy sites such as news sites.
 - o Digital clock widget: displays a digital clock.
 - o Weather widget: can give weather updates and forecasts.

Widgets save developers having to code these features from scratch. By embedding snippet of code in a website, existing software is utilised to make the website more engaging and successful.



Figure 22.11: *Social icons widgets*

Learning Tasks

Here are some tasks to help learners understand the three focal areas in week 23. Kindly take note of differentiated learning when applying these tasks

Task 1

Using a browser program, examine two different websites dictated by your teacher. Both these website should provide a similar function, for example, the websites of two football teams or the websites of two shoe shops. Poll the group members on which website they prefer and why. Collate the learners' responses and feed back to the class.

Task 2

Using printouts of the home page of the websites in Task 1 or another two websites, highlight/label the following components:

- A header
- A heading
- An image
- A footer
- A navigation bar or menu
- A hyperlink

Task 3

- a. State at least three possible uses of a web form in a website.
- b. Examine and then evaluate the feedback form on the Apple website, part of which is shown in Figure 22:12. What do you like/dislike about this form? Suggest possible improvements.

Figure 22:12

- c. Choose one of the websites listed below and describe a possible widget element that could be added. Describe what a website visitor could use this widget for.
 - A hotel website
 - A university or college website
 - A government department website

Task 4

Use the internet to explore some award-winning websites. Check out The Webby Awards and Awwwards for websites that have been nominated or have won awards for design and development.

For example, go to <https://www.awwwards.com/websites/Ghana/> for a selection of Awwwards winning websites in Ghana, such as the website for the Three Mountains Cocoa company at <https://threemountainscocoa.com/>. – see Figure 22:13 for parts of two pages from this website.

Create a slideshow that describes some elements from your favourite website from the websites explored to date this week. List reasons why you choose this website as your favourite.



Figure 22:13

Task 5

Choose one of the following topics to research and create a short report on your findings:

- Web content development
- Front-end web development
- Back-end web development

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Brainstorm what steps are involved in creating a website. This should lead to a whole class discussion on web development. The teacher could relate the process to the process of creating a program that was studied in a previous section – Analysis, Design, Implementation, Testing, and Maintenance.
2. The teacher can use a direct instruction approach with projected visual aids to outline the steps involved in the web development process. Possible resources include:

A short video that explains the difference between front-end and back-end development.



A graphic that could be used as a visual aid when discussing front-end and back-end web development and how they are different but connected.



3. Practical demonstration:

It is recommended that teachers demonstrate the exemplar website or similar. They could adapt the given HTML, CSS, and JS files to create their own website that would relate to the learners. Emphasise to the learners that the primary focus of this week's lesson is to understand components (elements) of a web page in preparation for a study of web design over the next two weeks. However, to be able to produce a good design for a website, an understanding of the coding required is important.

4. Use projected websites to identify the web page components outlined in this week's notes.

One example is the GES website – one way to access this website is to scan this QR code:.



Include some websites that will engage the learners. For example, if there are a number of football enthusiasts in the class, use <https://www.ghanafa.org/>.

Other websites (Epasmart and Jumia Ghana) that could be used here or in the tasks can be accessed by the teacher or learners using the QR codes:



6. Dividing the class into small groups. The learners in each group should work together to complete Tasks 1 to 4. These tasks could also be completed individually by the learners, if resources permit. Whatever the grouping, the teacher should guide feedback from each task and allow time for the learners to reflect on what they have learned.

7. Task 5 is an extension task for the more-able learners.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the three focal areas.

DOK Level 1: Recall/Reproduction

1. Complete the sentences:

- A _____ is a single document written in HTML on the web, while a _____ is a collection of multiple linked web pages.
- A website's _____ is the content area at the bottom of every page of a website whereas a website's _____ is the content area at the top of every page of a website.

- c. While website development adds functionality to the site through c_____, website d_____ seeks to enhance the user experience and interface and make the website one that customers/visitors will want to view.
 - d. The difference between front-end and back-end web development is that front-end serves the c_____ side (what we see on the front i.e. a screen) and back-end supports the s_____ side (what is under the hood of a website).
2. Name one thing that can be included with images on a web page to improve accessibility.
 3. State the name of the type of computer that stores websites on the internet and delivers web pages to viewers upon request.
 4. What software allows the user to view the web pages on the WWW? Give three examples.
 5. What is a set of links that allows users to find their way around a website?
 6. What is the first page of a website normally called?
 7. Name three possible components of a web page.
 8. What component of a web page helps you navigate and tells you what main pages are on the website?
 9. Every website has to have a footer.
 - a. true
 - b. false
 10. What is illustrated in Figure 22:14.



Figure 22.14

11. Expand the following acronyms:
 - a. HTML
 - b. CSS
 - c. JS
12. What are applications that are embedded into the body of websites commonly known as?
13. Complete the statement: Website a_____ is the practice of creating websites that are usable for all visitors regardless of a disability or impairment.
14. WoofWoof is a dog kennels business. The content area shown in Figure 22:15 appears at the bottom of each page of their website. What is this content area called?

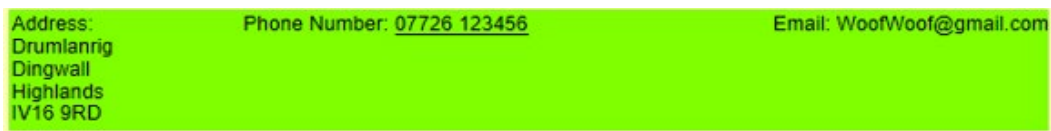


Figure 22.15

15. Highlight and label the following elements on the extract from website's home page shown in Figure 22.16:
 - a. An image
 - b. A form
 - c. A heading
 - d. A menu



Figure 22.16

16. Name three different widgets shown in Figure 22:17.

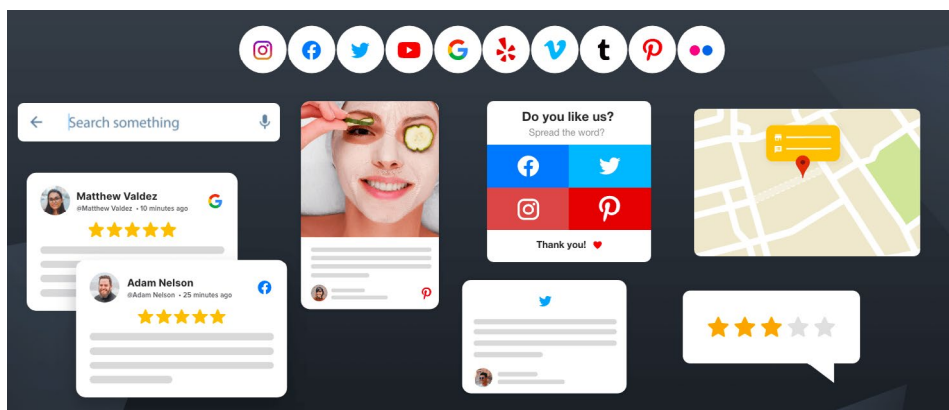


Figure 22.17

17. Web design comes before web development.

- a. true
- b. false

18. Which web element is used for collecting user inputs?

19. Identify the type of web element that allows users to submit a form.

DOK Level 2: Skills and Concepts

1. Explain the difference between a header and a heading on a website.
2. What items make up the header of the web page extract shown in Figure 22:16.
3. Explain how web design and web development differ and how they are linked.

4. WoofWoof is a dog kennels business. They are creating a website to advertise their business and process bookings. The following is the header for their website:

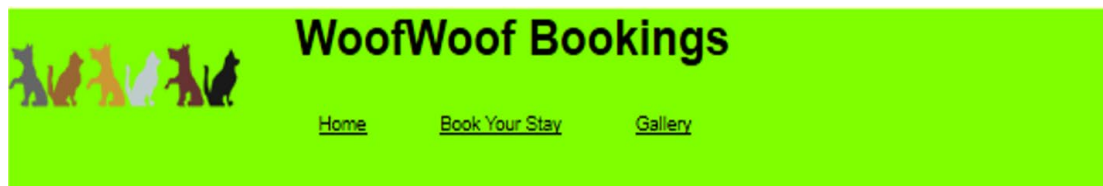


Figure 22.16

- Evaluate this header. Suggest two ways that it could be improved.
 - Identify three possible items that could be included in the footer for this website.
- An additional web page is being added to a school website about its recent sports day. Describe a possible video that could be included in this page.
 - Pretty Flowers, a chain of florists in Ghana, has a website to showcase their business. Identify three types of multimedia that could be included in their website and describe the purpose of this multimedia.
 - Describe how a user of a travel company's website could use three of the different widgets shown in Figure 22:17.
 - All the web pages in a bank's website contain a footer with a number of links. Describe two items that would be expected to be linked from the footer area of this website.

DOK Level 3: Strategic Thinking

- A website is being developed that will be used by runners to register online for an upcoming marathon. What component must this website have? Explain your reasoning.
- It is common practice to set the height of the header, navigation bar, and footer elements in a website to a fixed size. Explain how this would improve a website.
- A business having a social media presence can benefit both the business and its customers. Describe how a social media presence can be incorporated into a website.
- Thema uses the web form below to check whether there are rooms available to book at her preferred hotel. The entries in the form are validated before being processed.

Figure 22.17

- a. Describe two possible validation checks that could be performed on this form.
- b. Input validation is performed in the browser. Is this client-side or server-side validation?
- c. Identify one part of the website that is generated using a server-side script.
- d. Identify one part of the website that is generated using a client-side script.

DOK Level 4: Extended Thinking

1. Write a short guide on website accessibility. Consider a variety of potential website users in your guide, including those who are visually impaired, hard of hearing, colour-blind or elderly.
2. Propose a strategy for optimising a website that has issues with slow page loads, and discuss how you would implement changes in both the front-end and back-end.

(Learners are expected to provide a comprehensive plan that includes techniques like image compression and caching (front-end strategies), as well as database optimisation and server-side rendering (back-end strategies). They should explain the expected impact of these optimisations on the website's performance and discuss considerations for implementation, such as potential trade-offs between speed and functionality.)

WEEK 23

Learning Indicator: Draw and explain 2 web outline plans showing the wireframes.

Theme or Focal Areas

1. **The role of a web designer**
2. **Web outline plan – What is it, and steps involved**
3. **Sitemaps – What is a sitemap? Exemplars of visual sitemaps**

Key Concept Notes

Role of Web Designers

Web designers are responsible for designing the layout, usability, and visual appearance of a website. They focus on how the website looks, including aspects like images, colours, and fonts. They will use the brand identity that a brand designer has developed. They can create new websites from scratch, or simply make updates to the design and layout of existing pages. Web designers create the visual concept, while web developers bring that concept to life by writing the code and building the website (front-end and back-end). An important aspect of designing a website is to create a web outline plan. A developer may create the web outline if there is not a dedicated designer.

Web Outline Plan

What is a web outline plan?

A website outline plan, often just referred to as a web outline, is a detailed plan that maps out the structure, content, and functionality of a website before it is actually built. It can be considered a blueprint for a website. Producing a web outline plan is generally an iterative process, that is, a step or steps involved may be repeated, often multiple times to improve and refine the outcome. For example, based on user interaction with a prototype and feedback, changes are made to the design of the Home page. The agreed functional requirements between the client and the web development company will be a starting point for developing this plan.

General Guidelines in Developing a Web Outline Plan

1. *Identify website's goals and the target audience for the website.* Goals could include generating more traffic and boosting online sales.
2. *Create a user persona(s) for the website:* Develop a fictional representation of individuals or audience who are likely use the website. This persona will guide the content decisions and design choices. Figure 23:1 gives one possible template that could be used to create a user persona.

PERSONA NAME			
	Profile summary: Personality: Interests/behaviours:		Media/brands:
	Motivations:	Frustrations:	Influences: 
Age: Occupation: Status: Location: Archetype:			

Figure 23.1

3. *Categorise the information:* Group the web content into logical categories and subcategories. Think about the main topics or sections the website will cover. This will guide the number and names of the web pages and subpages.
4. *Create a sitemap:* This will lay out the basic structure and hierarchy of a website. See description and exemplars later in this section.
5. *Create a content outline:* This is like a detailed version of a sitemap. It will list out all the different types of content sections and functionality required by a website, page by page. It can be represented graphically such as shown in Figure 23:2 for the website, weignitegrowth.com. The navigation bar for this website when coded and displayed on a browser, is shown in Figure 23:3. A content outline could also be presented as a list as shown for the Home page of a different website in Figure 23:4.

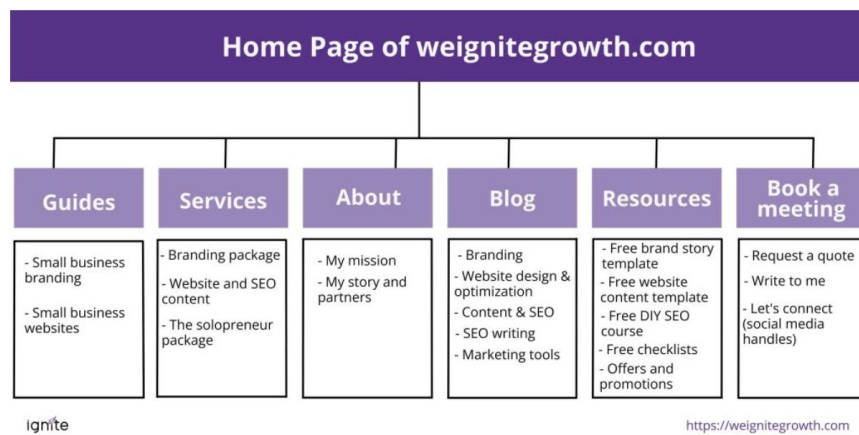


Figure 23.2

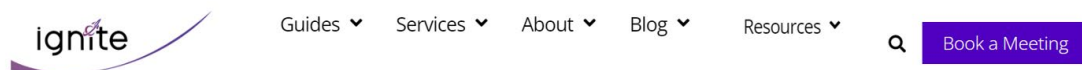


Figure 23.3

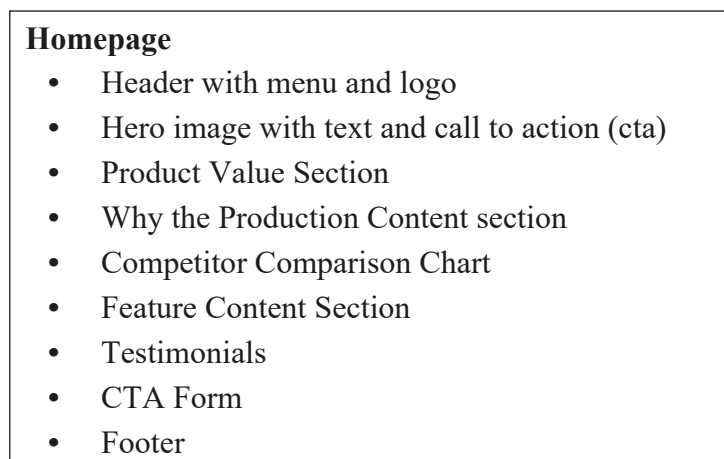


Figure 23.4

Note that CTA/cta stands for call to action which is a prompt that encourages visitors to take a desired action on a website. It is often designed in the form of a button with a clear command or action phrase.

6. *Wireframing and Prototyping:* Wireframes and prototypes help clients and developers visualise the layout and structure of web pages. Wireframes and website prototypes will be studied in week 24.

Sitemap

What is a sitemap?

A sitemap is a file that shows the structure of a website, including its pages and the relationships between them. Search engines use sitemaps to crawl websites more efficiently. Sitemaps focuses on the basic structure of the website, showing page hierarchy without distractions from visual design elements. It is an important part of planning a website's architecture and acts as a guide for web designers.

Nowadays, sitemaps are usually of two types visual and XML. At this level, we will only focus on visual sitemaps.

Exemplars of visual sitemaps

There are different ways that a website's structure can be illustrated. A visual sitemap is usually presented as a vertical or horizontal diagram with labelled boxes that represent different pages or sections of the website. These notes will focus on horizontal diagrams. It can be drawn using pen and paper, using Word, or using specialised software. The general advice from designers is that a web page should be no more than four levels deep. There is no point of having a web page if it is too hard for the user to access from the main menu or from multiple links on a site.

Example 1

See Figure 23:5 for an example of a horizontal sitemap with two levels drawn using Word. It consists of a home page with a navigation bar (indicated in this example by shading) that gives clear links to four main areas (pages). The BBC Bitesize page has a link to a sub-page (Quiz page).

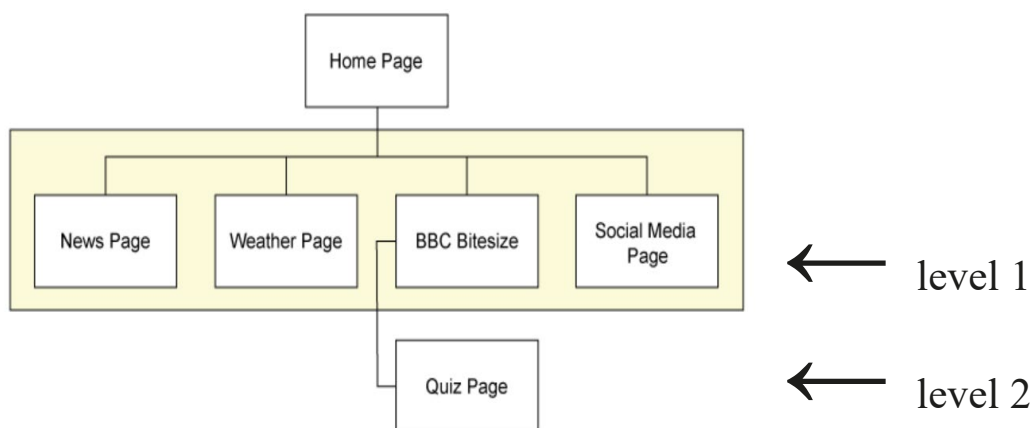


Figure 23.5: Sitemap example 1

Example 2

Barefoot is a business that sells shoes. The website being developed for this business should have a home page and three linked multimedia pages of the shoes currently on sale. These three pages (Men, Women and Children) should be navigated backwards and forwards. A link should exist to the www.howtomeasure.co.uk website from the home page.

A design of single-level visual sitemap for Barefoot's website, created manually, is shown in Figure 23:6. Note that the arrow to the external link is in one direction only. This indicates that there is no link from www.howtomeasure.co.uk back to the Barefoot's website.

The navigation bar is not indicated but is obvious in this case, and has links to three pages: Men, Women and Children.

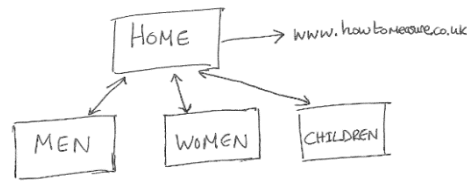


Figure 23.6: Sitemap example 2

Example 3

Digital Enhancers, an IT training company is developing a website for their customers about web development. A horizontal navigation bar will be used with links to three pages: Content & Structure, Styling, and Interactivity. Customers should be able to access further pages to learn about analysing or implementing the code for each of these three elements of web development.

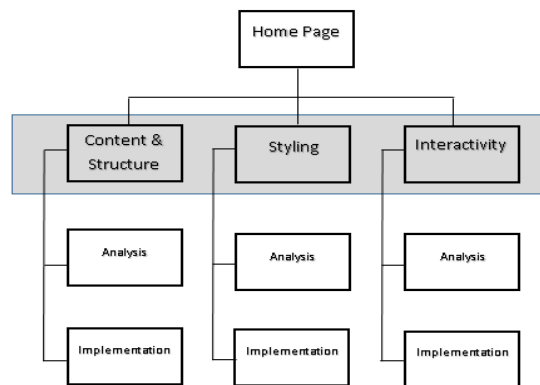


Figure 23.7: Sitemap example 3

A design a multi-level visual sitemap for the Digital Enhancer's website created in Word is shown in Figure 23:7. A dotted line is sometimes used to indicate the navigation bar. In this example and in example 1, the navigation bar is indicated by shading.

Next steps

Content creation: The content creators make or gather content for different parts of the website, and ensure that this content aligns with the website's goals and is engaging for the visitors. For example, a video editor may be commissioned to produce a promotional video for a university website. Copyright needs to be considered when making and gathering content for a website.

Development and testing: Web developers will use the web outline plan to code and test the website. Testers can adopt the role of the user personas as part of usability testing

Deployment and launch: Set up hosting, configure domain, and deploy the website to a live server. Conduct final checks to ensure all elements, functionality, and links work correctly. Create a backup and have a plan for ongoing maintenance and updates.

Analytics and monitoring: Implement website analytics to track user behaviour, traffic, conversions, and other relevant metrics. Monitor and analyse the data regularly to gain insights and make informed decisions. Use the data to improve the website's performance, user experience, and better achieve the defined goals of the website.

Conclusion

A web outline plan serves as a blueprint for the website's structure, content organisation, and navigation. It outlines the main sections, sub-sections, and the hierarchy of information to be presented on each page. The plan ensures a logical flow of content and helps users navigate through the website efficiently. It also guides the web development team, content creators, and designers during the website creation process. Thorough planning and understanding the target audience are crucial for creating a successful website. Thorough planning should mean using a web outline and will save you time and money, and lead to a more user-friendly site that ranks better in search engines.

Learning Tasks

Here are some tasks to help learners understand the three focal areas in week 23. Kindly take note of differentiated learning when applying these tasks.

Task 1 – Group of six activity

Give each member of the group a printed card with a different step in the creation a web outline plan. The learners rearrange themselves to give the correct order. Each learner should then describe the process for the step that they have been given.

Task 2 – Pairs activity (role play)

Assign the roles of client and designer. The teacher will issue a list of scenarios; for example, the client is a bakery owner who wants a website to showcase her products and process online orders for celebration cakes.

The designer will need to extract further details from the client and explain the process of how her website will be created.

Task 3 – Pairs or individual activity

Cuckoo Bakery is a chain of bakery shops. The new website being developed for this business. This website should have a home page and three linked multimedia pages of the products that the bakery currently sells. These three pages (Breads, Pastries, and Cakes) should be navigated backwards and forwards from the home page.

Create a single-level /top-level visual sitemap for this website.

Task 4

The owner of Cuckoo Bakery wants the customers to be able to access a further page on the website with an order form for a celebration cake.

Amend the sitemap from Task 3 to include this page.

Task 5

Create a diagram to show the structure of a website with the header as shown is Figure 23:8.

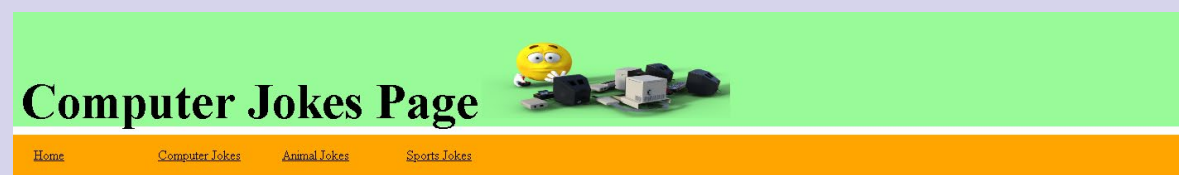


Figure 23:8

Task 6

Using given examples of general website specifications (such as that given in Task 3), complete a list of possible goals and at least two user personas for these websites.

See Task 3 for two examples of a bakery website's goals. An example of a user persona for this website could be Sisa, a busy 35 year old teacher who is mother to two young boys.

Task 7 – Extension task (multi-level website design)

The Ghanaian Laughter Society have commissioned a Jokes website. A horizontal navigation bar will be used with links to four pages: computers, animals, sports and voting. The website displays two chosen jokes of the week in three different categories (computers, animals, sports). On the voting page there is a form for users to vote for their favourite joke of the week from the website. The voting page also has a link to a sub-page which lists the top-voted jokes from previous weeks. There is a link to the laughter society website (ghanalaughs.org.gh) from the home page.

The structure of the website should include the following:

- a home page
- a computer page
- an animal page
- a sport page
- a voting page
- any subpages
- external links
- and the navigation bar

Draw a multi-level structure design for the Jokes website.

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Direct instruction: The teacher can use a slideshow with suitable visual aids to deliver the content of the three focal areas. The diagrams from this manual or similar can be used to illustrate visual sitemaps, both single-level/top-level and multi-level.
2. Demonstrations: The teacher should explain that visual sitemaps can be created using pen/pencil and paper, readily available software such as Word, or specialised software. The teacher could demonstrate creating a sitemap using a free online application such as the sitemap generator at www.lucidchart.com – see Figure 23:9.

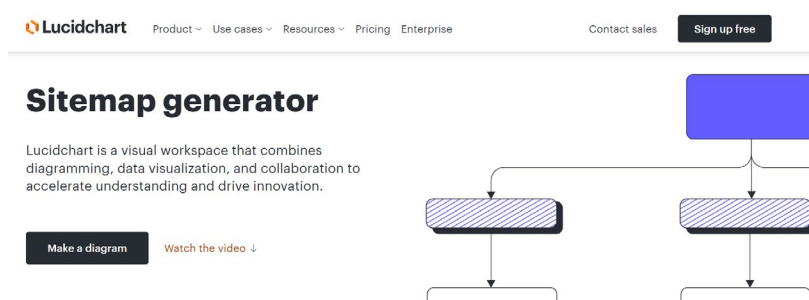


Figure 23.9

3. Projected websites: These can be used to identify a website's menu/ navigation. The teacher should guide the learners through drawing a corresponding top-level visual sitemap. (preparation for Learner Task 5)
4. Group work. The teachers should divide the class into groups of six to complete Learner Task 1 and manage the feedback from each group when this task is completed.
5. Role play: Working in pairs, the learners should complete Learner Task 2, Each learner The learners should reflect on their performance before swapping roles and moving on to a different scenario.
6. Two whole class brainstorm sessions: These could be used to explore possible goals and possible user personas for suggested website scenarios.
7. Video for Learning: A short video (approx. 4 minutes) on user personas on YouTube:
Personas for Understanding Your User | Understand the User | App Marketing | Udacity
8. Pair work: Each pair should work together to complete Tasks 3, 4, 5, and 6. Learners working at a lower proficiency level could focus on top-level sitemaps (Tasks 3 and 5).
9. Task 7 could be used as an extension task for the more able learners.
10. Class discussion: Conclude with a discussion how a web outline plan fits into the overall creation of a website. Scan this QR code for a possible resource.

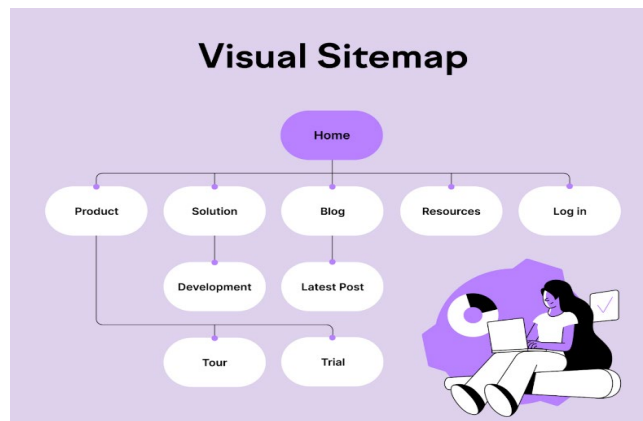


Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

DOK Level 1: Recall/Reproduction

1. Complete the sentences:
 - a. S_____ are created during the initial planning phase to show a website's structure, before diving deep into specific design details.
 - b. Once the website structure has been designed, the next step is to design each individual page using w_____ and content o_____.
2. Name a document/diagram that clearly shows the navigational structure of a website.
3. What is the plan/blueprint that covers a website's structure, content, and functionality before its development, more commonly known as?
4. The structure of a website which contains links from main pages to subpages, which may also contain more subpages is
 - a. a loop
 - b. linear
 - c. hierarchical
5. Which is created first – a visual sitemap or the wireframes for the web pages?
6. The number of levels of the sitemap shown in Figure 23:10 is:
 - a. 1
 - b. 3
 - c. 4



7. State what is being illustrated in Figure 23:11.

BIOGRAPHY
Sandeep is a PhD student at Jawaharlal Nehru University (JNU), a leading liberal arts university located in Delhi. He has always been dedicated to his studies because he believes in the value of a good education. He enjoys spending time with his professors and even thinks of them as friends.

Sandeep was introduced to computers in secondary school. He attended one of the most reputable English-medium private schools in his state, one for the children of military parents. There, he learned to type, conduct online research, and create high-quality reports and presentations.

His parents gave Sandeep a feature phone when he moved to Delhi to pursue a Bachelor's in History, because his mother wanted a means to stay in touch with him. After graduation, he enrolled in a Master's at JNU, and his friends convinced him to buy a smartphone so they could communicate on WhatsApp. Sandeep was initially hesitant to spend his meager JNU research stipend (Rs 3,000 or \$45 a month), but agreed to buy it after one of his professors hired him for a research project. His friends recommended he buy an LeEco Le 1s Android phone (Rs 12,000 or \$180) because it was a good value.

When on campus, Sandeep uses JNU's WiFi—he even has a private WiFi connection in his hostel room. When he's off campus, he has a data plan to access the internet on his phone. He uses a Dell laptop for research, writing, and studying for exams. He relies on his professors for academic information, but also supplements what they provide through Google searches and reliable sources (including IStOR.org and EPW.in, Economic and Political Weekly) recommended by his professors and peers.

For any research project, Sandeep begins by searching on Google. He typically starts with relatively broad search terms and, based on the results they yield, will make his queries more targeted over time. During his Master's program, he learned what types of sources can be trusted. As a result, he will only use Wikipedia for topic overviews and as a source for references.

DEVICE USE

LeEco Le 1s Android smartphone
PRIMARY USE: Voice calls, WhatsApp, following conversations between friends but rarely participates himself.
NETWORK: Internet usage is mostly via WiFi on university campus because it is free. Maintains data on his phone in case WiFi on campus is too slow or goes out. He pays Rs 288 a month for his plan.

Personal Dell laptop
PRIMARY USE: Uses it for research, writing papers, etc.
DETAILS: Goes online in his hostel, and carries it to campus where he accesses the university's internet.

NAME: Sandeep **AGE:** 28
FROM: Maharashtra, India
LIVES: Delhi, India
OCCUPATION: Research Scholar

Figure 23.11

8. Sitemaps can enhance the ranking of a website in search engine results.
 - a. true
 - b. false
9. List three examples of possible content that could be created for a garage website.

DOK Level 2: Skills and Concepts

1. Describe what is meant by the following:
 - a. Website structure
 - b. A hierarchical navigation website structure
2. Create a top-level visual sitemap for a website with the following header (Figure 23:12):



Figure 23.12

3. Mr. Habib, a computing science teacher, has created a website about AI (artificial intelligence).

The website has a home page and linked pages on four areas of AI: NLP, computer vision, robotics and machine learning. All pages can be navigated backwards and forwards. The file structure of the website is as shown in Figure 23:13. There is a link from the homepage (index.html) to the external website – www.google.ai.

Draw the navigational structure for this website. (see next page for solution)

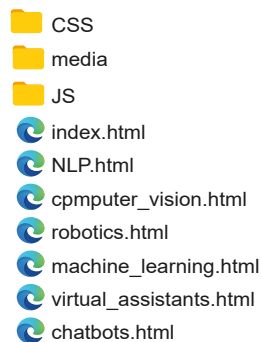


Figure 23.13

4. List three examples of possible content that could be created for a garage website. Explain what your chosen content would be used for on the website.
5. With respect to web design, give one difference between a sitemap and a content outline

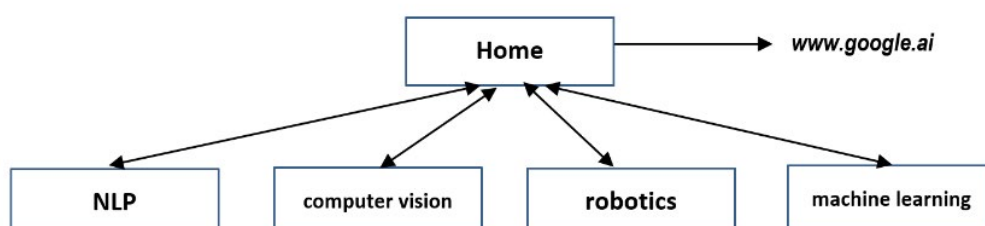
DOK Level 3: Strategic Thinking

- Write a job description for a web designer.
- Imagine that you are the owner of a new gym and you are working with a web designer to create a website for this business. Create a slideshow that will show the following for your website:
 - At least three goals
 - Three different user personas
 - A multi-level visual sitemap
 - A list of the items that you wish to appear on the Home page

DOK Level 4: Extended Thinking

- Research what is meant by an XML sitemap. Create a report on your findings. Include a description of how it differs from a visual sitemap and how search engines use XML sitemaps.
- Investigate the following and create a presentation summarising your findings:
 - The various techniques used in website usability testing such as the think-aloud protocol, co-discovery learning, and eye tracking.
 - Other types of website testing such as compatibility testing.

Solution to DOK Level 2 assessment question 3:



WEEK 24

Learning Indicator: Draw and explain 2 web outline plans showing the wireframes.

Theme or Focal Areas

- 1 **Web page wireframes – what are they, their purpose, and how to create them**
- 2 **Website prototypes – what are they and how they are created and used**
- 3 **Advantages of using wireframes and prototypes**

Key Concept Notes

Wireframes

What are wireframes?

Wireframes are visual representations of the layout and structure of webpages. A web page wireframe is a sketch outline of the information that needs to go on that web page. Wireframes should clearly show:

- navigational links
- text areas
- media used (including file format)
- position and type of hyperlinks on a page

They are simplified and grayscale designs that focus on the placement of elements, content placeholders, and functionality without the distractions of colours, images, or detailed graphics. Figure 24:1 shows a wireframe for a gym website.

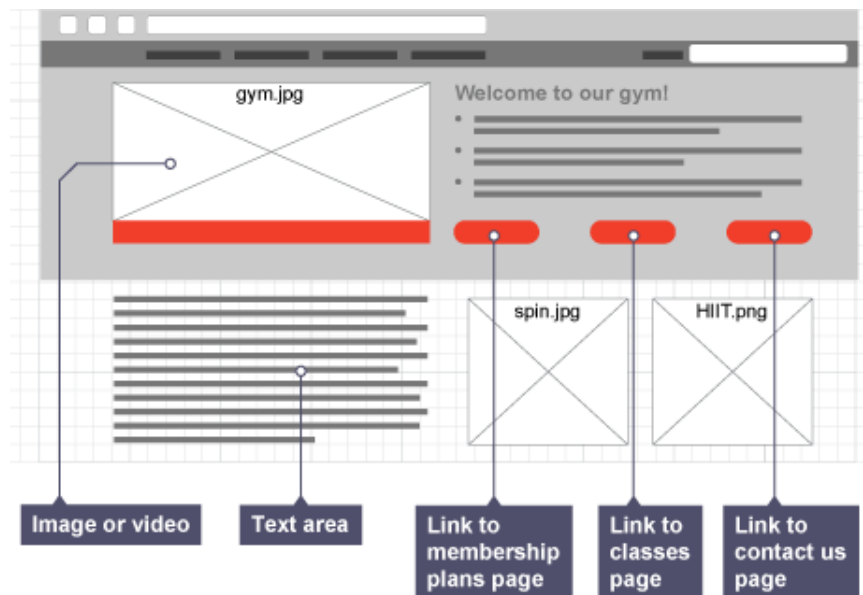


Figure 24.1

Box placeholders with an 'X' are used to signify images. A video placeholder will use a plain box with a centre play button or window controls, and an audio placeholder will also show media controls. Media file names should be shown.

Text areas are indicated by horizontal lines. Sometimes the text areas may include a summary of the text to be included or dummy text rather than the horizontal lines. Headings are included and aid understanding.

Links are indicated by a different colour and labelled. Wireframes may also include annotations giving some design specifications such as background colours, font details, and image sizes.

How to create a wireframe?

A wireframe could be created manually using pen/pencil and paper, or electronically using online/installed software tools (with templates). A screenshot of a digital wireframe template is shown in Figure 24:2.

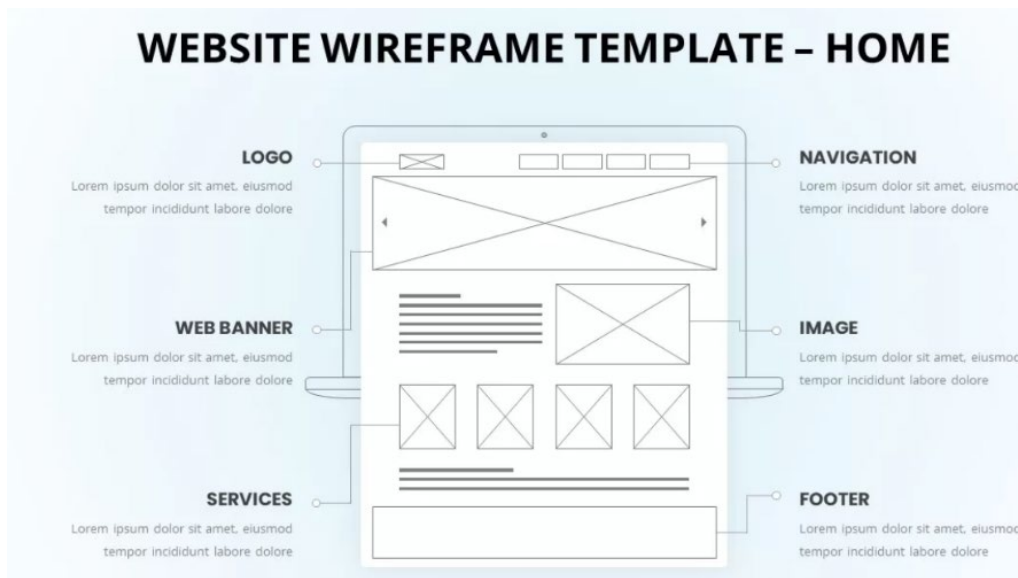


Figure 24.2

Prototypes

What is a website prototype?

A website prototype serves as a visual representation of a website design, and is used to demonstrate functionality and interactivity. The main difference between a wireframe and a low-fidelity prototype is that the latter offers interactivity

A *low-fidelity prototype* uses the wireframes and it is an easy way to translate rudimentary ideas from the wireframes into a basic testable product. A low fidelity prototype will often include more detail than a wireframe:

- text areas might contain all intended text
- image and video areas might contain hand-drawn versions of the final image/video still

A low fidelity prototype can be created by pen/pencil and paper and is quick and cheap to create. It can also be created electronically using presentation software, or specialist online/installed software (with templates).



Figure 24.3: Testing with a low-fidelity prototype

Using paper-based prototypes requires two people to demonstrate how the website should work. One person pretends to click on the ‘hyperlinks’ while the other person plays the role of the computer browser moving from page to page. At this stage, the user personas created earlier will be used.

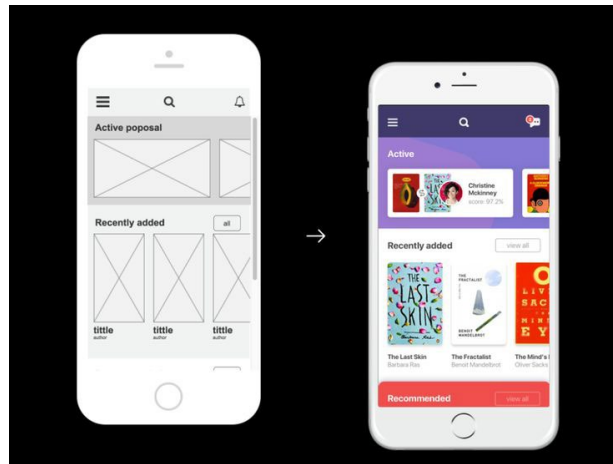


Figure 24.4: *Low-fidelity and high-fidelity prototype*

A *High-fidelity prototype* of a website is a highly interactive version of the website which has a large amount of functionality. It is quite close to how the final website will look and feel.

Advantages of using wireframes and prototypes?

It can be very helpful to show the wireframes of a site and prototype (low-fidelity and high-fidelity) demonstrations to a client before more advanced and expensive work begins. It enables the client to ensure the site has all the functions and information it needs to offer. If you just present a site design to a client, it is common for them to focus on how the site looks, which means they may not raise issues about its functionality until after the site has been built.

To summarise, some of the main advantages of using wireframes and prototypes in web design are:

- Identifying design flaws early on in the development process.
- Reducing the costs of the website development
- Receiving feedback from the client.
- Improving communication between the client and those creating the website
- Testing functionality and interactivity using prototypes
- Providing a guide to the web developers on how the website they are tasked to create should look like and behave.

Learning Tasks

Here are some tasks to help learners understand the three focal areas in week 24. Kindly take note of differentiated learning when applying these tasks

Task 1 – Think-pair-share activity

Question 1: What is the difference between a wireframe and prototype in web design and how could both be used in designing a website for your school.

Question 2: Where do wireframes and prototypes fit in a web outline plan?

Question 3: What happens in web development after wireframes and prototypes have been made?

Task 2 – Group Activity

Pretty Pictures is a business that takes photographs of family events and celebrations. This includes birthdays, weddings as well as photographs of other events.

The business wants a website containing the following elements:

- ◆ the title ‘Pretty Pictures’
- ◆ a short statement about the business’s quality assurance to customers
- ◆ separate pages for each category of photograph package: birthday, wedding and other special occasion
- ◆ a ‘Recommend a Friend’ page that gives details on how to register for rewards for customers and their friends
- ◆ at least one photograph on each page showing previous examples of images taken
- ◆ a video on the home page showing the business owner taking about their three packages on offer
- ◆ an external link on the ‘Recommend a Friend’ page to ‘The International Photography Awards’ website (www.tipa.com)

Design a top-level visual sitemap for this website and then a wireframe for each page.

Task 3 – Individual activity

Pedal Power is a charity bike shop. They would like a single page website to encourage donations of bikes and to advertise what bikes they have in stock.

This web page should have:

- the charity logo (PedalPower.jpg) and name at the very top of the page
- a heading with the title “We need your old bikes!”
- a graphic of a bike (bike.png)
- a subheading entitled “What we need”.
- a numbered list detailing the top 5 types of bikes the charity shop would like donated
- a subheading titled “Our current stock”
- a video showing the current stock (ppstock.mp4)
- a page background of pale blue/cyan

Other design specifications are at the discretion of the designer.

Draw a wireframe for this page.

Task 4 - Individual activity

Using a hand drawing, create a wireframe for the following:

- A web page about your family
- A web page about your favourite animal
- A Home page for a website about Ghana

Task 5 - Individual activity

Create at least one of the designs in Task 3 using the shapes in Microsoft Word or an online wireframing tool.

Task 6 – Pairs activity

Create a quiz to test the other learners on their knowledge and understanding of the Section 5 content. The quiz questions can be written by hand or created using a computer program such as PowerPoint, or online tool such as Kahoot or Quizlet. Learners should test the quiz questions on each other.

Task 7 – Individual and Group activity

Create a concept map on web design using the knowledge and understanding that you have gained over the past three weeks.

Pedagogical Exemplars

These examples are only to serve as a guide to the teacher.

1. Direct instruction: The teacher can use a slideshow with suitable visual aids to deliver the content of the three focal areas. The diagrams from this manual or similar can be used to illustrate wireframes. The topic of website prototypes should only be introduced when the learners have completed some of the Learner tasks and have demonstrated a grasp of wireframes.
2. Demonstrations: The teacher should explain that wireframes can be created using pen/pencil and paper, readily available software such as Word, or specialised software. The teacher could demonstrate creating a wireframe using a free online application such as the wireframe generator at www.lucidchart.com – see Figure 24:5.

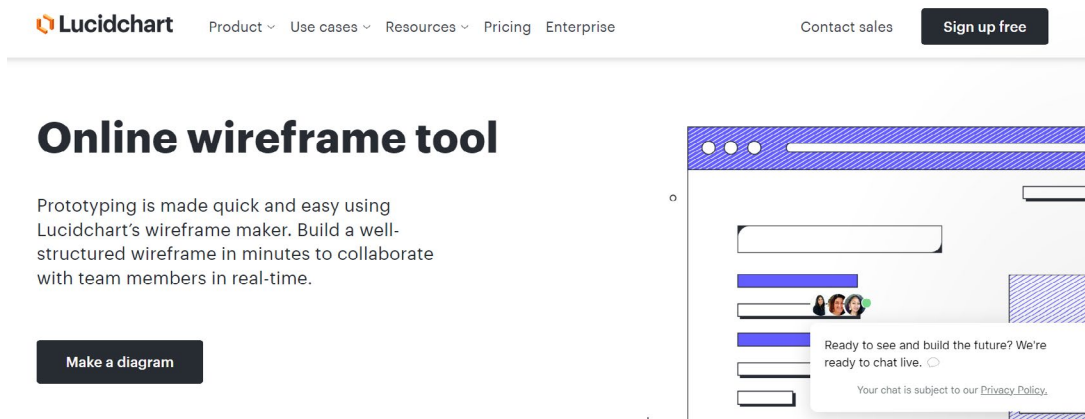


Figure 24.5

3. Group work. The teachers should divide the class into small groups complete Task 2 and manage the feedback from each group when this task is completed..
4. Individual work: The learners should then complete Learner Tasks 3 to 5 where they will be able to demonstrate understanding of how wireframes are created and have full creative control over the designs. The teacher should showcase the best (most accurate and creative) wireframes to the whole class.
5. Video for Learning: An example of a short YouTube video on web wireframes and prototypes (each approximately 3 minutes):

What is Low Fidelity Wireframes vs High Fidelity Wireframes in Figma**Low fidelity prototype testing of the EE app**

6. Think-pair-share and quiz: Each pair should work together to complete Tasks 1 and 6. The quiz questions could be used as a revision tool of the focal areas in this section.
7. Concept map: Task 7 should be completed individually by the learners and then a concept map should be created by a group with learners working collaboratively.
8. Plenary: The teacher should conduct a plenary session for the Section 5 lessons using a selection of the group concept maps.

Assessment

Teachers should assess learners during the learning process. Marks can be assigned to presentations, contributions during work, research projects, and more.

The summative assessment questions that follow are only to serve as a guide for the teacher when creating questions to measure learners' comprehension of the three focal areas.

DOK Level 1: Recall/Reproduction

1. Which is created first when a website is being designed – the wireframes of the web pages or a prototype of the website?
2. What is the purpose of a wireframe?
 - a. A text file that lists the content of a web page
 - b. A graphical design that shows the intended user interface and page layout prior to implementation
 - c. To hold a gallery of images for a web page
3. State two ways that a low-fidelity prototype of a website can be created.
 - a. State the name of the diagram shown in Figure 24:6.
 - b. Name one person who would use this diagram.
 - c. This diagram was created using pen and paper. State another method of creating this diagram.

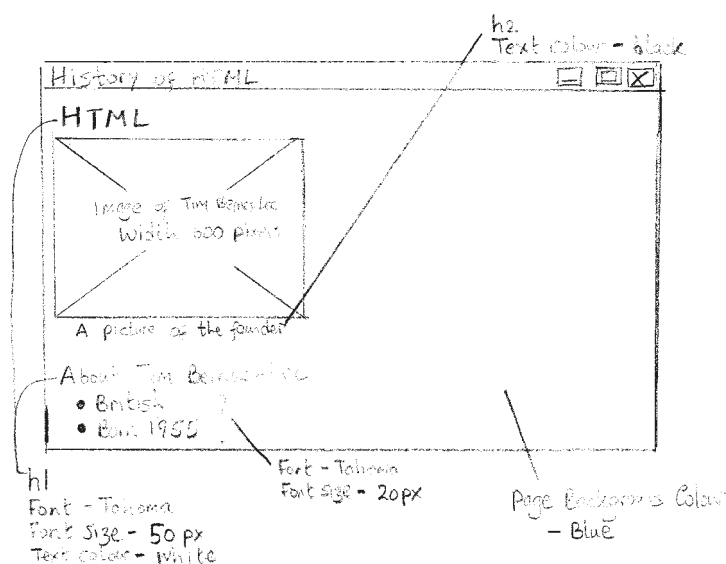


Figure 24.6

4. What is another name for quick sketches/outlines of page layouts that show navigation and content elements.
5. Name two things that would appear on a wireframe of a web page.

6. Complete the sentences:

P _____ are representations of a website that includes some functionality.

A _____ fidelity wireframe is when actual text and images are used.

7. Wireframes will come before any coding of the web pages.

- a. true
- b. false

9. Which type of prototype does not use actual text content or images?

10. The symbol to represent an image placeholder on a wireframe is

- a. 
- b. 
- c. 

11. Add another image placeholder to the wireframe in Figure 24:6 for the file called *computer.jpg*, with width of 400 pixels. The image placeholder should be placed to the right of the other image placeholder.**DOK Level 2: Skills and Concepts**

- When should wireframes of web pages be created in the development process of a new website? Justify your answer.
- Why do web designers seek feedback from the client on wireframes and prototypes?
- Describe two differences between wireframes and prototypes in web design.
- A web development company has been asked to develop a website for a film festival. Part of the documentation produced by the web designers in the company is shown in Figure 24:7.

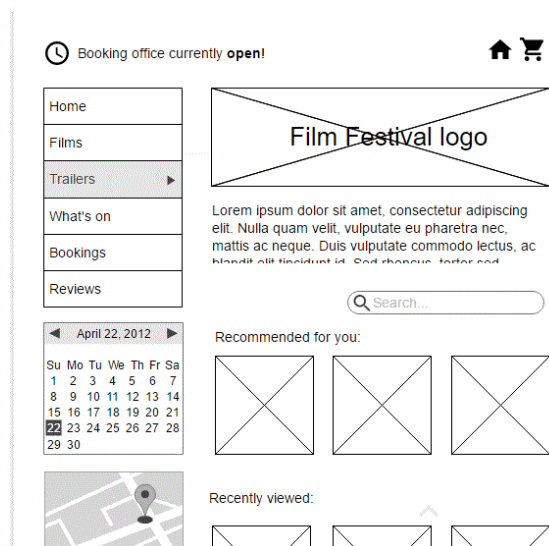


Figure 24.7

- Name the design notation that was used.
 - Identify two examples of dynamic content on the design above and explain how each could be used on the website.
5. Describe the purpose of wireframes in web development.

6. Explain why prototypes are used in web design.
7. Describe where wireframes fit in a web outline plan.
8. Explain how presenting wireframes to a client early in development can prevent future functional issues with a website.
9. Describe two reasons why the use of wireframing may save time at the implementation stage of the development of a website.
10. a. Compare the presentation of the wireframe given in Figure 24:8 to the presentation in Figure 24:6.
b. Annotate the wireframe in Figure 24:8 to add some design details.

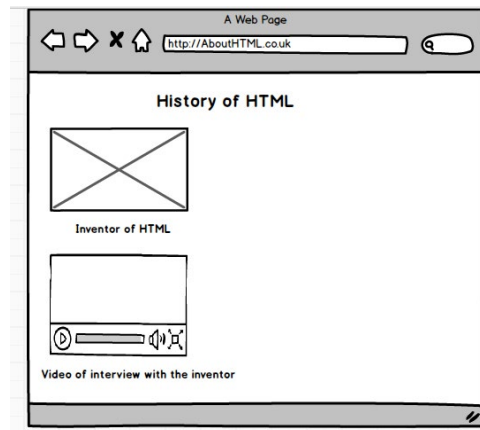


Figure 24.8

11. Camping is one of the activities run by a youth club. The ‘Camping’ page on the club’s website will include content of what is covered in this activity, as follows:
 - a link to a website which contains a video of how to put up a tent
 - a main heading ‘Camping Trip Preparation’
 - a numbered list detailing the kit list for the camping trip
 - a graphic called ‘tent.jpg’, 500 pixels by 400 pixels
 - a paragraph of text introducing the numbered list

Draw a wireframe design showing how you would position the above content on the web page.

DOK Level 3: Strategic Thinking

1. Evaluate how each of the main elements of a web outline plan can impact the effectiveness of a website.

DOK Level 4: Extended Thinking

1. Investigate how AI might impact web design in the future and if AI is a threat to web designer jobs.
2. Create the same wireframe (e.g. question 22). using two different digital tools (for example, Word and Lucidchart). Evaluate both methods based on your experience with this practical task.
3. Create a slideshow that identifies the typical personnel employed by a web development company and describe their roles.
4. Implementing an SSL certificate is one security feature that should be considered when designing an e-commerce site. Describe this feature and at least three other security features to be recommended. Your description should be written in terms that a non-techy client would understand.

Section 5 Review

In this section, the steps involved in web development were outlined, but the main focus was on web design. By the end of week 24, the learners should have a good understanding of what a web outline plan is and the significant part it plays in the development of a website. The learners will have been afforded many opportunities to draw visual sitemaps showing a website's structure and navigation, as well as wireframe designs of individual web pages. This knowledge and acquired skills will have laid the foundations for the continued study of this area of computing in Years 2 and 3: translating these designs into web pages using a web page editor.

Teachers were required to use various teaching methods, including direction instructions and group activities, supplemented by multimedia and software tools.

Attention was given to inclusive education strategies to support all students equally. A wide range of formative and summative methods of evaluation were included, guided by the DoK framework. These assessments will have allowed the teacher and the pupil to identify gaps in the pupils' learning, and to arrange the necessary support to target these gaps.

Teaching and Learning Resources

- Paper and card
- Photos
- YouTube videos
- Data projector
- Desktop/Laptop computers
- Tablets
- Smartphones
- Open Educational Resources (Including YouTube, MOOCs-Udemy/Coursera, Khan Academy, TESSA)
- Internet and browser software
- A number of exemplar websites e.g. <https://threemountainscocoa.com/>
- MS Word and MS PowerPoint
- Online sitemap and wireframe generators and templates, e.g. Lucidchart
- Text editor (e.g, Notepad) for showing Llama website and code

Additional Reading

1. Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). Data Structures and Algorithms in Python. Wiley.
2. Lee, K. D., & Hubbard, S. (2015). Data Structures and Algorithms with Python. Springer. DOI: 10.1007/978-3-319-13072-9

References

1. Ziff Davis. (n.d.). Widget. PCMag Encyclopedia. Retrieved from <https://www.pcmag.com/encyclopedia/term/widget>
2. Gaba, A. (n.d.). Streamlining Backend-Frontend Integration: A Quick Guide. Medium. Retrieved from <https://medium.com/@adityagaba1322/streamlining-backend-frontend-integration-a-quick-guide-145eca3cca05>
3. Ghana Education Service. (n.d.). Ghana Education Service. Retrieved from <https://ges.gov.gh/>
4. Doe, J. (2021, April 1). How to create a website outline & structure. We Ignite Growth. <https://weignitegrowth.com/how-to-create-a-website-outline-structure/>
5. SlideBazaar. (2022). Website wireframe infographic template. SlideBazaar. <https://slidebazaar.com/items/website-wireframe-infographic-template/>
6. <https://www.bbc.co.uk/bitesize/subjects/zft3d2p>

ACKNOWLEDGEMENTS

Special thanks to Professor Edward Appiah, Director-General of the National Council for Curriculum and Assessment (NaCCA) and all who contributed to the successful writing of the Teacher Manuals for the new Senior High School (SHS), Senior High Technical School (SHTS) and Science Technology, Engineering and Mathematics (STEM) curriculum.

The writing team was made up of the following members:

NaCCA Team	
Name of Staff	Designation
Matthew Owusu	Deputy Director-General, Technical Services
Reginald Quartey	Ag. Director, Curriculum Development Directorate
Anita Cordei Collison	Ag. Director, Standards, Assessment and Quality Assurance Directorate
Rebecca Abu Gariba	Ag. Director, Corporate Affairs
Anthony Sarpong	Director, Standards, Assessment and Quality Assurance Directorate
Uriah Kofi Otoo	Senior Curriculum Development Officer (Art and Design Foundation & Studio)
Nii Boye Tagoe	Senior Curriculum Development Officer (History)
Juliet Owusu-Ansah	Senior Curriculum Development Officer (Social Studies)
Eric Amoah	Senior Curriculum Development Officer (General Science)
Ayuuba Sullivan Akudago	Senior Curriculum Development Officer (Physical Education & Health)
Godfred Asiedu Mireku	Senior Curriculum Development Officer (Mathematics)
Samuel Owusu Ansah	Senior Curriculum Development Officer (Mathematics)
Thomas Kumah Osei	Senior Curriculum Development Officer (English)
Godwin Mawunyo Kofi Senanu	Assistant Curriculum Development Officer (Economics)
Joachim Kwame Honu	Principal Standards, Assessment and Quality Assurance Officer
Jephtar Adu Mensah	Senior Standards, Assessment and Quality Assurance Officer
Richard Teye	Senior Standards, Assessment and Quality Assurance Officer
Nancy Asieduwaa Gyapong	Assistant Standards, Assessment and Quality Assurance Officer
Francis Agbalenyo	Senior Research, Planning, Monitoring and Evaluation Officer
Abigail Birago Owusu	Senior Research, Planning, Monitoring and Evaluation Officer
Ebenezer Nkuah Ankamah	Senior Research, Planning, Monitoring and Evaluation Officer

NaCCA Team	
Name of Staff	Designation
Joseph Barwuah	Senior Instructional Resource Officer
Sharon Antwi-Baah	Assistant Instructional Resource Officer
Dennis Adjasi	Instructional Resource Officer
Samuel Amankwa Ogyampo	Corporate Affairs Officer
Seth Nii Nartey	Corporate Affairs Officer
Alice Abbew Donkor	National Service Person

Subject	Writer	Designation/Institution
Home Economics	Grace Annagmeng Mwini	Tumu College of Education
	Imoro Miftaw	Gambaga Girls' SHS
	Jusinta Kwakyewaa (Rev. Sr.)	St. Francis SHTS
Religious Studies	Dr. Richardson Addai-Mununkum	University of Education Winneba
	Dr. Francis Opoku	Valley View University College
	Aransa Bawa Abdul Razak	Uthmaniya SHS
	Godfred Bonsu	Prempeh College
RME	Anthony Mensah	Abetifi College of Education
	Joseph Bless Darkwa	Volo Community SHS
	Clement Nsorwineh Atigah	Tamale SHS
Arabic	Dr. Murtada Mahmoud Muaz	AAMUSTED
	Dr. Abas Umar Mohammed	University of Ghana
	Mahey Ibrahim Mohammed	Tijjaniya Senior High School
French	Osmanu Ibrahim	Mount Mary College of Education
	Mawufemor Kwame Agorgli	Akim Asafo SHS
Performing Arts	Dr. Latipher Osei Appiah-Agyei	University of Education Winneba
	Desmond Ali Gasanga	Ghana Education Service
	Chris Ampomah Mensah	Bolgatanga SHS, Winkogo
Art and Design Studio and Foundation	Dr. Ebenezer Acquah	University for Education Winneba
	Seyram Kojo Adipah	Ghana Education Service
	Dr. Jectey Nyarko Mantey	Kwame Nkrumah University of Science and Technology
	Yaw Boateng Ampadu	Prempeh College
	Kwame Opoku Bonsu	Kwame Nkrumah University of Science and Technology
	Dzorka Etonam Justice	Kpando Senior High School

Subject	Writer	Designation/Institution
Applied Technology	Dr. Sherry Kwabla Amedorme	AAMUSTED
	Dr. Prosper Mensah	AAMUSTED
	Esther Pokuah	Mampong Technical College of Education
	Wisdom Dzidzienyo Adzraku	AAMUSTED
	Kunkyuuri Philip	Kumasi SHTS
	Antwi Samuel	Kibi Senior High School
	Josiah Bawagigah Kandwe	Walewale Technical Institute
	Emmanuel Korletey	Benso Senior High Technical School
	Isaac Buckman	Armed Forces Senior High Technical School
	Tetteh Moses	Dagbon State Senior High School
	Awane Adongo Martin	Dabokpa Technical Institute
Design and Communication Technology	Gabriel Boafo	Kwabeng Anglican SHTS
	Henry Agmor Mensah	KASS
	Joseph Asomani	AAMUSTED
	Kwame Opoku Bonsu	Kwame Nkrumah University of Science and Technology
	Dr. Jectey Nyarko Mantey	Kwame Nkrumah University of Science and Technology
	Dr. Ebenezer Acquah	University for Education Winneba
Business Studies	Emmanuel Kodwo Arthur	ICAG
	Dr. Emmanuel Caesar Ayamba	Bolgatanga Technical University
	Ansbert Baba Avoles	Bolgatanga Senior High School, Winkogo
	Faustina Graham	Ghana Education Service, HQ
	Nimako Victoria	SDA Senior High School, Akyem Sekyere
Agriculture	Dr. Esther Fobi Donkoh	University of Energy and Natural Resources
	Prof. Frederick Adzitey	University for Development Studies
	Eric Morgan Asante	St. Peter's Senior High School
Agricultural Science	David Esela Zigah	Achimota School
	Prof. J.V.K. Afun	Kwame Nkrumah University of Science and Technology
	Mrs. Benedicta Carbiliba Foli	Retired, Koforidua Senior High Technical School
Government	Josephine Akosua Gbagbo	Ngleshie Amanfro SHS
	Augustine Arko Blay	University of Education Winneba
	Samuel Kofi Adu	Fettehman Senior High School

Subject	Writer	Designation/Institution
Economics	Dr. Peter Anti Partey	University of Cape Coast
	Charlotte Kpogli	Ho Technical University
	Benjamin Agyekum	Mangoase Senior High School
Geography	Raymond Nsiah Asare	Methodist Girls' High School
	Prof. Ebenezer Owusu Sekyere	University for Development Studies
	Samuel Sakyi Addo	Achimota School
History	Kofi Adjei Akrasi	Opoku Ware School
	Dr. Anitha Oforiwah Adu-Boahen	University of Education Winneba
	Prince Essiaw	Enchi College of Education
Ghanaian Language	David Sarpei Nunoo	University of Education Winneba, Ajumako
	Catherine Ekuu Mensah	University of Cape Coast
	Ebenezer Agyemang	Opoku Ware School
Physical Education and Health	Paul Dadzie	Accra Academy
	Sekor Gaveh	Kwabeng Anglican Senior High Technical School
	Anthonia Afosah Kwaaso	Junkwa Senior High School
	Mary Aku Ogum	University of Cape Coast
Social Studies	Mohammed Adam	University of Education Winneba
	Simon Tengan	Wa Senior High Technical School
	Jemima Ayensu	Holy Child School
Computing and Information Communication Technology (ICT)	Victor King Anyanful	OLA College of Education
	Raphael Dordoe Senyo	Ziavi Senior High Technical School
	Kwasi Abankwa Anokye	Ghana Education Service, SEU
	Millicent Heduvor	STEM Senior High School, Awaso
	Dr. Ephriam Kwaa Aidoo	University for Education Winneba
	Dr. Gaddafi Abdul-Salaam	Kwame Nkrumah University of Science and Technology
English Language	Esther O. Armah	Mangoase Senior High School
	Kukua Andoh Robertson	Achimota School
	Alfred Quaittoo	Kaneshie Senior High Technical School
	Benjamin Orrison Akrono	Islamic Girls' Senior High School
	Fuseini Hamza	Tamale Girls' Senior High School
Intervention English	Roberta Emma Amos-Abanyie	Ingit Education Consult
	Perfect Quarshie	Mawuko Girls Senior High School
	Sampson Dedey Baidoo	Benso Senior High Technical School

Subject	Writer	Designation/Institution
Literature-in-English	Blessington Dzah	Ziavi Senior High Technical School
	Angela Aninakwah	West African Senior High School
	Juliana Akomea	Mangoase Senior High School
General Science	Dr. Comfort Korkor Sam	University for Development Studies
	Saddik Mohammed	Ghana Education Service
	Robert Arhin	SDA SHS, Akyem Sekyere
Chemistry	Ambrose Ayikue	St. Francis College of Education
	Awumbire Patrick Nsobila	Bolgatanga SHS, Winkogo
	Bismark Tunu	Opoku Ware School
	Gbeddy Nereus Anthony	Ghanata Senior High School
Physics	Dr. Linus Labik	Kwame Nkrumah University of Science and Technology
	Henry Benyah	Wesley Girls High School
	Sylvester Affram	Kwabeng Anglican SHS
Biology	Paul Beeton Damoah	Prempeh College
	Maxwell Bunu	Ada College of Education
	Ebenezer Delali Kpelly	Wesley Girls' SHS
	Doris Osei-Antwi	Ghana National College
Mathematics	Edward Dadson Mills	University of Education Winneba
	Zacharia Abubakari Sadiq	Tamale College of Education
	Collins Kofi Annan	Mando SHS
Additional Mathematics	Dr. Nana Akosua Owusu-Ansah	University of Education Winneba
	Gershon Mantey	University of Education Winneba
	Innocent Duncan	KNUST SHS
Intervention Mathematics	Florence Yeboah	Assin Manso SHS
	Mawufemor Adukpo	Ghanata SHS
	Jemima Saah	Winneba SHS
Robotics	Dr. Eliel Keelson	Kwame Nkrumah University of Science and Technology
	Dr. Nii Longdon Sowah	University of Ghana
	Isaac Nzoley	Wesley Girls High School
Engineering	Daniel K. Agbogbo	Kwabeng Anglican SHTS
	Prof. Abdul-Rahman Ahmed	Kwame Nkrumah University of Science and Technology
	Valentina Osei-Himah	Atebubu College of Education



Subject	Writer	Designation/Institution
Aviation and Aerospace Engineering	Opoku Joel Mintah	Altair Unmanned Technologies
	Sam Ferdinand	Afua Kobi Ampem Girls' SHS
Biomedical Science	Dr. Dorothy Yakoba Agyapong	Kwame Nkrumah University of Science and Technology
	Jennifer Fafa Adzraku	Université Libre de Bruxelles
	Dr. Eric Worlawoe Gaba	Br. Tarcisius Prosthetics and Orthotics Training College
Manufacturing Engineering	Benjamin Atribawuni Asaaga	Kwame Nkrumah University of Science and Technology
	Dr. Samuel Boahene	Kwame Nkrumah University of Science and Technology
	Prof Charles Oppon	Cape Coast Technical University
Spanish	Setor Donne Novieto	University of Ghana
	Franklina Kabio Danlebo	University of Ghana
	Mishael Annoh Acheampong	University of Media, Art and Communication
Assessment	Benjamin Sundeme	St. Ambrose College of Education
	Dr. Isaac Amoako	Atebubu College of Education
Curriculum Writing Guide Technical Team	Paul Michael Cudjoe	Prempeh College
	Evans Odei	Achimota School

