Computer Science

Year 1

SECTION

4

COMPUTATIONAL THINKING AND PROGRAMMING LOGIC

Computational Thinking (Programming Logic)

Algorithm and Data Structure

INTRODUCTION

Hello Learner, welcome to an exciting journey into the world of computational thinking and programming! This section is designed to introduce you to the fundamentals of creating algorithms and writing code, primarily using Python. As you embark on this adventure, you will learn how to tackle problems by breaking them down, just like computer programmers do when they develop software.

We will start by mastering basic programming concepts, such as variables, data types, and arrays. These are the foundations you will use to build more complex programs as you progress. Your teacher will guide you in using tools like pseudocode and flowcharts to visually represent your solutions, making complex ideas simpler to understand and implement.

Remember, learning to code is like learning a new language; it is about practice and persistence. So, stay curious, ask questions and take advantage of this opportunity to learn skills that are crucial in today's digital world.

At the end of this section, you will be able to:

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
3. Explain in detail the concepts of Data Structures and their importance in organising and manipulating data efficiently
4. Differentiate between the types of Data Structures

Key Ideas

1. **Variables** are named storage locations that hold a value or an object which can be changed.
2. **Computer programming** is the process of writing code that tells a computer, application, or software program what to do.
3. **Algorithms** are the steps by way of solving a problem.
4. **Flowcharts** are the use of shapes to represent the steps of solving a problem.

5. **Pseudocode** is the use of plain text to write algorithms.
6. **Analysis** is the initial stage where the problem or project requirements are thoroughly understood and defined
7. **SDLC (Software Development Life Cycle):** is a structured process used by software developers and project managers to design, develop, test and deploy software**.**
8. **Data types** is the category that tells a computer what kind of data you're working with.
9. **Data structures** are a way of organising and storing data so that it can be used efficiently
10. **Arrays** are a data structure that stores a collection of the same elements, typically of the same data types, in a specific order.

VARIABLES IN COMPUTING PROGRAMMING

Earlier in the course, we learnt that:

1. A computer program is a set of instructions used to solve a given problem.
2. A RAM chip on a computer's motherboard has storage locations. These locations hold data.

In programming, when we talk about variables, we mean information that can change. For instance, if a website like the SHS admission portal request that you enter your index number, the answer will be different based on who is using the website (program). So, variables are part of a program that can have different values each time the program is used.

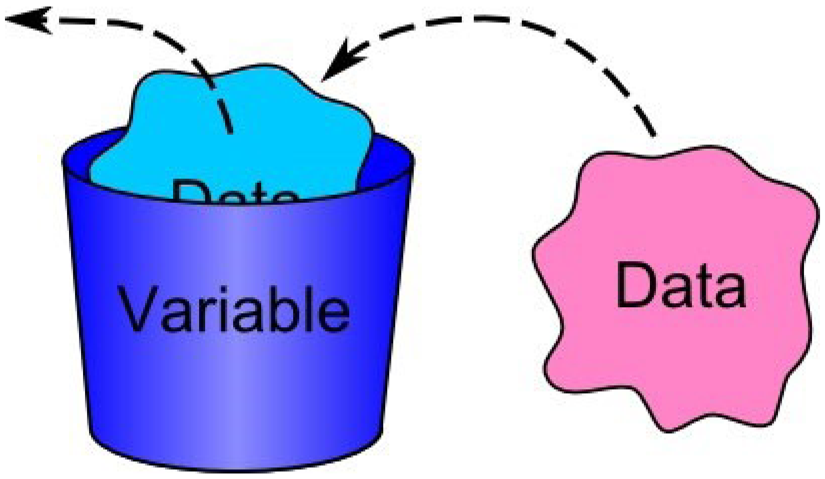


Figure 4.1: A variable is like a container. Its variable can change

Variable Naming

1. Choosing appropriate names for variables is important because it makes your program easier to understand.
2. Always use variable names that clearly describe what they hold.

For example, use ‘Form1_Class_Register’ instead of ‘x’ because it tells you what the variable represents.

It is important to note that each programming language has its own set of rules and conventions for naming variables. It is important to learn these rules for the language you are using.

Most programming languages, including Python, will not permit variable names to start with a number and 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** or firstName would be acceptable. camelCase is a way to separate the words in a phrase by making the first letter of the second and subsequent words capitalised and not using spaces.

Activity 4.1

1. Identify and share with a friend 2 daily routines that could be described as a step-by-step process of solving a problem(algorithm).

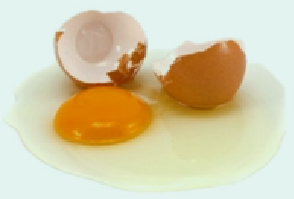
   E.g.: Frying an egg
2. Identify which of the following algorithms for frying an egg is correct

A **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



B **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

Note that the process of frying an egg put together becomes an algorithm.

Algorithms

An algorithm is a series of steps that you follow to solve a problem or complete a task. Think of it like a recipe (instructions) in cooking that tells you what to do, step by step.

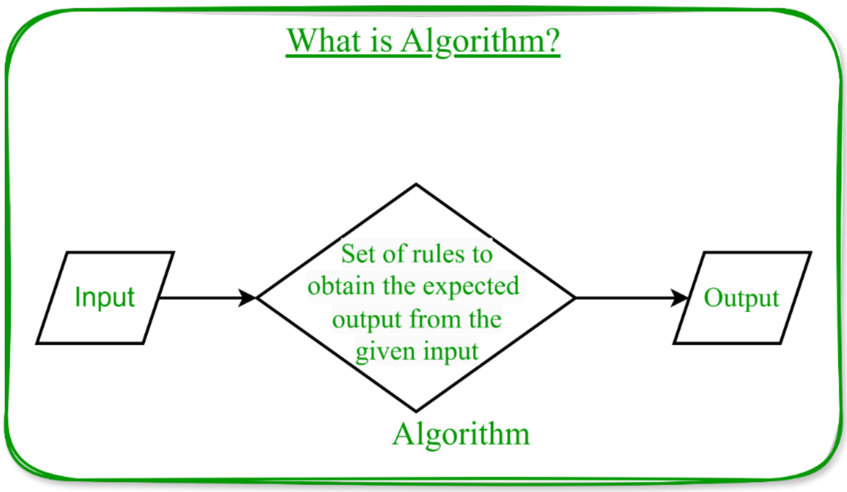

What is Algorithm?
Input
Set of rules to obtain the expected output from the given input
Output
Algorithm

Figure 4.2: Algorithm Description

Key Characteristics of Algorithms

1. **Input:** Is the data the user enters to initiate a process that will yield the output.
2. **Output:** Is the result or solution to the problem based on the given inputs.
3. **Definiteness:** each algorithm step must be precise, leaving no room for interpretation or uncertainty.
4. **Finiteness:** an algorithm must have a finite number of steps, meaning it should eventually terminate after a finite number of operations.
5. **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.
6. **Language independence**: algorithms must contain instructions that can be implemented in any suitable programming language, yet the output will be as expected.

Examples of Real-life Algorithms

1. An algorithm for a child getting ready for school could be:
   a. Wake up
   b. Brush teeth
   c. Wash
   d. Dress
   e. Eat breakfast
   f. Get school bag
   g. Travel to school

h. Arrive at school

i. Enter school premises

Note that an algorithm can be contained in another algorithm, for example brushing teeth is an algorithm within the algorithm of a child getting ready for school.

Also, for some children, the steps might be in a different order, like Step c (Wash) being switched with Step b (Brush teeth), or the formula might be more detailed, for example replacing Step a (Wake up) with additional steps such as Turn off alarm clock, Get out of bed and Make bed.

2. Another example of an algorithm could be for a child eating breakfast. This example involves an iteration (repetition)
   a. Put porridge in a bowl
   b. Add sugar and milk to the porridge
   c. Stir to mix in sugar and milk
   d. Spoon porridge into the mouth
   e. Repeat step 4 until all porridge is eaten
   f. Rinse bowl and spoon

Types of Algorithms

1. **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.

Activity 4.2

Work with your peers to perform this searching task

1. Write your names on the cards.
2. Mix your groups cards with another group in the class.
3. Choose a group leader to search for his/her members cards.
4. Observe the process the group leader used to look for the group's cards.
5. Discuss the process and write your findings in your book(s).

Congratulations, you have just completed a search algorithm.

2. **Sorting Algorithms:** These are algorithms that puts elements of a list into an order, for example, arranging the name cards in alphabetical order.

Activity 4.3

Work with your peers to perform this sorting task

1. Using the same cards from Activity 2.
2. Arrange the group's cards in alphabetical order.
3. Discuss how you arranged the cards in alphabetical order.

   Wow, you've got great sorting skills, the process of the activity you just completed is termed as 'sorting'.

3. **Encryption Algorithms:** These are algorithms that encode (hide) data to make it more secure when being stored or transmitted. For example, HTTPS websites that transmit credit card and bank account numbers encrypt (hide) this information to prevent identity theft and fraud. These websites will use encryption algorithms.
4. **Mathematical Algorithms:** These are algorithms that perform mathematical operations, such as finding the average of two numbers like activity 4 or as calculating the least common multiple (LCM) of two numbers. Figure 4.3 is another simple example of algorithm that adds two given numbers and output the sum. Figure 4.4 shows an implementation of this algorithm in Python.

Activity 4.4

Observe and complete the partially completed algorithm for finding the average of two numbers.

1. Let the first number be "8"
2. Let the second number be "7"
3. ....
4. ....
5. The result "7.5" is the average of the numbers

   The completed activity represents a mathematical algorithm.

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.

It is important to note that algorithms can be represented in different ways. In this content, the two ways we will focus on are pseudocode and flowcharts.

Pseudocode

The word **pseudo** means the opposite of something, therefore the word **pseudocode** in a computing environment simply refers to the use of plain text to solve a problem rather than using code. Pseudocode is an algorithm (a list of instructions) written in a plain language or text. These instructions need to be in the correct order to complete a process and the list is read from top to bottom. It is a good practice to number the instructions/steps as shown in the real-life example shown in Figure 4.3.

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 4.3 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 4.4 below.

Pseudocode	Code/program	Output

1. Assign *x* the value 432
2. Assign *y* the value 161
3. Assign *z* the sum of *x* and *y*.
4. Output the value of *z*
	x = 432
	y = 161
	z = x+y print(z)
		593

Figure 4.4: *Algorithm with program and output*

Another example of a simple algorithm written in pseudocode is given in Figure 4.5. 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):

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

Output:

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

Figure 4.5: A Simple algorithm written in pseudocode

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 (plain text) and not written in a programming language. As mentioned earlier, pseudocode can be translated into any suitable programming language. Figure 4:6 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 any other programming language.

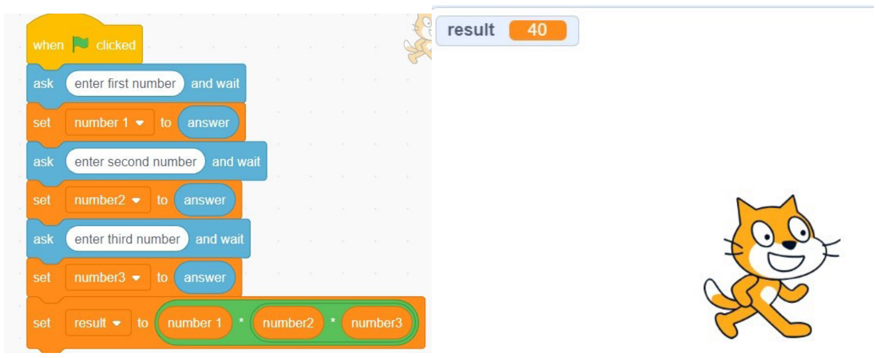

```
when [flag] clicked
ask (enter first number) and wait
set [number 1] to (answer)
ask (enter second number) and wait
set [number2] to (answer)
ask (enter third number) and wait
set [number3] to (answer)
set [result] to ((number 1) * ((number2) * (number3)))
```

Figure 4.6: *Scratch program with output*

Activity 4.5

1. As a group copy and complete the pseudocode for multiplying 37 and 70 in your books.

```
Step 1: number1 = 37
Step 2: number2 =
Step 3: result =
Step 4: print(result)
```

2. Write in your books a program to output the sum of four numbers entered by the user.
   Refer to Figure 4.5 as a guide.
3. Present your completed work to the class for discussion.

Activity 4.6

1. Produce the pseudocode for the following tasks in the correct order (printed by the teacher with lines, if applicable) or with the labelled numbers and share with your colleagues.
   a. Calculate the circumference of a circle with radius 5cm.

1	2	3	4
Print (circumference)	circumference = 2 * Pi * radius	Pi = 3.14159	radius = 5

b. In Ghana, you are eligible to vote when you are 18 years.

1	2	3
if age = < 18	age = Enter your age:	else

5	4
output(age, "You are not eligible to vote.")	output(age, "You are eligible to vote.")

Congrats on completing these pseudocode activities, next you will learn about flowcharts.

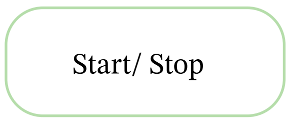
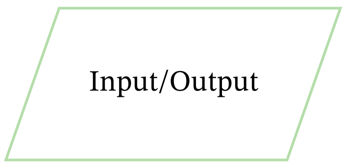
FLOWCHARTS

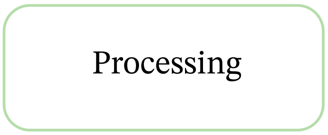
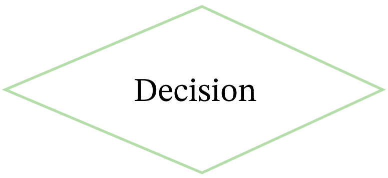
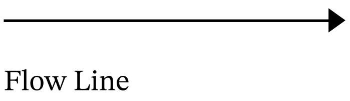
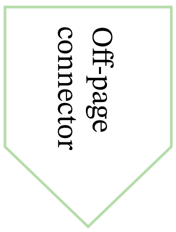
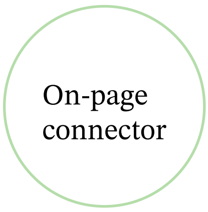
While pseudocode uses plain text or language to show the step-by-step approach of solving a problem, flowcharts also use shapes to solve the same or similar problem. So, we shall now consider what a flowchart is.

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, you can visually understand the logic and structure of an algorithm.

Basic Symbols Used in Flowchart Designs

Table 4.1: Symbols and their functions

Functions	Symbols
1. Terminator: The terminator symbol shows where a process begins or ends.	
2. Input/Output: A parallelogram shape is used to show any action where information is either taken in (input) or given out (output) in a flowchart.	

Functions	Symbols
3. Processing: Imagine a box that does math operations like addition, subtraction, division and multiplication. Numbers go into the box through one line (incoming flowline), and the result comes out through another line (outgoing flowline).	
4. Decision: A diamond shape shows a choice. It has one line for incoming info and two lines for different outcomes based on the decision.	
5. Flow lines: Arrows show the order of instructions and how different parts of a flowchart are connected.	
6. Off-page connector: A symbol with a letter inside shows that the flow continues on another page. Look for the matching letter on that page to follow the process.	
7. On-page connector: A symbol that links different parts of the flowchart on the same page, with one incoming flowline.	

Here are examples of how a flowchart is used when solving problems

1. This flowchart shows the process of admitting a candidate to No. 1 High School, depending on their score. The candidate is admitted to No. 1 High School if their total score is greater than 800. They're not admitted to No. 1 High School if their total score is less than 800.

Flowchart:

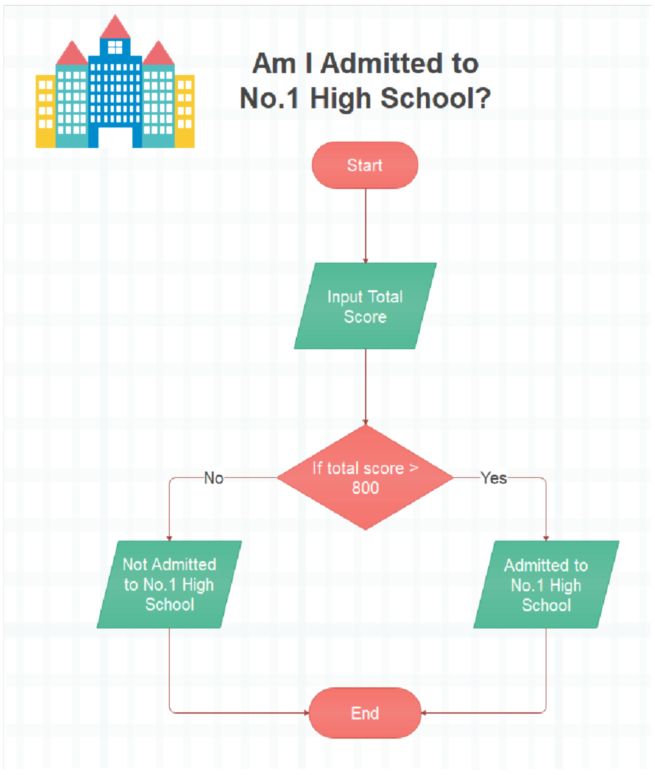
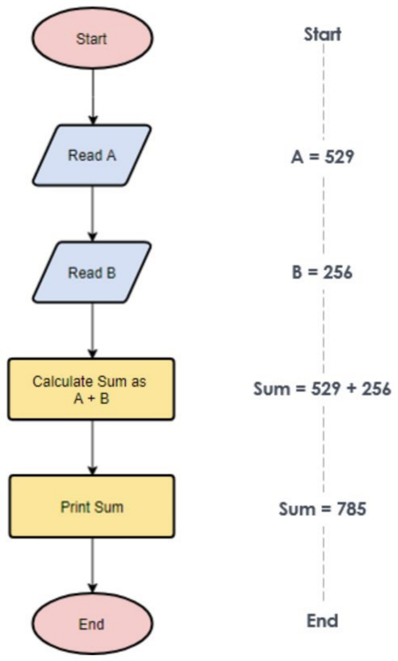

Am I Admitted to No.1 High School?
Start
Input Total Score
If total score > 800
No
Yes
Not Admitted to No.1 High School
Admitted to No.1 High School
End

Figure 4.7: Flowchart showing admission process

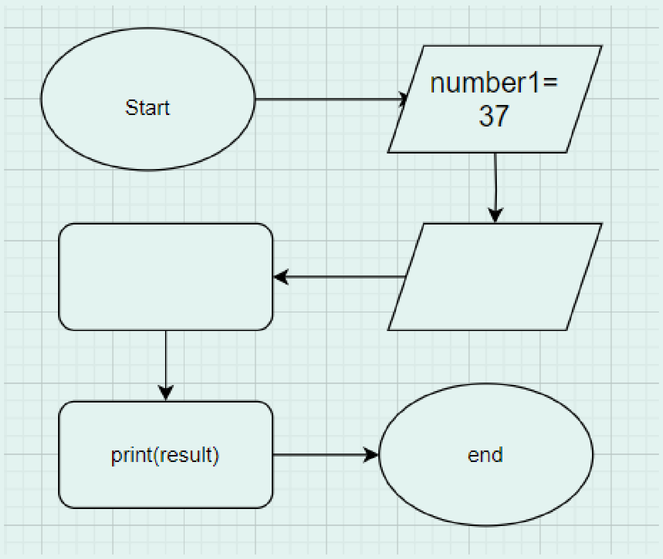
Figure 4.7 is a real-life example showing if a candidate can be admitted to No. 1 High School based on the condition that the candidate's total score should be greater than 800.

2. Finding the sum of 529 and 256 using flowchart

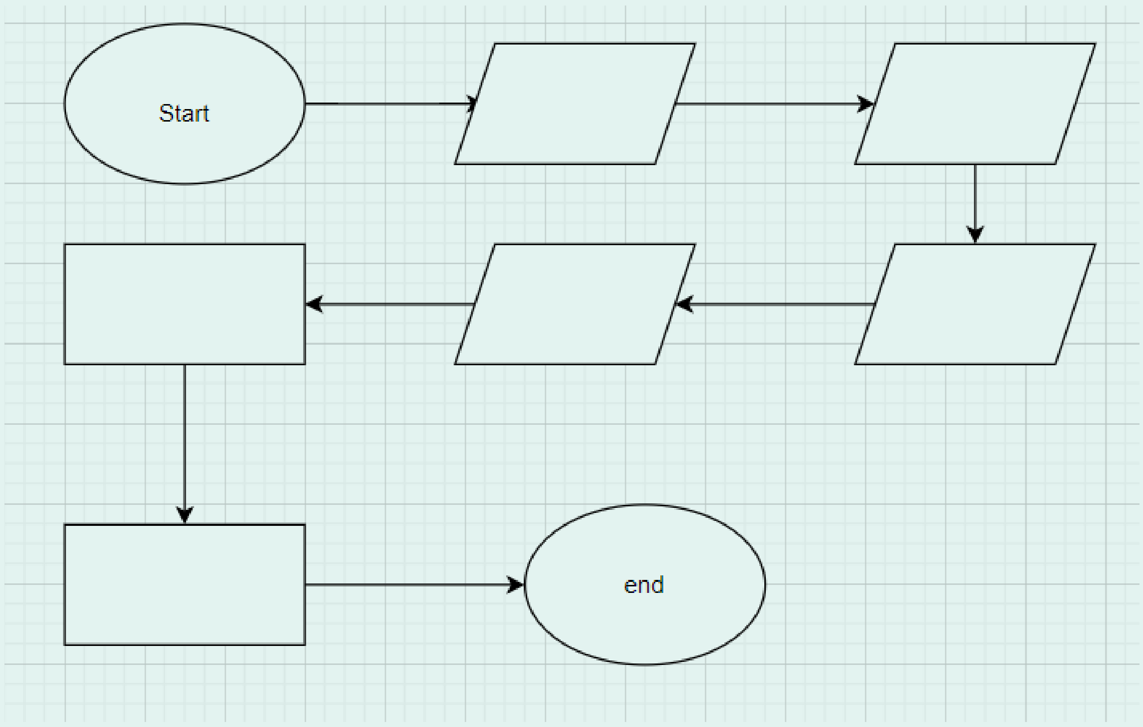


Activity 4.7

1. Refer to Activity 4.5 (item 1), to complete the flowchart below as a group in your books



2. In our book, copy and complete the following flowchart for writing a program which outputs the sum of four numbers entered by the user.



3. Present your completed work to the class for discussion.

Activity 4.8

1. Individually, write pseudocode, and draw a flowchart of a game where a user enters a letter which is compared to a selected list of letters to see if it exists.

   The user enters a letter as an input, if the letter is found in the selected list of letters, the system gives a message as 'Congratulations' and if it is not found among the selected letters, a message is given as 'Sorry, the letter is not what we are looking for'.
2. Compare your results with your peers.

PROGRAM DEVELOPMENT CYCLE/PROGRAM DEVELOPMENT LIFE CYCLE

This is also commonly known as software development cycle/ software development life cycle. When creating new software or programs, a software development team usually follows a process called the Software Development Life Cycle (SDLC). The SDLC is a step-by-step method used to design, develop, and test software. Its goal is to create high-quality software that meets the client's needs, at the lowest cost, and in the shortest time possible.

The process starts when a client, who is the person or organization needing the software, asks the development team to create a program. The client provides a detailed description of what they need, and the team then follow the SDLC phases, which typically include Analysis, Design, Coding (Implementation), Testing, and Maintenance (see Figure 4.9). These phases guide the development process from start to finish.

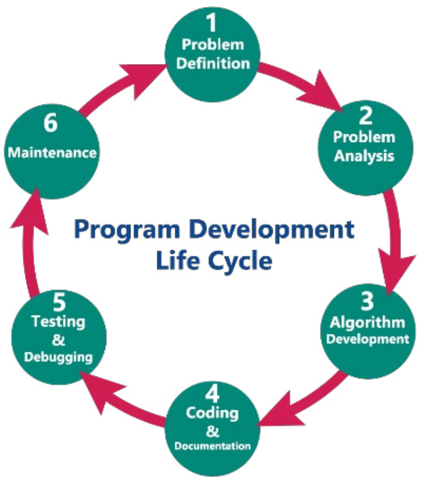


Figure 4.9: SDLC

Activity 4.9

In your groups, do the following activities:

1. Individually, rearrange the stages of the SDLC in the correct order by writing on the dotted lines and compare your answer with your peers.
   - a. Implementation
   - b. Analysis
   - c. Testing
   - d. Design
   - e. Maintenance

   ……………., ……………., ……………., ……………. and ………….
2. Develop a mnemonic for the stages of the SDLC and share with the whole class.

   Hint: **Mnemonic** refers to a technique or device designed to help remember information more easily

   E.g. **A D**ance **I**n **T**he **M**oonlight.

Problem Analysis

Problem Analysis is the first step in developing a software program. During this phase, a team member, usually called a systems analyst, talks to the client to understand what the program is supposed to do. They discuss the purpose of the program, which means they figure out what the program will be used for. They also identify what the program needs, such as what data will be input, what processes will be done, and what outputs are expected. Additionally, any guesses or assumptions made about how the program will work are clearly stated during this stage.

Example 1

Problem specification: 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.

Table 4.2: *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.

Table 4.3: *Functional requirements*

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

Assumptions

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

Design

In the design phase, the information gathered during the analysis phase, such as the purpose of the software, the main requirements and any assumptions, is used to start planning how the software will be built. A key part of this planning is creating an algorithm, which is a step-by-step way to solve the problem. You might remember learning about different ways to show algorithms, like pseudocode and flowcharts.

During this phase, programmers also make a list of the variables and data types needed and plan how the user interface (UI) will look.

The user interface is the part of the software that users interact with, such as the screens where they enter data or see results. When designing these interfaces, programmers often use wireframes, which are simple sketches that show how the UI will be laid out.

Wireframes can be drawn by hand on paper or drawn using a computer, as shown in Figure 4.10.

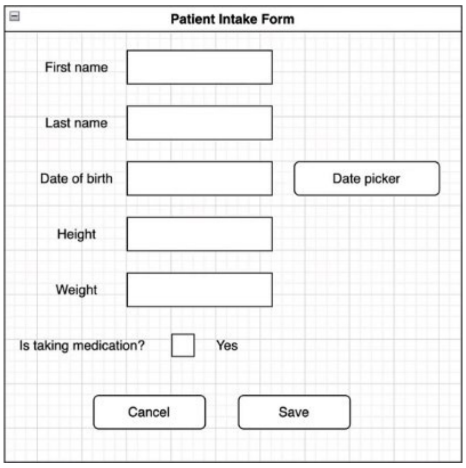


Figure 4.10 Wireframe for an input screen

Activity 4.10

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, design the UI of the program.

Share and discuss your design with the class.

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 in the spelling or grammar used when coding) is shown in Figure 4:11, where the command word '*print*' is incorrectly entered as '*pront*' so the program will not run correctly.

```
messagel = "Believe you can and you're halfway there." print (messagel)
print("\n") #prints a blank line in Python
message2 = "Have a good day!" print (message2)
```

```
Believe you can and you're
halfway there. Have a good
day!
```

syntax error

```
messagel = "Believe you can and you're halfway there." pront (messagel)
print("\n") #prints a blank line in Python
message2 = "Have a good day!"
print (message2)
```

```
Traceback (most recent call last):
   File "./prog.py", line 2, in <module>
NameError: name 'pront' is not defined
```

Figure 4.11: *A correct program and the program with a syntax error, each with corresponding outputs*

Testing

Testing is an important part of making sure a program works correctly. During the Testing stage of the Software Development Life Cycle (SDLC), you check if the program gives the right answers by trying it out with different sets of data.

Testing should be well-planned, and you should keep track of what happens during each test. It's also important to test the program in different ways, using normal data (what you expect), exceptional data (unusual or unexpected data), and extreme data (very high or very low values).For example, if you're testing a program that counts how many students passed a test where a pass is 50% or higher, you might use test scores like 56, 78, 47, 90, -82, 79, 58, 60, 50, and 77. This includes different types of data to see how the program handles them.

Normal data refers to data that the program is designed to accept as valid input, such as the numbers 56, 78, 47, 90, 79, 58, and 77 in the example. When the program processes this data, it should produce the correct output, like how a score of 90 should register as a pass.

Extreme data is data that lies on the boundary of what is considered normal, like the number 50 in the example. A result of 0% or 100% would also be considered extreme data because they are at the limits of possible outcomes.

Exceptional data is data that is out of range or invalid, like the number -82 or a test result written as "forty" instead of a number. This type of data is not expected to be accepted by the program.

In large programs, such as games, bugs can exist because it may not be possible to test every possible input or action that a future user might make. To test a program effectively, you should try to manually figure out what the correct output should be before running the program. It is also helpful to have someone other than the person who wrote the program test it, as they may spot mistakes that the original coder missed.

Aside from syntax errors (which are mistakes in the code itself), other errors that can occur during testing are **run-time errors** and **logic errors**. A run-time error happens when a program tries to do something it can't, like dividing by zero, which can cause the program to crash. A logic error is a mistake in how the program is designed, such as using the wrong formula to calculate something, like the circumference of a circle.

After testing and before the software is deployed (making the software available to the client), installation instructions, user guides, and training materials are created to help the client use the software effectively. This is part of a phase called **Documentation**. After Documentation, there is often an **Evaluation** phase, where the program is reviewed to make sure it meets the original problem's requirements.

Maintenance

After the program is delivered and the client starts using it, the maintenance phase begins. Maintenance includes fixing any issues that arise during regular use, improving the program's design to make it better, or making adjustments for different situations, like creating a version that works in another country or on a different operating system.

Activity 4.11

1. In your groups, write a program to calculate and output the area of a circle using a radius value in centimeters entered by the user and taking pi as 3.14 as shown in table 1 (note that ** in Python means exponent in mathematics)
2. Test the program with different numbers (e.g., 4, 12.5, -6 and 0.0014) to test the program as shown in table below
3. Write your observation in the comment if the programming is correct or wrong.
4. Correct the program (debug), if needed and test the program again with your own numbers or with the numbers above.
5. Discuss with your peers why a program needs testing and maintenance.

Table 4.4: Template

	INPUT	PROCESSES	OUTPUT	Comment
E.g.	Enter radius = 5cm	Area = pi *radius * 2 = 3.14 * 5*2	Area = 31.4	
1				
2				
3				
4				

Earlier in this course, you were introduced to data and the basic concept of data type as used in a computing environment. I hope you can recall that data was explained as the raw fact that can be fed into the computer, which means that a data is not informative until it is processed or organised. You will now explore data structures and data types, but before that try your hands on this activity, based on your previous experience:

Activity 4.12

Study the statement below and answer the questions which follows

10 x 5 = 50

1. Identify the data or data part in the statement above.
2. Where in the above statement will the "processing" be seen or identified?
3. Identify the information (output) in the above statement.
4. Compare your answers with your peers.

DATA TYPES

In computer science, a data type is a label or an element that tells a computer how to understand and handle a piece of data. 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.

Common examples of data type in programming 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.

Note that data types can vary from one programming language to another.

Variables

In the previous lesson, we touched on what a variable is. A variable was explained as a named location in memory that stores a value of a specific data type. On the other hand, 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 (such as x and y) or a more descriptive name (such as age, carName or total_volume).

Rules for Naming Python Variables

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

Data Types in Python

There are different data types depending on the programming language you are using. The data types in Python are shown in Figure 4:12. Integer and real values are specified by *int* and *float* respectively. A string variable will 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.

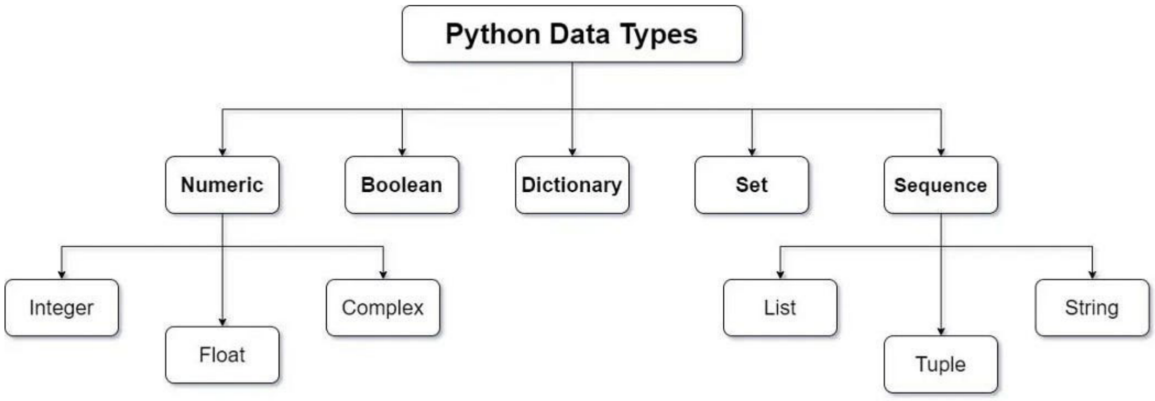


Figure 4.12: Python Data Types

In some programming languages, 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 Table 4.5, using a variable called 'x'.

Table 4.5: Data Types in Python

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. This means that you can change the type of a variable's value to another type by using specific functions known as type conversion functions.

The point here is that "conversion functions" allow you to convert data from one type to another. For example, you can convert a number stored as a string to an integer or change a floating-point number to an integer.

Example 3

int("123") converts the string "123" to the integer 123.

Table 4.6: Data Types in Python

Line of code	Data type
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

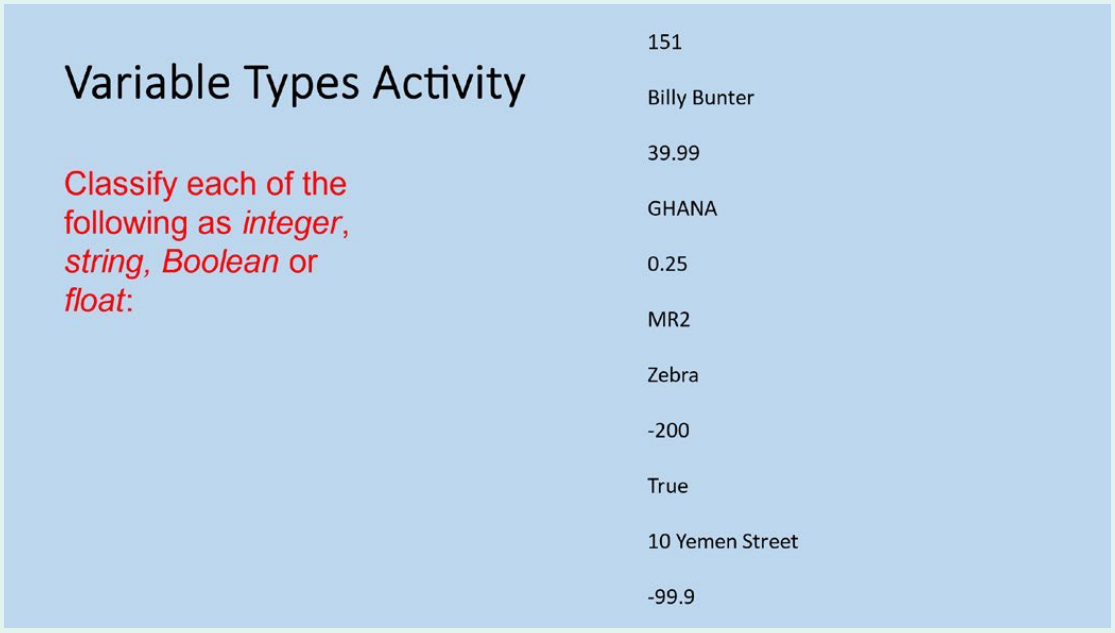
Mathematical operations will not work as expected– see Figure: 4.13 below

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 4.13 will aid understanding how basic type conversion and the type() function works in Python.

Activity 4.13

1. Individually, write in your books, the data type of each item in the list on the slide below.



Variable Types Activity

Classify each of the following as *integer*, *string*, *Boolean* or *float*:

151

Billy Bunter

39.99

GHANA

0.25

MR2

Zebra

-200

True

10 Yemen Street

-99.9

2. Share your finding with your peer or teacher.

3. Justify your answer to your peer or teacher and take feedback.

Activity 4.14

1. Individually, choose a suitable variable name to store the data item in Python using the same list of data items in **Activity 4.13**.
2. Compare your answers with a peer and talk about the reasons for your choices.

Python program	Output
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))	9 <class 'int'> 45 <class 'str'> 34.86 <class 'float'> 34

Figure 4.13: Storing the data item in Python

Activity 4.15

1. Study the following code and write your prediction on what the output could be.

Code			Prediction
1.	w = False	print (type (w))	
2.	x = 1	print (type (x))	
3.	y = 2.8	print (type (y))	
4.	z = "hello"	print (type (z))	

Table 4.6: Coding

2. Run the code in a python environment to check your predictions.
3. If the output of your code is "error" or "syntax error", seek assistance from your peer whose code runs.

Data Structures

The term “structure” refers to the collection of two or more systems working together as a whole. For example, the body system such as digestive system, reproductive system, excretory system, circulatory system and the rest put together is called the body structure.

Though 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.

Importance in learning 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

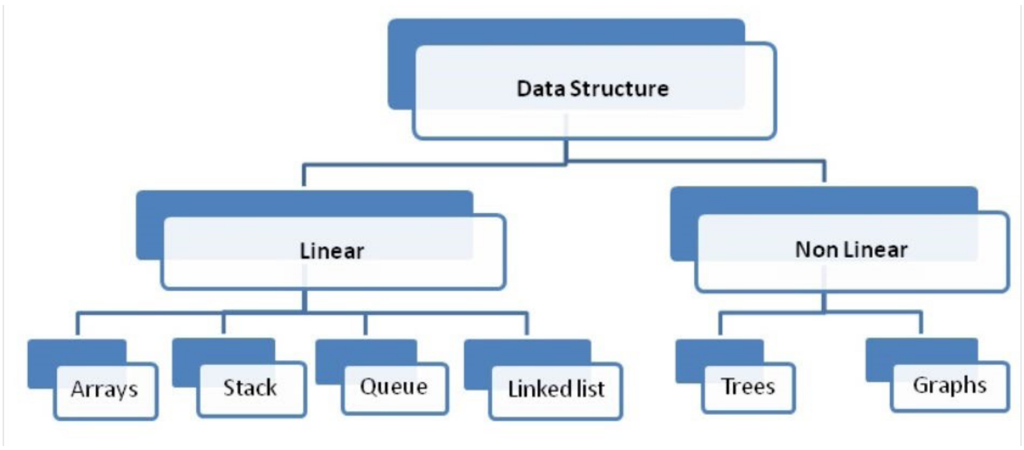


Figure 4.14: *Classifications of data structures*

Linear data structure

A data structure that preserves an end-to-end or straight connection among its data elements is known as a linear data structure. The arrangement of the data is made on a straight path, 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 structures 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 but in a real-life situation, you can consider the images below as linear structures since queues are typical examples of linear data structures.

Figure 4.15: Real-life examples of linear data structure

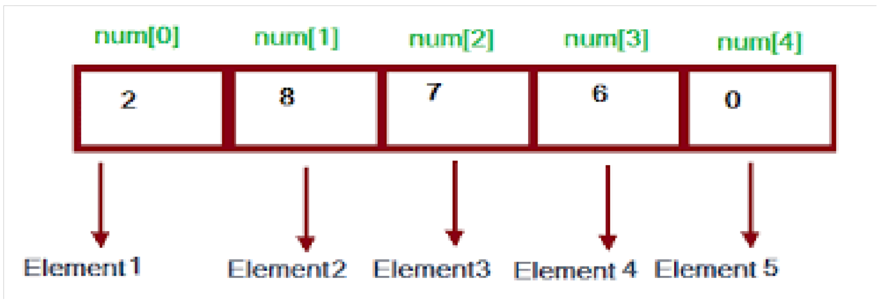


Figure 4.16: An array named num(). An array is a linear data structure

Non-linear data structure

Non-linear simply means that the items or elements are not arranged or organised in an end-to-end structure. In computing, non-linear data structures are those where the data elements are not organised sequentially, but rather, in an 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 (on the left) and graphs (on the right).

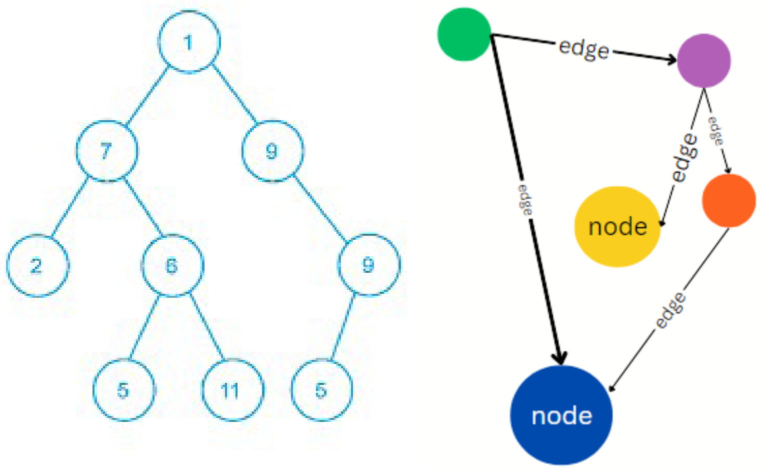


Figure 4.16: Examples of non-linear data structures

Activity 4.16

1. The image below shows a non-linear data structure of the administration hierarchy of Kumasi Wesley Girls High School. Study the diagram and expand the following headings to their respective next two levels.

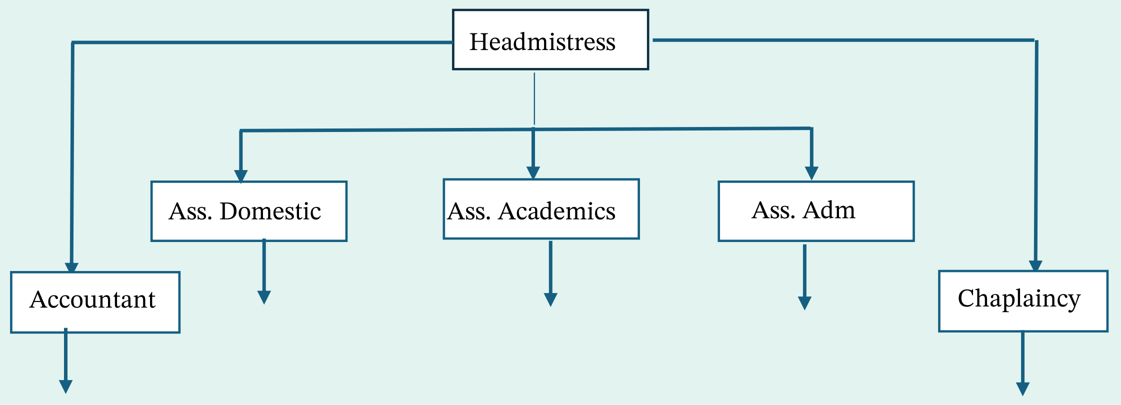


Figure 4.17: Non-linear data structure of the administration hierarchy

Using your experiences acquired on non-linear data structures, identify a real-life situation such as the hierarchy of students' administration or the leadership in your current school, draw a non-linear data structure of it.

Difference Between Linear and Non-Linear Data Structures

Table 4.7: Difference Between Linear and Non-Linear Data Structures

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.
Implementation is Easy.	Implementation is Difficult.
Examples: Array, Stack, Queue, Linked List.	Examples: Tree, Graph.

Static and Dynamic Data Structures

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

1. ***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.
2. ***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 4.15
5. **Deletion:** Removing the element from the data structure.

Python program	Output

num = [2,8,7,6,0]
print('The third element in the array is:')
print(num[2])
print() # this print a blank line
num[2] = 12 # the update is happening here at index 2
print('The third element in the array is:')
print(num[2])
	The third element in the array is: 7
	The third element in the
	array is: I
	12

Figure 4.18: 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 4:15), 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 more detail in the next content.

Activity 4.17

1. Work together in groups and study the Table 4.8.

Table 4.8: Linear and non-linear data structures

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

2. Complete the table on linear and non-linear data structures using the given criteria.
3. Compare your answer with other group and justify your results

Conclusion

Variables, data types, and data structures are three important interconnected concepts in programming. Variables are containers for data, and their data type defines 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.

As we move on in this course, we are building on data structures. Looking at Arrays and building on from the linear data structures that we looked at in the previous learning. We will be looking at the types of arrays and their applications in programming.

To picture how arrays work in programming, imagine you are at the Makola or Kejetia market with a list of ingredients for your favourite Ghanaian dishes—akple, fufu, kenkey, Red Red, Tuo Zaafi and more. Instead of memorising the list in your head, you neatly arrange the items on a piece of paper in rows, making it easy to find, add, or swap items. That is exactly what an array does in programming: It keeps your data organised and easy to manage, just like your market list.

This learning will help you have knowledge, understanding and application of arrays in programming languages.

Activity 4.18

1. You have ten minutes to suggest at least five lists of items of the same type that a computer program may need to store individually.
   E.g., a list of scores obtained by forty-five (45) students in an exam.
   a.
   b.
   c.
   d.
   e.
   f.
2. *Share and compare your result with your peers.*
   Note: *Since the items in the list are of the same type and stored in a single line, we collectively refer to them as an array.*

ARRAYS

Arrays are like organised lists (just like you listed in Activity 4.18) where you can store multiple items of the same type, all in a row in the computer's memory. Instead of using different names for each item, you use one name and can quickly find any item by its position (index) in the list.

In an array, all the data elements share the same name, but each one has a unique number called a subscript. This number helps you find a specific item in the array by using the array's name along with the subscript. One important thing to know about arrays is that the data is stored in connected (contiguous) memory spaces (locations), which makes it easy to move (traverse) through the items using their index numbers.

Array variables offer a simpler way to handle multiple values without needing to create separate variables for each one. For instance, instead of using six different variables to store the numbers 2, 6, 11, 7, 18, and 4, you can use a single array variable, as shown in Figure 4.19, to store all these values together.

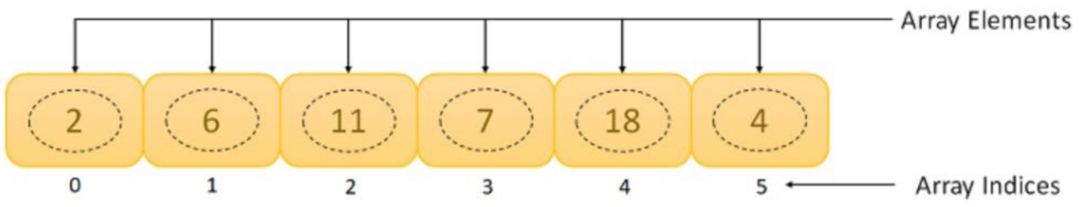


Figure 4.19 An array of integers

If this array was implemented in Python using the array variable name "numbers", the code in Figure 4.20 could be used. Print statements are included to show how array indexing works.

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

print(numbers)

\n

print(numbers[0])

print(numbers[5])
```

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

2

4
```

Figure 4.20: Numbers program, version 1- sets up an array of six integers and outputs the array and two elements

Other examples of how arrays can be used instead of declaring multiple variables include:

Instead of having separate variables for each student's score such as
score1 = 75, score2 = 88, score3 = 92, etc.

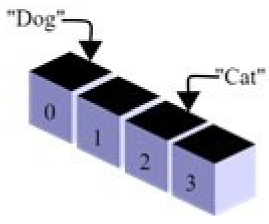
You can use a single array called scores to store all the values together
scores = [75, 88, 92, 67, 84]. This is an integer array.

Also, storing Names of Friends can be done as friends = ["Ama", "Kwame", "Kojo", "Akua", "Yaw"] instead of having individual variables for each friend's name,

like friend1 = "Ama", friend2 = "Kwame", etc. This variable is a string.

Another example of an array containing string elements is shown in Figure 4.21. In this array, two of the elements are empty strings. The second print statement includes text to label the output. In Python, the '#' symbol is used to write comments. If you want to include notes or explanations in your code that should not be executed, you can start the line with a '#' 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
```

Figure 4.21: Array

Arrays are often used in different situations to organise and store lists of related information that are all the same type. For example, imagine you have a shopping list with items like tomatoes, rice, and fish (these are all strings). Instead of using separate variables for each item, you can store them all in one array. Similarly, you can use an array to keep track of the names of students in your class (also strings), the number of goals your school's football team scored in 12 matches (integers), the yes or no answers from 20 people in a survey (booleans), or the money collected by a charity in five different churches (floats). Arrays help you organise this data in a way that is easy to manage and use.

Activity 4.19

Together with your peers, do the following task in your books:

1. Describe at least five (5) real-life situations that can be considered as arrays.
2. What variable will be suitable to store each of the arrays mentioned above? Justify your answer.
3. Share your responses with your peers.

Let's look at **Array Size and Indexing**: In most programming languages, the size of an array declared as arrayName(size) indicates the total number of elements it can hold, directly corresponding to the number specified. For example, if you declare score(9), it means the array score can hold exactly 9 items, indexed from 0 to 8 in languages like Python, C++, and Java.

For example, in Figure 4.17 and Figure 4.18, the arrays for numbers and animals could be referred to as numbers(6) and animals(4).

Activity 4.21

1. Individually, find out the number of items these arrays can hold
   a. Grade[23]
   b. Food[19]
   c. Score[54355]
   d. Houses[6, 4]
   e. Teachers[1021, 13, 40, 45, 60]
2. Justify your findings with a reason
3. Compare your answers to your peers.

Why Use Arrays?

Arrays are one of the simplest and most widely used data structures in programming. Since arrays are so fundamental to organising and managing data, they are usually the first data structure that you learn when you start studying programming.

Using an array can make a program simpler by allowing you to store related data under one name. This means you can write code to search or sort through all the data in the array quickly, instead of writing separate lines of code for each individual item. This not only makes the program shorter but also makes it easier to find and fix any errors (debug).

Figure 4.22 shows an alternative program to the program shown in Figure 4.21 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.

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

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

Figure 4.22: Numbers program, version2– uses six different variables to store six integers

Activity 4.21

Using your response to Activity 4.18, suggest a suitable data type and array variable name for use in a Python program.

*E.g.: The type and suitable namee for the example given in Activity 1 could be **float** and **exam_score (44).*** Think about your responses and how they could be improved if needed.

Arrays and Memory [more advanced optional content]

To understand how arrays work in memory, think of an array as a row of rooms in a building. Each room (or element) has a specific room number (address number or memory address). For example, if the first room's address is 1000, the next one might be 1001, and so on. So, the fifth room would be at 1004, and the tenth would be1009.

Similarly, the elements of an array are stored in consecutive memory locations. If the first element of an integer array is at address 1024, and each integer takes up 4 bytes, the second address will be 1028, the fifth element would be at 1024 + 4*4 = 1040.

This means that if you know the memory address of the first element, you can easily find the address of any other element in the array. This efficient arrangement allows for quick access to data in the array (see Figure 4.23 below).

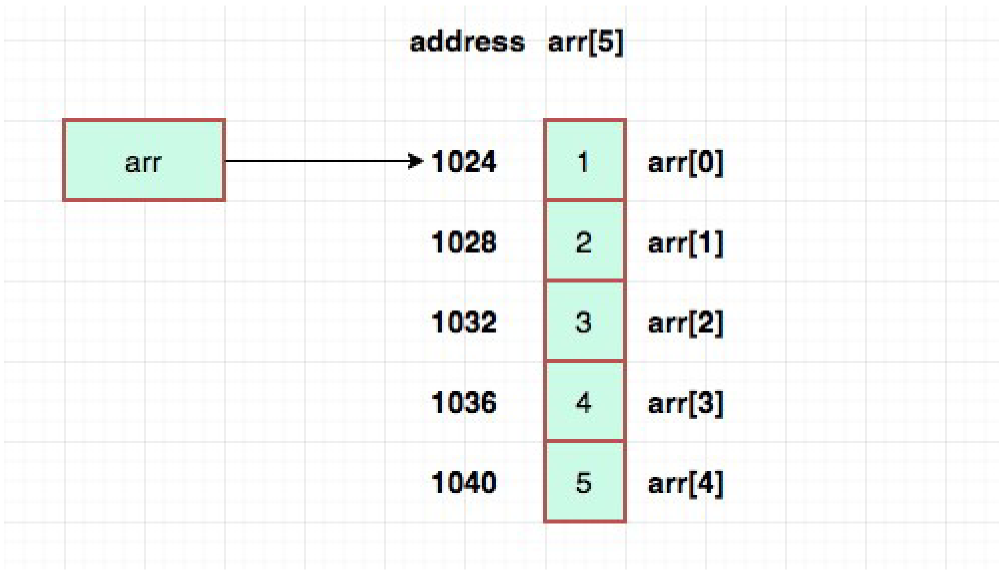


Figure 4.23: An integer array variable called "arr" with five elements.

Understanding One-Dimensional and Two-Dimensional Arrays

Arrays are available in various types, the most common being one-dimensional and multi-dimensional arrays. Multi-dimensional arrays are basically arrays within arrays and can have several levels. However, in this content, we will concentrate on one-dimensional array (1D) and two-dimensional (2D) arrays.

1. **One-Dimensional Array**: This is the type of array we have discussed so far. It consists of a single row of data elements stored in a sequence of memory locations. When we refer to an "array" without specifying, it typically means a one-dimensional (1D) array.

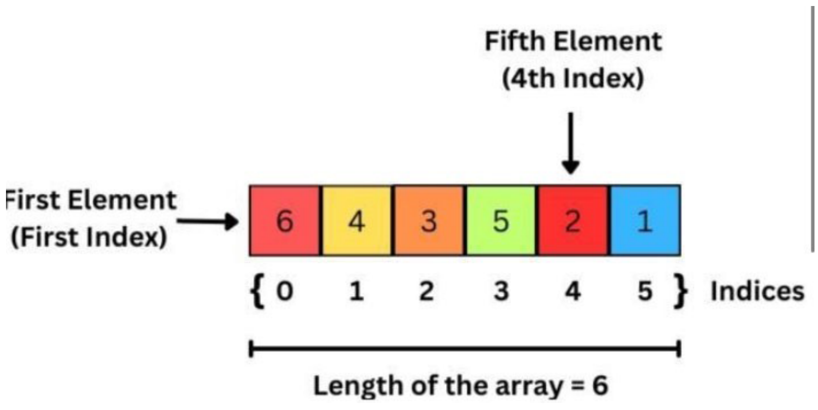


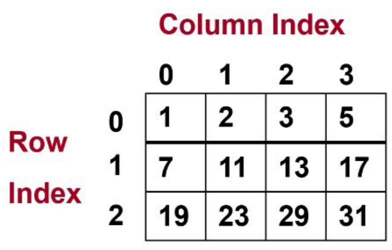
Figure 4.24: A one-dimensional array of integers

Parallel arrays use multiple arrays to represent related data, where each index in one array corresponds to the same index in the other arrays. For example, if you have one array for names and another for ages, the elements at the same index in both arrays represent the same person. So, if `names[5]` is "RaF" and `ages[5]` is 12, this means RaF is the sixth person and she is 12 years old, as illustrated in Figure 4.25. This approach allows you to store different types of related data using arrays.

names	Daneil	Eric	Oyampo	Millicent	Alberta	RaF
ages	12	13	12	14	13	12

Figure 4.25: A set of two parallel arrays

2. **Two-dimensional array**: This type of array has multiple rows and columns, forming a grid-like structure. For example, a two-dimensional array with three rows and four columns, as shown in Figure 4.26, can be visualised as a table. Each element in the array is accessed using two indices: one for the row and one for the column, typically denoted as (r, c). The value at a specific position is determined by these two indices, where "r" represents the row number and "c" represents the column number.



Column Index

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

Figure 4.26: Rows and Columns

In reading the table or 2D array, you read the row then the column. For example, (Row, column) is ...

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

A 2D array offers a more efficient way to manage related data compared to parallel arrays, but all elements must be of the same type.

For example, if a school wants to store the marks of five students (Daniel, Kojo, Beatrice, Nii, and Nana) across four subjects (Computing, Science, Mathematics, and English), you will use a 2D array. This array would have four rows (one for each subject) and five columns (one for each student), organising the marks in a tabular format. Each cell in this table represents the mark a student received in a particular subject.

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

Figure 4.27

Activity 4.22

1. Categorise the list of problem specifications as either 1D or 2D.
   a. Modelling a game of tic tac toe.
   b. Processing a set of test marks registered by a class to find out the number of passes.
   c. Tracking the monthly rainfall in different regions across the country.
   d. Storing the daily temperatures recorded at various weather stations across Ghana.
   e. Storing the names of students in a class along with their corresponding scores in three different subjects.
   f. Representing the seating arrangement of students during entertainment in the assembly hall, with rows and columns.
   g. Tracking the weekly attendance of students in different classes in the school.
   h. Recording the sales of different food sold by multiple vendors at the school market.
   i. Recording the inventory of different items in multiple malls in Ghana.
   j. Storing the scores of players in different rounds of the All-African games competition held in Ghana.
2. Write two (2) differences between 1D and 2D arrays

3. Compare your categorised list and difference(s) to your colleague(s) for justification and classification.

Implementation (Application) of a 2D Array in Python

In Python, you can store data in a grid or table format using something called a 2D array. This is just a list of lists, where each inner list represents a row in the table.

Example:

Suppose we want to implement the array in Figure 4.27 using Python, we can write the codes as shown in Figure 4.28.

```
marks = [
[10, 55, 90, 70, 60],  # Marks for the first student
[50, 60, 75, 65, 95],  # Marks for the second student
[95, 75, 55, 85, 45],  # Marks for the third student
[35, 95, 65, 55, 75]   # Marks for the fourth student
]
```

Figure 4.28 Array Implementation in Python

After writing the code, we noticed the total marks for Beatrice in Science is to change from 75 to 92.

Instead of rewriting the marks code all over, we can just update Beatrice's marks. Let's do this as an activity together.

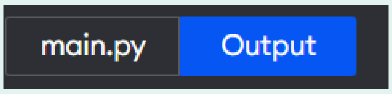
Activity 4.23

1. Using a Python environment, copy the code in Figure 4.28. (Optional: Click "here" to access a sample environment).
2. Identity the index of Beatrice's Science score.
3. Write the Python program for the update using this guide "marks[row_index][column_index] = 92".
4. Run the code
5. Confirm the update by writing the code "print([row_index][column_index])"
6. Alternatively, confirm the update by writing the code "print(marks).

Notice for using the environment

a. Click on Run/play icon to run the code

b. Navigate the typing area and output area by click on these icons respectively.



Activity 4.24

Complete this list by adding onto the applications of 2D arrays. Ensure to add at least three items with explanation and examples.

1. **Representing Data in Tables**: Just like the marks sheet your teacher uses to record scores in different subjects, you can use a 2D array to represent this data. For example, a 2D array can store the marks of students across various subjects, like a spreadsheet or table (as shown in Figures 4.25 and 4.26).
2. **Game Boards**: Think of a game like Ludo or Tic-Tac-Toe. The board can be represented using a 2D array, where each cell corresponds to a specific position on the board. For example, a chessboard, which is 8x8, can be modelled using a 2D array.
3. **Computer Graphics and Images**: Imagine you are editing a photo on your computer. Each tiny square (pixel) in the image can be stored in a 2D array, with the array holding the colour information for each pixel. Image processing, such as applying filters or adjusting brightness, often involves manipulating these pixel values stored in 2D arrays.
4.
5.
6.

Advantages and Disadvantages of Arrays

The advantages and disadvantages of arrays are discussed below:

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 is smaller than required.

Activity 4.25

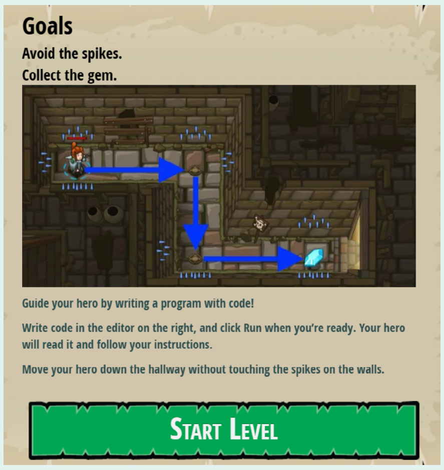
Write at least three additional advantages and three disadvantages of the array data structure and present your findings to class orally.

Have you ever played a game that connects you from one level to another level as you win?

If your answer is **No** then do this activity for 15 minutes

Activity 4.26

1. Click this link to play code combat on the internet (ensure you have access to the internet before playing).
2. Click on start level after the game is loaded as shown in Figure 4.29.



Goals

Avoid the spikes.

Collect the gem.

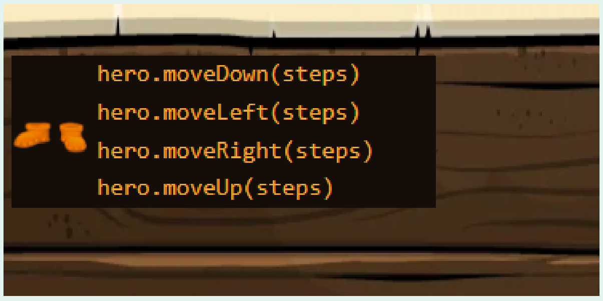
Guide your hero by writing a program with code!

Write code in the editor on the right, and click Run when you're ready. Your hero will read it and follow your instructions.

Move your hero down the hallway without touching the spikes on the walls.

Figure 4.29

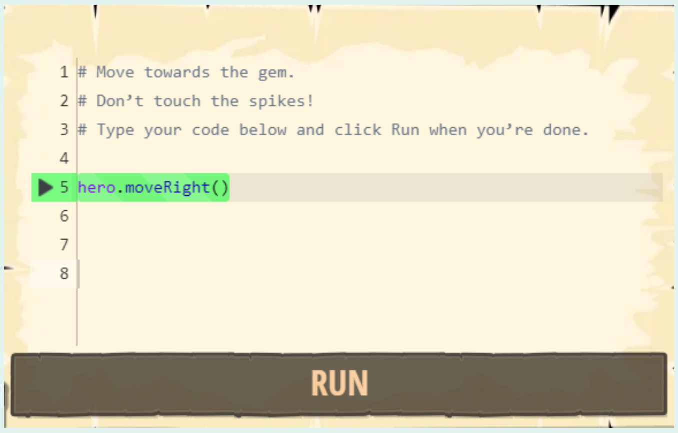
3. Use the same codes as seen in the image below to instruct your hero avatar to move in the directions and pick up the gem.



```
hero.moveDown(steps)
hero.moveLeft(steps)
hero.moveRight(steps)
hero.moveUp(steps)
```

Figure 4.30

4. Type your code in the coding environment as shown in the screenshot



```
# Move towards the gem.
# Don’t touch the spikes!
# Type your code below and click Run when you’re done.

hero.moveRight()
```

Figure 4.31

5. Click Run to test your code
6. If successful, the avatar will move to pick up the gem
7. Click “done” then “continue” to move to the next level.
8. Play only for 15 minutes.

Wow congratulation, you have done a lot in this activity, you coded in Python, and you also learnt more about arrays.

LINKED LISTS – FEATURES, OPERATIONS, EXAMPLES, TYPES, APPLICATIONS, ADVANTAGES & DISADVANTAGES

A linked list is similar to how you moved the hero avatar from one point to the other until you have picked up the gem. Again, it is like moving from level 1 to level 2 to level 3, etc.

We can define a **linked list** as a type of data structure that stores items where each item, or 'node', contains two parts. The data itself, and a reference, often called a 'pointer', that points to the next item in the list. An example is shown in Figure 4.32 below. Note that the arrow to the next node is the pointer.

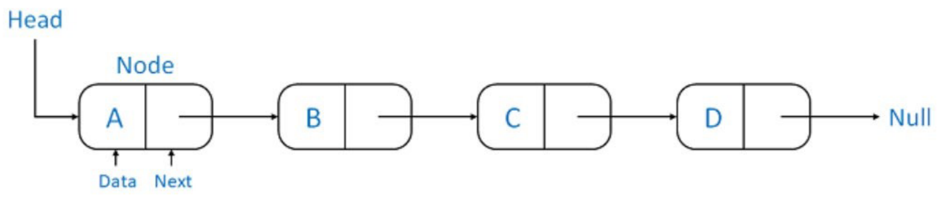


Figure 4.32: Linked list

A linked list is not fixed like an array, the size of a linked list can be changed anytime, making it dynamic.

How are Nodes Added and Deleted

In Figure 4.31, to delete node B from the list, the pointer from node A would need to be changed to point to node C. In this case, the memory that node B used would be marked as open and free space.

Inserting a node P between C and D would be achieved by changing the link of the pointer from C to point to the memory location of P and then linking the pointer from P to the memory location of D.

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.

Activity 4.27

1. Use Figure 4.32 as a guide to draw illustrations for both the deletion and addition of nodes in the linked list.
2. Show your illustrations to your peers for improvement if needed.

Types of Linked Lists

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

1. **Singly Linked List:** Figure 4.32 and the correct illustrations of Activity 4.26 are examples of singly linked lists. Singly linked lists are uni-directional (one directional). They can only point to the next node in the list, not the previous one.
2. **Doubly Linked List:** A doubly linked list is a more advanced version of the singly linked list. Here, each node, contains three parts: the data itself, a pointer that points to the next item in the list, and another pointer that points to the previous item. These doubly linked lists can be traversed both **forwards and backwards.**

Activity 4.28

1. Based on your understanding of singly and doubly linked list, draw two illustrations to represent them.

   Use Activity 4.26 as a guide to help you.
2. Observe your illustrations and give 1 advantage and 1 disadvantage between singly and doubly linked list.

In addition to the advantage and disadvantage identified in Activity 4.27, doubly linked list has the advantage of allowing elements to be passed through in two ways. While singly linked lists allow elements in only one way. One disadvantages of doubly linked lists include greater memory requirements (two pointers rather than one pointer per node) and more code is required for implementation, while a singly linked list needs lesser memory and less code for implementation.

Activity 4.29

With your peers, list at least four additional applications of linked lists.

1. For GPS guidance systems, linked lists can be used to store and organise a list of places and routes that make it easy for users to get where they want to go.
2. Because a web browser's search results are linked together as a linked list, you can access the previous and next URL by using the back and next buttons.

3. The songs in the media player app are connected to the songs that came before and after them; using a doubly linked list.
4. ....
5. ....
6. ....
7. ....
8. ....

Let us now discuss the advantages and disadvantages of Linked Lists in general.

Advantages of Linked Lists

1. Dynamic Size: A linked list can change its size during use, growing or shrinking as needed to fit the data it holds. This means it adapts to what you need at any moment.
2. Flexibility: You can change the order of items in a linked list without moving the actual data. Instead, you just change the links that connect the items.
3. Efficient Use of Memory: Linked lists use memory more effectively than arrays. They only need enough memory for the number of items they actually hold, not for the maximum number they might hold.

Disadvantages of Linked Lists

1. Accessing Items: Unlike arrays, where you can quickly find an item by its position (or 'index'), in a linked list, you must start at the beginning and follow the links to find a specific item. This can take more time, especially if the list is long.

Now that your interest is aroused, let us study another data structure known as **Stacks**.

STACKS – FEATURES, OPERATIONS, EXAMPLES, TYPES, APPLICATIONS, ADVANTAGES & DISADVANTAGES

A stack is a type of list where you can only add or take away items from the top. This is called LIFO, which stands for Last In, First Out. Imagine it is like a stack of books, as shown in Figure 4.33; the last book you put (push) on the top of the pile is the first one you take off (pop).

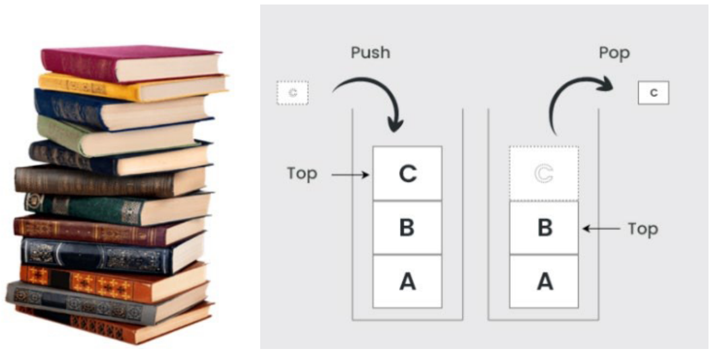


Figure 4.33: Stack

Primary Operations in Stacks

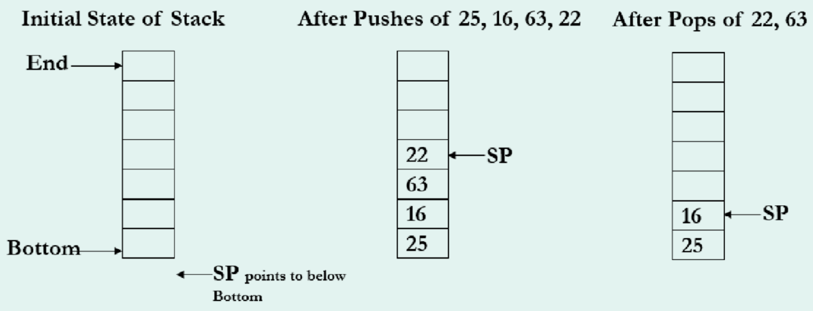
1. **Push**: the operation to insert (add) a new element to the stack.
2. **Pop**: the operation to remove or delete elements from the stack.

   These two operations are shown in Figure 4.33.

Activity 4.30

Let us explore by observing the process of how stacks work in the illustration below. Note that "SP" means Stack Pointer.

Answer the following questions based on your observation.



1. Identify how many elements are in the initial state of the stack.

2. Write your thought on why the SP at the initial state is below the bottom.
3. Use the appropriate terminology to explain what happened between the middle diagram and the last diagram on the right.

Good work finishing the activity, it is important to know that stacks can be made using different types of memory like:

1. **Contiguous Memory**: This is like an array where all the items are stored side by side.
2. **Non-Contiguous Memory**: This uses a linked list where each item can be stored in a different place, but each one points to the next.

Again, as new data is added to the stack, it cannot go past the End pointer. If it does, there is a 'Stack Overflow' and the program ends. Also, data must not be popped (removed) below the Bottom pointer. If it does, there is a 'Stack Underflow' and the program ends.

Activity 4.31

With your peers, list at least four additional applications of stacks

1. To reverse the order of a list of items – see Figure 4.34.
2. For the Undo and Redo functions in an edit.
3. 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.
4.
5.
6.
7.

The following images show the use cases of stacks

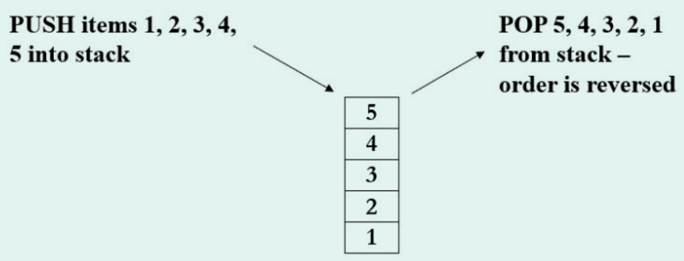

PUSH items 1, 2, 3, 4, 5 into stack
POP 5, 4, 3, 2, 1 from stack – order is reversed
5
4
3
2
1

Figure 4.34: Reverse the order of a list of items

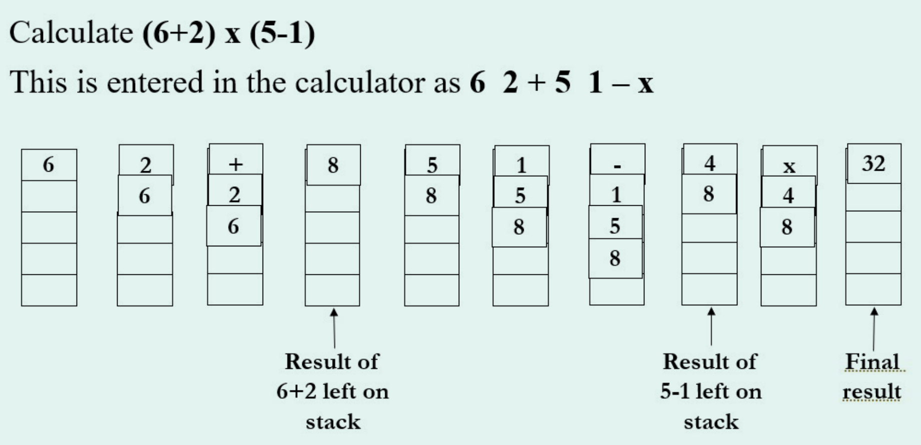

Calculate (6+2) x (5-1)
This is entered in the calculator as 6 2 + 5 1 – x
6
2 6
+ 2 6
8
5 8
1 5 8
- 1 5 8
4 8
x 4 8
32
Result of 6+2 left on stack
Result of 5-1 left on stack
Final result

Figure 4.35: How some calculators make calculations

Activity 4.32

With your peers, list at least two additional advantages and disadvantages of stacks

Advantages of stacks

1. Stacks are a simple data structure that is easy to understand, which means they can be used in many situations.
2. Stacks use the LIFO principle to make sure that the first thing taken off the stack is the last thing added to it. There are many times when this behaviour is important, like when a program calls a function.
3.
4.

Disadvantages of stacks

1. You can only get to the elements at the top of a stack, which makes it hard to get to or change elements in the middle of the stack.
2. If more elements are pushed onto a stack than it can hold, an overflow error will occur, resulting in a loss of data.
3.
4.

Activity 4.33

Together with your colleagues:

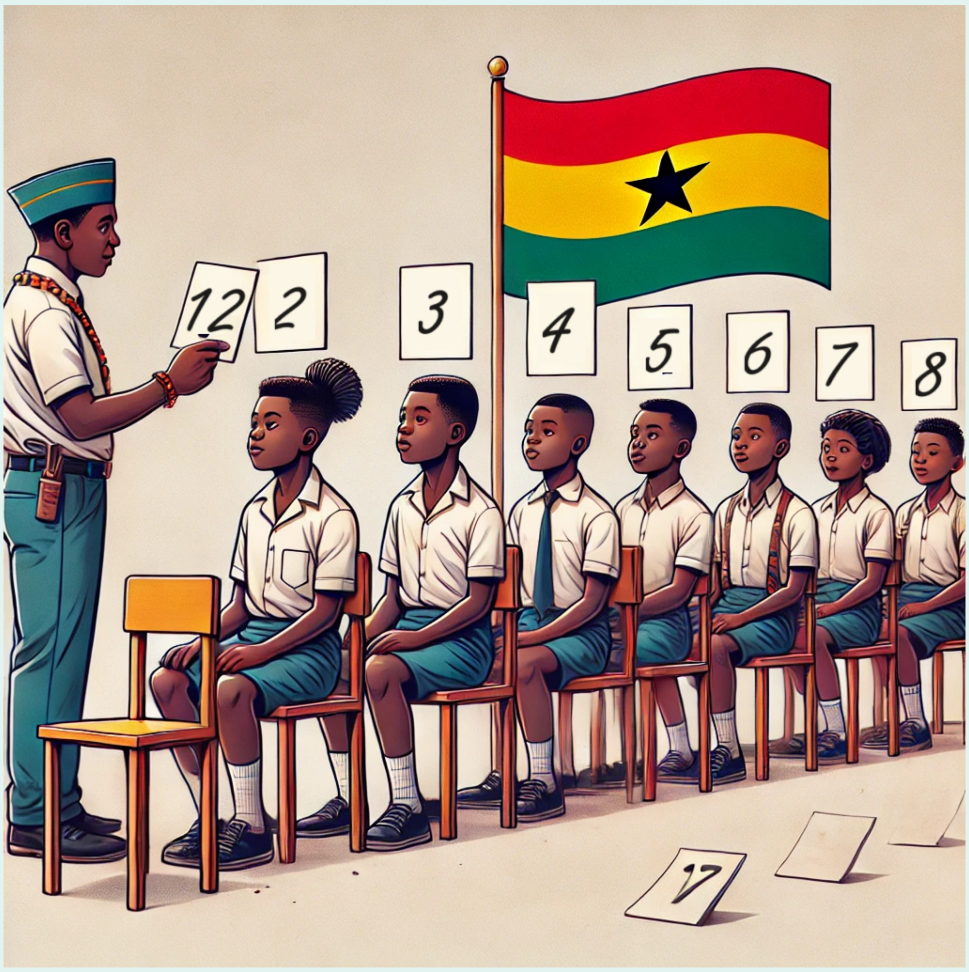


Figure 4.36

1. Identify the student who was removed from the line
2. Identify which student is to be removed next according to their seating numbers?
3. According to the seating numbers which student is supposed to be added next at the end of the line
4. Discuss with your colleague what national value is violated by the teacher by calling the 12th card (representing a student) to join the end of the queue.
5. What will be your actions taking into consideration national values.

Congrats, you have just learnt about linked lists and stacks.

QUEUES – FEATURES OF A LINEAR QUEUE, OPERATIONS, EXAMPLES, APPLICATIONS, ADVANTAGES

What is a Queue?

Queue simply means being in a line, just like how the papers were in line waiting to get ink printed on them. In computing, data or tasks line up to be processed. A queue is this line of items waiting to be handled.

Primary Operations of the Queue

1. **Enqueuing**: Is the process of adding an item to the end of the queue. It's just like when a new person comes and stands at the back of the line. In computing, when you add a new data item to a queue, you are enqueuing it.
2. **Dequeuing** is the process of removing an item from the front of the queue. Think of it as the person at the front of the line getting their ticket and leaving the queue. In computing, dequeuing means taking the first item off the queue to be processed or used.

The second and third items are the primary operations of queuing.

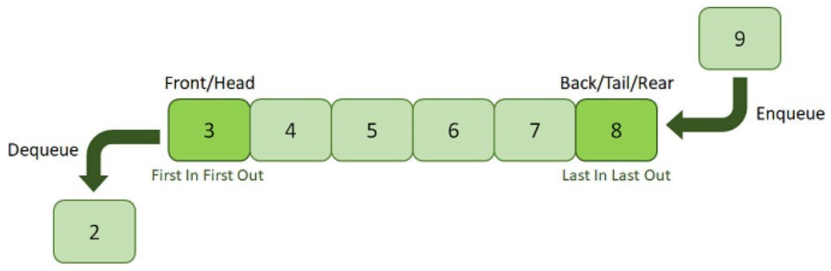


Figure 4.37: An examples of a queue

A queue is similar to a stack because both involve adding and removing elements. However, in a queue, you add elements at one end and remove them from the other end. This method is called FIFO, which stands for First In, First Out. It means the first item that goes into the queue is the first one to come out.

We also need to know that in computing, queues can be implemented using **arrays** or **linked lists, both of which you** have studied earlier in the course.

A real-life example of a queue is a line at a voting booth or a ticket counter as shown in images below.

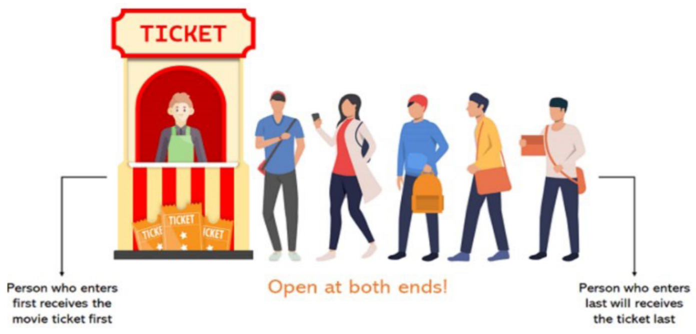

TICKET
Person who enters first receives the movie ticket first
Open at both ends!
Person who enters last will receives the ticket last

Figure 4.38: A real-life example of queues

Activity 4.34

1. In five minutes, write a list of at least ten (10) real-life examples of queues, together with your colleagues.
2. Explain with reasons, why any of your listed examples in (a) is a queue but not a stack.

Applications of Queues

In addition to your list, some applications of queues are:

1. Printer Queues: Printers use a system called queues to manage print jobs. When you send a document to be printed, it joins the end of this line, and the printer tackles each job sequentially.
2. Task Scheduling: Queues are employed to organise tasks by order or priority. For example, a task management system might use a queue to ensure that high-priority tasks are addressed promptly.
3. Operating Systems: Operating systems utilise queues to keep track of tasks that need processing. This helps manage the computer's resources effectively, such as distributing CPU time among various programmes.
4. Traffic Management: Just as with managing road traffic, systems like airport controls use queues to regulate the flow of vehicles and people, ensuring smooth and safe movement.
5. Network Protocols: Protocols like TCP employ queues to sort data packets for transmission over the internet. This ensures that messages are sent and received in the correct order.
6. Computer Memory: Some types of computer memory use queues to order the steps required to execute programs. This process starts with fetching an instruction, then decoding, and finally executing it. New instructions join the back of the queue to await their turn.

Well done, we have covered content on Queues and Linear Data Structures. Let's now look at what non-linear data structures are and how they are used in computing.

Non-Linear Data Structures

Non-linear data structures arrange data in ways that resemble branches of a tree or connections in a network. This means that each piece of data can be linked to many other pieces, not just the one before or the one after. To access different data items, you can take different routes or follow various links between them. This kind of structure helps when the relationships between data are not simple and straightforward.

Activity 4.35

1. Watch this video (click on the word "video" to access it) to deepen your understanding of non-linear data structures. Do not worry if you do not understand the calculation, it will be covered in detail in year 2.
2. Draw a diagram illustrating what a tree and graph are in your books.
3. Write beneath the diagrams a brief note of what trees and graphs are.
4. Write a brief key difference between graphs and trees.

Graphs

A graph is a collection of nodes with data elements that are connected to each other nodes by lines (edges). The nodes can also be referred to as vertices.

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.

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 4.39

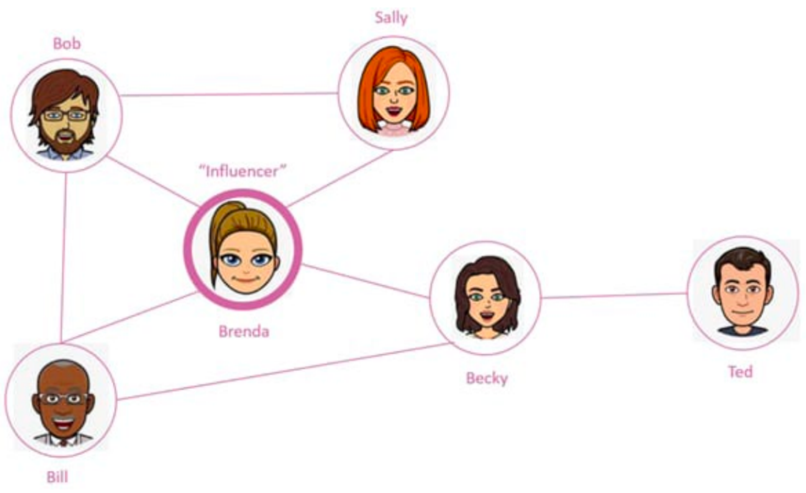


Figure 4.39: Social Media Graph

Now that we have studied both linear and non-linear data structures, let's compare them.

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.

1. 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.
2. Non-linear data structures may have faster access time but require more memory to store pointers or references.
3. 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.
4. 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.

Activity 4.36

1. In your books, draw two large bubbles or boxes. Label one "Linear Data Structures" and the other "Non-linear Data Structures."
2. From the "Linear Data Structures" bubble, draw lines to make smaller bubbles or boxes and label them with examples like "Arrays," "Queues," and "Stacks."

3. From the “Non-linear Data Structures” bubble, do the same for examples like “Trees” and “Graphs.”
4. Next to each example, write a short description or key characteristic of them.
5. Write at least two key differences and similarities between linear and non-linear data structures below your drawing.
6. Once you have completed your concept map, show it to your teacher for approval.
7. After your teacher checks your map, discuss it with your classmates. Share what you have learnt and see what interesting points others have in their maps.

Beautiful, you have successfully compared the linear and non-linear data structures.

In addition to what you drawn and written, it is important to note that when choosing between linear and non-linear data structures, consider factors such as memory usage, access time, insertion and deletion, and search and sort capabilities.

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 they require more memory to store pointers or references.

Hello learner, as we progress with SDLC, in this learning, our focus will be on its last stage, the implementation stage/phase of the cycle. It involves planning and implementing the algorithm, flowchart and pseudocode as a program, using programming languages such as Python, etc. Get ready for such fun!

PROGRAMMING BASICS USING PYTHON

An introduction to Python

There are many programming languages, like Python, Java, C#, and Swift. Python is very popular and can be used for many different things. It is called a **high-level programming language**, which means its code looks a lot like the English we speak, making it easier to understand and learn, meaning beginners can pick it up quickly.

What can Python do?

Python is used for many things, including software development, web development, making games, analysing data, controlling robots, mathematics modelling and more. It is a very powerful programming language, which is why big companies like Google, Dropbox, Spotify, and Netflix use it.

For example, the Dropbox desktop app is written entirely in Python, showing how Python can work on different types of computers and operating systems. Its ability to run on various platforms makes it a popular choice for many businesses.

Steps for Getting Started with Python

It is time for us to see the Python console (also known as the Python shell) so that we can try out some basic Python code. The Python console is a good place to experiment with small code snippets. There are some basic steps we need to follow before we can write Python programs. These include the following:

1. **Select an Integrated Development Environment (IDE) for writing and running your Python code**
   a. An IDE (integrated development environment) is a piece of 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.
   b. Popular choices of IDE's for Python include Integrated Development and Learning Environment (IDLE), Spyder, PyCharm, Visual Studio Code, and Jupyter Notebook.
   c. At this level, the basic Python IDE called IDLE will be enough. Download the latest version of IDLE that is compatible with your computer's operating system at https://www.python.org/downloads/
   d. To create and test Python programs on an iOS or Android, there are a good number of Python editor apps available from the App Store or Play Store, including Python Editor (free) - see, Juno (free), Pythonista, and Pyto, Python Code-Pad-Compiler&IDE, etc.

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

Python Editor App 4+
Create and Run your Python 3
Bloop Software
Designed for iPad
#7 in Developer Tools
★★★★★ 4.6 • 778 Ratings
Free

Figure 4.40: Python Editor App

   e. Replit (https://replit.com/languages/python3) 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**

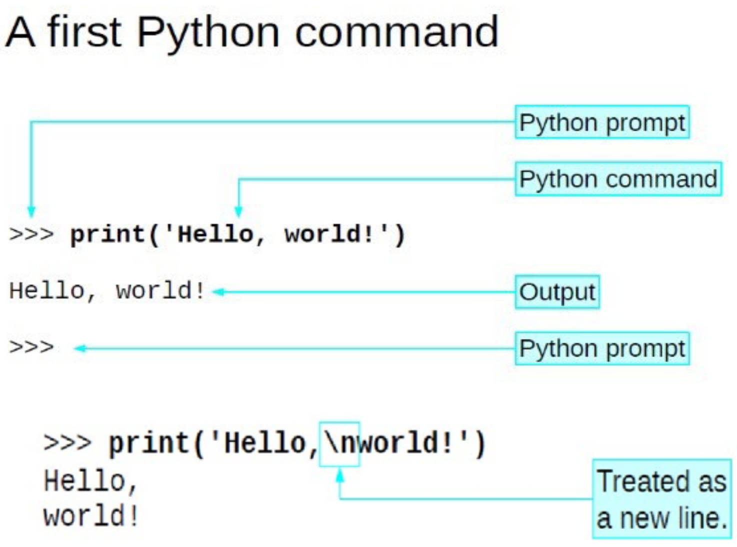
   At this stage, you will need to create a folder to store your Python programs. When saved, Python files will have the ***.py*** extension.

3. **Start coding!**
   a. **<u>print () function</u>** is an in-built function in Python to output (display) specific messages to the screen.

      To print a blank line, use the following command(s):

      print () *or* print("\n").

      An example of print is shown below.



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

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

Activity 4.37

1. Individually, download your chosen Python IDE to your device and install it using the steps above.
2. Create a folder location where you would like to keep your python programs (applicable to laptops).
3. Create and open a new Python project
4. Copy the following python code into your Python IDE and see the output:
   a. print("My name is ....") # Type your name in the dotted space.
   b. print("I am .... years old") # Type your age in the dotted space.
   5. Now, write your own python code to display your aggregate and the name of your basic school.
   6. Share your result to your peers.

Congrats, you have just completed your first python coding activity.

b. Assignment operator

The assignment operator in Python is the "=" symbol.

For example,

- i. age = 16.
- ii. reply = True.
- iii. cost = 34.56.
- iv. city = "Kumasi".
- v. console = 'PlayStation'.

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

See Figure 4:41 on how the assignment operator is used in labelling in Python.

```
message = "Python is fun"

print(message)

information = " My message to you is "

print(information + message) #we use the + symbol to concatenate (join) variables

teacher_prize = "Eric Asomani Asante"

print(teacher_prize, "won the 2020 Ghana Teacher Prize Award ")
```

Output

```
Python is fun

My message to you is Python is fun

Eric Asomani Asante won the 2020 Ghana
Teacher Prize Award
```

Figure 4.41: Labelling in Python

Activity 4.38

1. Write a single line of code in python for each of the following:
   - a. Rate to 11.75.
   - b. Pi to 3.14.
   - c. Area of triangle to ½ x base x height.
   - d. Area of rectangle to length x breath.

2. Use the print function to output your code in (a).
3. Discuss the output with your peers for justification or correction.

c. Arithmetic operators

The equality comparison is defined in Python with a *double*

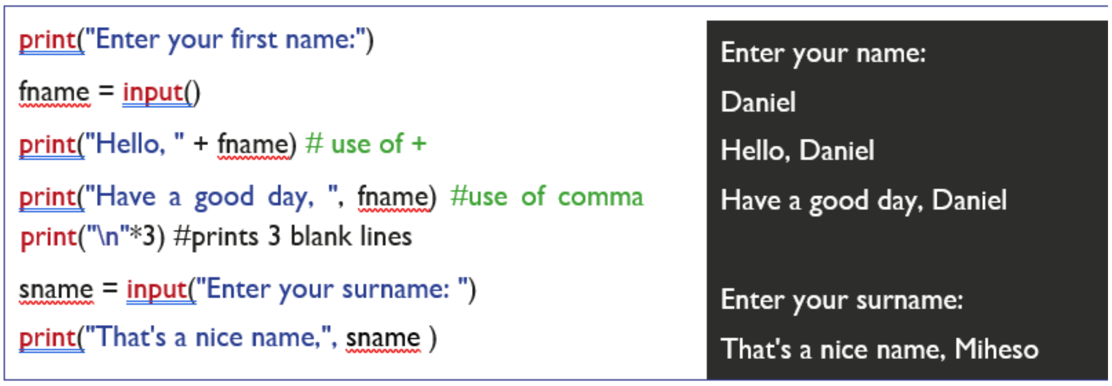
Table 4.9: Arithmetic operators

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

equals sign, "==". The sign is doubled to distinguish comparison from assignment**.**

d. <u>input</u> () function accepts user input. It allows you to enter your data.

Study the code in Figure 4.42 to understand how this function works. This code was written in Replit (at www.replit.com).



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

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

Enter your surname:
That's a nice name, Miheso
```

Figure 4.42: using input function in python

The above code exemplars should enable you to now code simple sequence programs, that is, programs that consist of a simple input, a process, and the output steps. We will now move on to focus on the implementation of algorithms, involving simple sequences using the Python programming language.

Activity 4.39

1. Correct the errors in the following lines of code. These errors could be syntax or logical. Two examples are:
   a. size = input(“Enter your shoe size)
   b. average = (number1 + number2)/3
2. Correct the errors in the following Python programs.

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

3. Enter your corrected code in Python to check it works.
4. Show the results to your facilitator and peers for review.

Activity 4.40

1. Follow the below steps to write a program that asks the user for their name and then greets them.
   a. Use the input() function to prompt the user for their name.
   b. Store the input in a variable called name.
   c. Use print() to display a greeting that includes the user’s name.
2. Follow these steps to create a program that asks the user for their favourite movie and why they like it.
   a. Ask the user for the name of their favourite movie.
   b. Ask why they like that movie.
   c. Combine and display the responses in a complete sentence.

Note an alternative for writing this activity using f-string in the print function is as follows:

```
movie = input(“What is your favourite movie? “)
reason = input(“Why do you like it? “)
print(f”Your favourite movie is {movie} because {reason}.”)
```

The f-string (formatted string literal) is a way to embed variables and expressions inside of a string. In Python, you create an f-string by placing the letter f before the opening quotation mark of the string and the variables are wrapped in braces { } within the statement to be output.

Input-Process-Output Model

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 a program can be coded, 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. Inputs are the information you get to use to solve the problem. Processes are the steps needed to get the output results from the input stage. Outputs are the goal of the problem solution.

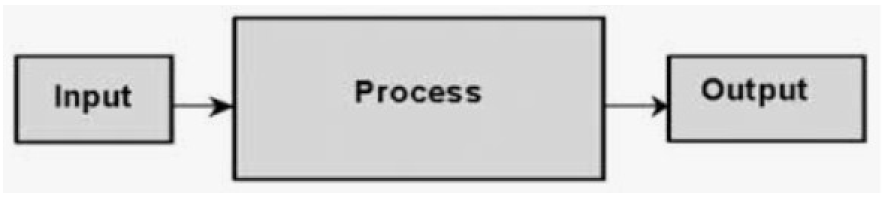


Figure 4.43: Input-process-output model

Activity 4.41

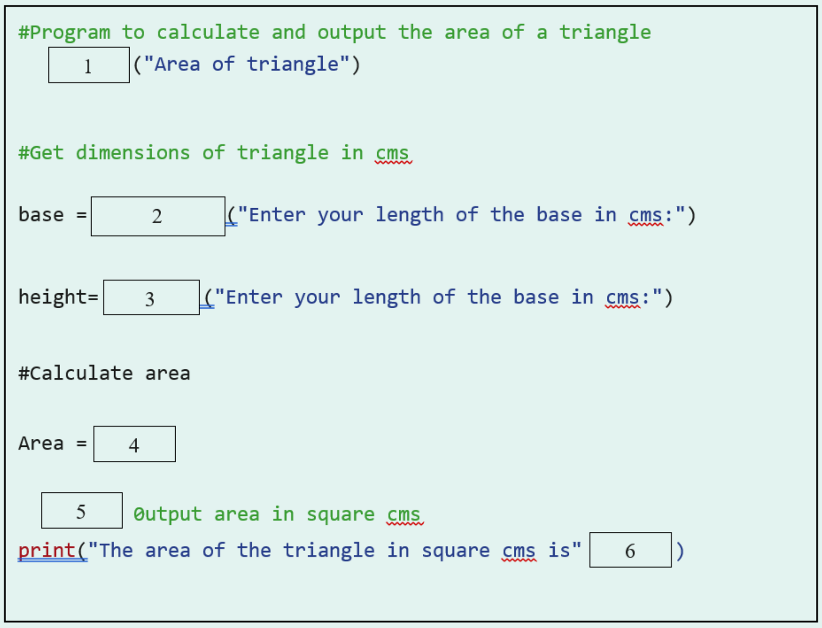
1. Write code to output 'Happy Birthday' three times, separating each Happy Birthday message with a blank line.
2. Write two similar pieces of code to output repeated messages of your own choosing, as practiced in (a).
3. Show the result to your peers and discuss.

Activity 4.42

1. Using the set of given algorithms, complete the partially completed corresponding programs.

   Pseudocode to calculate the area of a triangle

   a. Enter length of base in cms.
   b. Enter perpendicular height in cms.
   c. Area of triangle = (length of base x perpendicular height)/2.
   d. Output area of triangle in square cms.



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


#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] )
```

2. Show the result to your teacher and peers for review.

Activity 4.43

1. For each example, rearrange the lines of code to match the given pseudocode. Note that the operator to represent exponent in Python is **, so print(10 ** 3) would output 1000.

   Pseudocode

   a. Let x equal 2
   b. Let p equal 5
   c. Calculate the value of x^p
   d. Output the value of x^p

Table 4.10: *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	

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

Table 4.11: Re-arranged code

Code	Lines of pseudocode	Order

x = 2
 	
p = 5
	Let p equal 5	

answer = x**p	Output the value of x^p	
print(“The “, answer is “, answer)	Calculate the value of x^p	
	Let x equal 2	

3. Copy and complete the program below

```
# 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 [                              ]
```

4. Show the result to your facilitator and peers for review.

Algorithms Implementation

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

The step-by-step guide in implementing an algorithm in Python is discussed below.

1. **Understanding the algorithm:**
   Make sure you fully understand the steps of the algorithm that you are trying to implement. The design of the algorithm can either be broken into smaller parts (modular) or kept as one big part (non-modular). It is recommended that, as new coders, you start with the non-modular approach.

At this point, it is a good idea to do a “dry run” of the algorithm. This means that you as the programmer, should manually go through the algorithm using a pen and paper to track how the values of the variables change, to ensure everything works as expected.

2. **Translating the algorithm steps to code:**
   a. If needed, break the algorithm down into smaller parts.
   b. Write Python code that matches each step, using the right programming tools. These tools include simple sequences (one step after another), decisions/selections (if statements), loops (repeating steps), and functions. If your algorithm needs data structures such as arrays or stacks, make sure to include them in your code.
   c. Handling inputs and outputs: Write code that manages the inputs needed by the algorithm. You can use the assignment operator (=) for storing values, the input() function to get input from the user, and the print() function to display results. You can also read and write data from a file in Python, using special file-handling functions.
3. **Creating a new Python file:** Open your chosen IDE and create a new Python file. Type your code into the file. To make your program easier to understand, use clear and meaningful variable names, add some spaces between the lines of code (whitespace), and include comments (using the ‘#’ symbol) to explain what your code is doing. Finally, save your program with a good file name in the folder where you keep your Python files.
4. **Testing and debugging:** Test your code using different sets of data to make sure it gives the correct results. You have already learned how to create good test data sets in earlier learning. The coding environment (IDE) you are using has built-in tools to help you find and fix any mistakes (debugging).
5. **Optimisation and refinement:** Take time to review your code and look for ways to make it run faster (optimize performance), how to make it easier to understand (improve readability), and to get rid of any parts that are not needed (remove unnecessary code or comments).
6. **Code review and feedback:** If applicable, share your code with your peers or mentors for code review. Feedback can help you enhance your implementation and overall solution to the algorithm.
7. **Integration (if applicable):** If your algorithm is part of a bigger application or project, make sure to properly integrate (connect) your Python code with the rest of the project where it is needed.
8. **Deployment (if applicable):** If your code is part of a software product, make sure it is properly set up (deployed) and works as expected in the real-world setting where people will use it (production environment).

Implementation Exemplars

Example 1 – a simple maths algorithm

We have already seen this algorithm, both in pseudocode and as a flowchart in our earlier learning.

Algorithm	Program code	Output

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*
	x = 529
	y = 256
	z = x+y print(z)
		785

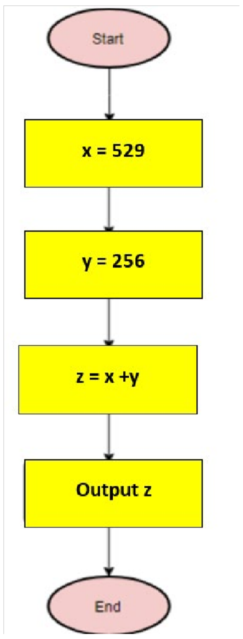
		Flowchart

Figure 4.44: *Algorithm in two formats with program and output*

Example 2 – a swap algorithm

A swap algorithm is a simple way to exchange the values of two variables. Imagine you have two cups, one with water (Cup A) and one with juice (Cup B) and you want to swap the contents without spilling anything. Here is how it works:

Step 1: Find an empty cup (Temporary Cup).

Step 2: Pour the water from Cup A into the Temporary Cup.

Step 3: Pour the juice from Cup B into Cup A.

Step 4: Pour the water from the Temporary Cup into Cup B.

Now, Cup A has juice, and Cup B has water. This is exactly how a swap algorithm works in programming.

Here is an example in Python programming

```
# Let's say we have two variables:
a = 15   # Cup A with water
b = 10  # Cup B with juice
print(f" a = {a} (water) and b = {b} (juice)")
# Create another variable “temp” to help with the swap
temp = a  # Temporary Cup holds water
a = b
b = temp
print(f" Now, a = {a} (juice) and b = {b} (water)")
```

Activity 4.44

1. Study the given problem specification below and the corresponding algorithms.
2. Do a dry run of the algorithm to calculate the expected output (using your chosen user inputs, where applicable).
3. Implement your algorithm using a programming language that you are familiar with (Python is good choice here).
4. Break down the steps further where required before implementation and add comments to explain what your code is doing.
5. Test your program, making any corrections and improvements before saving the final version.

Problem Specification

You are hired to write a program to be used in Accra Mall, by a salesperson. The program is to find the total price of a pair of trousers and a jacket on sale. The original prices of the two items should be entered by the user (salesperson). 10% discount should be applied on the trousers and 5% on the jacket. A detailed receipt, showing both discounted items, should be output, including the total cost.

Algorithm

1. Enter cost of the pair of trousers in GH¢
2. Enter cost of the jacket in GH¢
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.

Activity 4.45

1. Write an algorithm for each of the following set of problem specifications.
2. Use pseudocode to illustrate your solution.
3. Plan what data you will use to test your program.
4. Do a dry run of your algorithm and record the outcomes.
5. Implement the algorithm using your chosen programming language and suitable commentary.
6. Test your program with your test data.
7. Share your result to your peers and facilitator.

Problem Specifications

1. A program is required to calculate the area and perimeter of your classroom. The program should ask the user for the length and width of the classroom.
2. Write a program to convert and display a temperature from Celsius to Fahrenheit. Using the formula .
3. Write a program to find and output the maximum of two numbers entered by the user.
4. Write a program that checks whether a number entered by the user is even or odd.
5. Write a program that generates a random number between 1 and 10. The program should allow the user to guess the number and then give the user feedback to say whether the guess is too high, too low or correct.

***Hint**. The input should be int and the program should use Python's random. randint() function to generate the random number.*

6. Write a program that generates and displays the multiplication table for a number entered by the user. The multiplication table for that number should go from 1 to 12.
7. Create a simple quiz program that asks the user five (5) mathematics related questions (e.g. addition, subtraction, or multiplication). The program should then give the user feedback on whether the answer is correct or notand show the user's total score at the end.
8. Write a program that calculates a person's age based on the year that they were born.
9. Create a program that generates and outputs a random password of 8 characters, which includes letters (both uppercase and lowercase), digits and symbols.

***Hint**: use Python's random module and string manipulation to generate the random password.*

Activity 4.46

Do this activity in pairs. A pair forms a team for this activity. Each team gets two items (e.g., a pen and a pencil) and a basket or bowl that will act as a temporary location (Temp) for storing one item during the swap. Each team starts with the two distinct items.

For example: student A holds a pen while student B holds a pencil.

Let us begin:

1. On your facilitator's signal ("Ready, Set, Swap!"), swap the items using the following steps:

   Step 1: Student A puts their item (pen) in the basket (Temp).

   Step 2: Student A takes the item from Student B (pencil).

   Step 3: Student B takes the item from the basket (pen).
2. The first pair to successfully swap the items without confusion wins the round.

NB: Each team will be timed, and the fastest team wins the competition.

The activity you just performed is called **swapping**. Let us look at swapping in more detail.

1. The Concept of Swapping

The swap algorithm is a simple way to exchange the values of two variables in programming. It helps us switch the contents of two variables, so that the first one takes the value of the second, and the second takes the value of the first. This is useful when we want to change the values between two variables without losing any information.

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

a. Store the value of the first variable in a temporary variable.

b. Assign the value of the second variable to the first variable.

c. 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, which prevents the loss of data.

2. Manual Swapping

Imagine you and your friend each have a piece of fruit. You have an orange, and your friend has a mango, but you both want to swap fruits.

How do you swap your fruits without mixing them up?

Since you do not want to drop the orange or mango, you get an **extra basket** called **Temp** to hold one fruit temporarily.

Put your orange into the extra basket, take the mango from your friend and hold it, now, let your friend take the orange from the basket.

This scenario and the steps you used to swap the pen and pencil in Activity 46 show how manual swapping is done.

In programming, this would be the code to swap two variables manually:

```
Temp = A    # Step 1: Store A (Orange) in Temp
A = B       # Step 2: Move B (Mango) to A
B = Temp    # Step 3: Move Temp (Orange) to B
```

Activity 4.47

1. Using one hand, demonstrate how you will swap these two items: fufu and banku.
2. Write the ordered steps you used to do the swapping in your jotter.
3. Share your notes with your peers.

3. Pseudocode for Swap Algorithm

Note that, pseudocode is a way to represent an algorithm in a plain text or language.

Figure 4.45 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
b = 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 4.45: Swap algorithm 1

It is important to note that the names of the variables in the code can be different from those used when explaining the algorithm. In the example provided, the names are the same but are written in different cases. For example, in Python, it is common practice to use lowercase letters for variable names. This helps maintain readability and follow Python’s naming conventions.

Figure 4.46 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 4.46: Swap algorithm 2

If you are familiar with **subroutines** (or functions) in Python, you can experiment with creating a **swap function** to swap the values of two variables. In the example shown in Figure 4.47, you can define a function called swap that takes two variables, **A** and **B**, as inputs, swaps their values, and then returns them.

Algorithm for swap function:

```
Swap (A, B)
  1.  Temp = A
  2.  A = B
  3.  B = Temp
```

Coding (Modular):

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

#Main program
firstNumber = 49
secondNumber = 61
print ("The original values of first number and second number are", firstNumber, " and " secondNumber)
firstNumber, secondNumber = swap(firstNumber, secondNumber)
print ("The new values of first number and second number are", firstNumber, " and ", secondNumber)
```

Output:

```
The original values of first number and second number are 49 and 61
The new values of first number and second number are 61 and 49
```

Figure 4.47: Swap algorithm 3

The **swap algorithm** is a fundamental concept in programming that plays a key role in many tasks. It is used to exchange the values of two variables, and this simple operation is critical in several areas of computing, including: Sorting Algorithms such as the Bubble Sort and the Quick Sort, Data Manipulation, and Variable Swapping tasks.

Activity 4.48

1. Design and code a program that will swap two string variables using:
   a. A flowchart,
   b. Pseudocode,
   c. Python code.

2. Show the results to your teacher for feedback.

Activity 4.49

1. Write a Python program to swap two variables, x and y. Use a temporary variable and show the values before and after the swap.
2. Modify the above program to take input from the user for the values of x and y. Then, swap the values and print the result.
3. Compare your result with your peers.

Activity 4.50

1. Write a Python program that swaps the values of two variables **without using a temporary variable**, using arithmetic operations.
2. Share the result with your peers and teacher.

Example 3 – using if statement

Imagine you are deciding what to wear before leaving for school. You have two choices based on the weather.

If it is raining, you will wear a **raincoat** and **if it is not raining**, you will wear your **normal school uniform**.

In this scenario, the condition is: **Is it raining or not? The action is: if it is raining, wear a raincoat; if it is not raining, wear your normal school uniform.** This is exactly how an if/else statement works in programming.

An if/else statement is used when making decisions in real life. In programming, we use if/else to tell the computer what to do based on certain conditions, just like how we make decisions based on situations in everyday life. An example of an if/else statement in Python is as follows:

```
if raining:
   wear_raincoat()
else:
   wear_school_uniform()
```

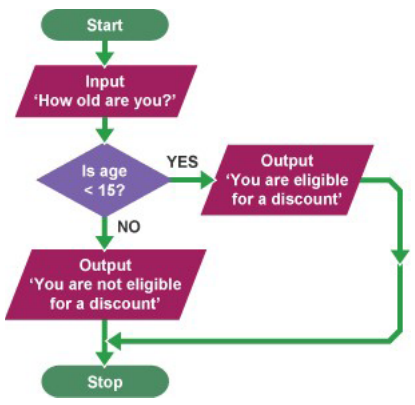
Another application of using an if/else statement is given in the problem specification below.

Problem: Takoradi Mall is giving a discount coupon to anyone who is below 15 years old. Find out if someone is eligible for a discount ticket.

Decomposing this problem, gives:

1. Find out how old the person is.
2. If the person is younger than 15 then say, "You are eligible for a discount coupon."
3. Otherwise, say "You are not eligible for a discount coupon."

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."	 Start Input 'How old are you?' Is age < 15? YES Output 'You are eligible for a discount' NO Output 'You are not eligible for a discount' Stop

Coding:
age = int(input("How old are you?")) #we use int() function to convert the string into an integer
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.")
Run 1 output:
How old are you? 26
You are not eligible for a discount.
Run 2 output:
How old are you? 13
You are eligible for a discount.	

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

Activity 4.51

1. You are at a local provision store at Krofrom and you want to buy a snack. Write pseudocode and a Python program for this condition: if you have GH¢5 or more, you will buy a meat pie and if you have less than GH¢5, you will buy a bofrot.
2. Show the result to your facilitator and peers for feedback.
3. What better way can you write the Python code to get same result? (note, this is called refactoring your code to provide improvements)

Activity 4.52

1. For his upcoming annual Ashaiman to the World Concert, Stonebwoy is giving discount tickets to anyone who sings any ten (10) or more of his songs. Find out if someone is eligible for a discount ticket.
   a. Write pseudocode for the program.
   b. Draw a flowchart for the program.
   c. Implement the algorithm using any programming language of your choice. *Python is preferred.*
2. Show your answer to your facilitator for review.

Hello learner! We are sure your excitement for this course keeps growing as we move forward. This time, we will look at implementing and manipulating 1D arrays and built-in (array) list methods in Python.

Activity 4.53

1. Reflect and write a short note on what you remember from the lesson on data structures.

2. Complete Table 4.11 to give a suitable description, name and data type for the set of array descriptions.

Table 4.11: Array descriptions

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		
	favouriteSongs ()	
Top 10 songs in the charts this month		
Answers to five True/False questions		
Answers to five multiple-choice questions		
		string
		float
		bool

3. Share your responses with your peers and teacher.

ARRAYS IN PYTHON

In our earlier learning, we looked at how an "array" is one of the most fundamental data structures in any programming language. You learnt that the elements in an array data structure are arranged on a straight path or has 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.

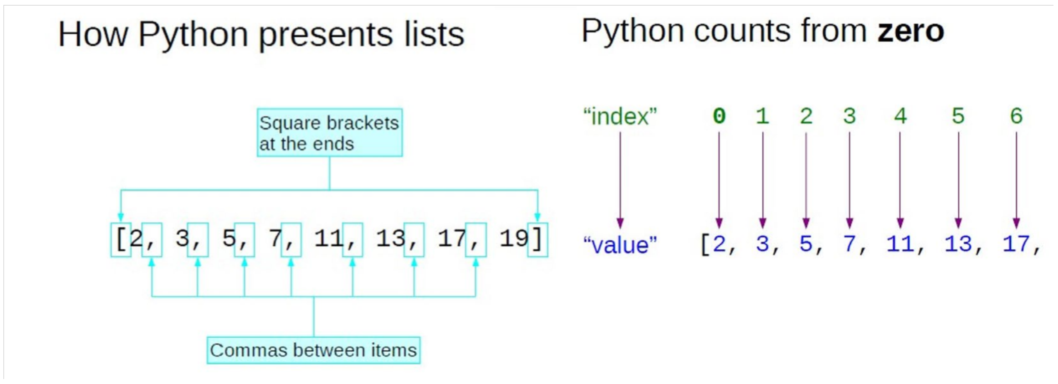


Figure 4.48: List in Python

Python lists can contain duplicate values as seen in Figure 4.39 below. "Apple" appears twice in the array.

Please note that as we progress through this learning, we will use the term 'array' instead of list.

You already know how to reference a particular element in an array as we covered that earlier in our learning – we use the index of the element.

For example, if the array in Figure 4.48 was called *numbers*, then the code print (numbers[1]) would output 3.

To be able to code and fully manipulate arrays in a Python program, you need to understand the terms 'iteration' and 'selection' in programming. These will be covered later in your learning. For the time being, we will use in-built codes called functions and methods for the implementation of algorithms that use arrays. 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, we can use the **len() function** – see Figure 4.39.

```
4
```

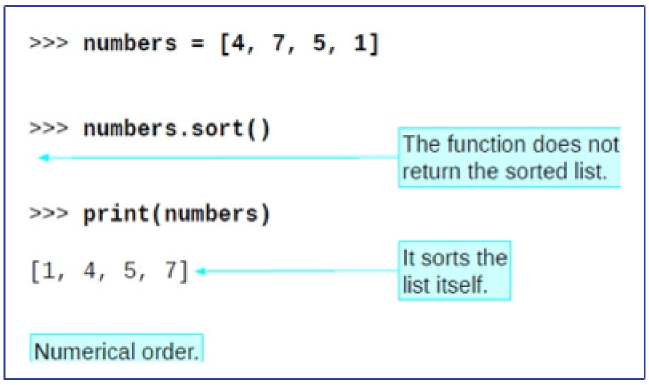
```
fruitList = ["apple", "banana", "mango", "apple"]

print(len(fruitList))
```

Figure 4.49: *A program using len() and output*

A brief description of some common list methods in Python

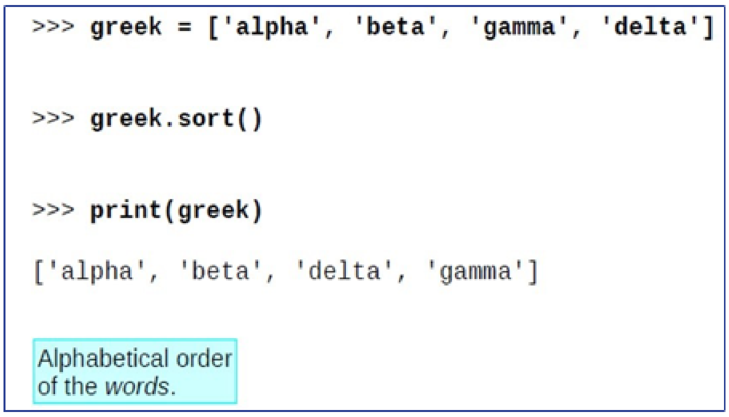
1. *clear*(): Removes all elements from the list.
2. *copy():* Creates a duplicate of the list.
3. *count():* Returns the number of occurrences of a specified value in the list.
4. *index():* Returns the index of the first occurrence of a specified value.
5. *reverse():* Reverses the order of elements 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 4.50



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

>>> numbers.sort()

>>> print(numbers)
[1, 4, 5, 7]
```

The function does not return the sorted list.
It sorts the list itself.
Numerical order.



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

>>> greek.sort()

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

Figure 4.50: Checking out how Sort () method works in the Python console/shell

Algorithms Implementation (continued)

Now that you have been introduced to the concept of "swapping" and "if statements", let us look at them in detail.

In the swap algorithm, we seen that it is a standard algorithm used to exchange the values of two variables. It allows us to interchange the contents of two variables without losing their original data. On the other hand, an "if statement" in coding allows the program to make decisions based on a condition. If the condition is true, the code inside of the "if" block is executed and if it is false, the code inside of the "else" block is executed, enabling conditional execution.

Now, let us consider the reversing of an array in the examples below.

Example: reversing an array

Problem: Design and write a program that sets up an array of three daily activities of your choice (you are to choose elements of your choice but do not duplicate any element). Use in-built Python code to reverse the order of the elements in the array. Each element in the reversed array should then 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.

Code	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 4.51: Reverse list in Python

Activity 4.54

Plan and write a program that set up an array of seven activities of your choice that happen in the school without duplicating any of the element.

1. Write an algorithm for the planned activities.
2. Use in-built Python code to reverse the order of the elements in the array.
3. Print each element on a separate line.
4. Work with a peer to come up with a more efficient way algorithm to give the same output and discuss as a group.

Example: counting occurrences in an array

Problem: Design and write a program that sets 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 computers are 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 – tablet and mainframe.
3. Use an in-built method in the programming language to find the number of occurrences of the two types in the array.
4. Output the number of occurrences of each type to be checked in the array.

Code	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

Activity 4.55

Example: copying an array

The term copying an array simply means creating and duplicating the elements in the data structure from the original array to a new one.

On the other hand, copying an array in data structures means creating a new array that contains the same elements as the original array. However, depending on how the copy is made, the new array may be independent (deep copy) or linked to the original (shallow copy).

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 to make a copy of this array with the name *copy “SnakeList”*.
3. Output the original array and the copied array.

Code	Output

#set up array of 5 types of snakes
snakesList=[“python”,”cobra”,”mamba”,”viper”, “adder”]
#Make a copy of this array
copySnakessList = snakesList.copy()
#print out both arrays
print (“SnakesList”)
print(snakesList)
print (“Copy of SnakesList”)
print(copySnakessList)
	SnakeList
	[‘python’,’cobra’,
	‘mamba’,’viper’, ‘adder’]
	Copy of SnakeList
	[‘python’,’cobra’,
	‘mamba’, ‘viper’, ‘adder’]

Activity 4.56

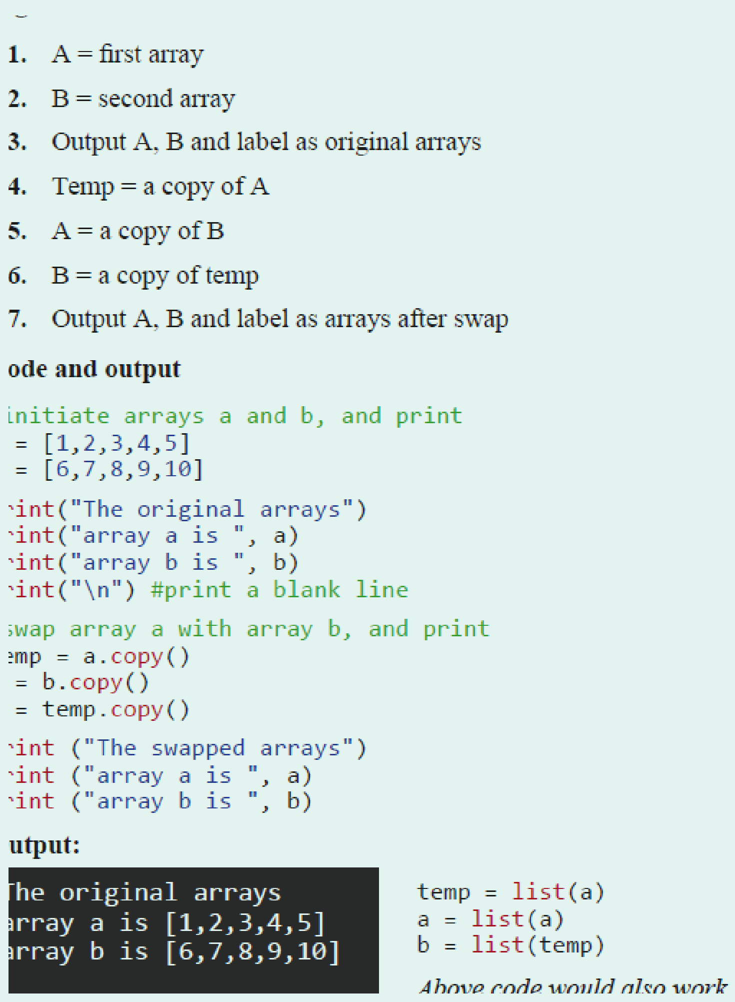
1. Create a list of your favourite books and make a copy using the copy() function.
2. Create another list of your hobbies and make a copy using the list() function.
3. Write a program that duplicates a list of your favourite foods using list slicing ‘[:]’.
4. Write your observation of using the copy() function, the list() function and slicing ‘[:]’.

Example: swap array algorithm

The swap array algorithm normally swaps the positions of the elements in an array. For instance, if you have an array `[1, 2, 3, 4]` and want to swap the first and last elements:

Before: `[1, 2, 3, 4]`
After: `[4, 2, 3, 1]`

Another instance for swapping arrays A and B is shown in Figure 4.52.



```
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
```

ode and output

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

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

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

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

utput:

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

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

Above code would also work

Figure 4.52: Swap array algorithm

Activity 4.57

Copy and complete the code below and complete it to that the variable values are swapped

```
# Original variables
student1 = “Gyampo”
student2 = “Daniel”
print(“Before swapping:”)
print(“Student 1:”, student1)
print(“Student 2:”, student2)
# Swapping using a temporary variable
temp =
student1 =
student2 =
print("\nAfter swapping:")
print("Student 1:",............)
print("Student 2:", ..........
```

EXTENDED READING 4.1

1. Click on here or https://repository.dinus.ac.id/docs/ajar/algoritma.pdf to read on Algorithms, Flowcharts, Data Types and Pseudocode, pages 6 – 27.
2. Click on here or https://www.codecademy.com/article/pseudocode-and-flowcharts to read further on flowcharts and pseudocode.
3. Click on here or https://www.youtube.com/watch?v=xUmyzEr8P-A to watch a video on how to convert flow chart to pseudocode (computer).
4. Research on how to 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.
5. Problem Solving with Algorithms and Data Structures by Brad Miller, David Ranum (2013), pages 3 – 9.

EXTENDED READING 4.2

1. Click on here or https://www.youtube.com/watch?v=Xe5qQBI8MeM to watch a video on SDLC system development life cycle.
2. SDLC by Alan D., Barbara H. W., Roberta M. R., (2012) System Analysis and Design, pages 10 – 17, 5th Edition, VP & Publisher.

EXTENDED READING 4.3

1. Hello!, for further reading on data structure click here or use this link Data Structure Types, Classifications and Applications - GeeksforGeeks.
2. Click here or the link below to read on the differences between data structure and data type. https://www.geeksforgeeks.org/what-is-data-structure-types-classifications-and-applications/.
3. Problem Solving with Algorithms and Data Structures by Brad Miller, David Ranum (2013), pages 61 – 116.
4. 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.

EXTENDED READING 4.4

1. Click here or https://www.youtube.com/watch?v=5rVWUrYBz4M to watch a video of a Tic-Tac-Toe game using an array as a data structure.
2. Click here or https://www.youtube.com/watch?v=3_x_Fb31NLE&list=PLqM7alHXFySEQDk2MDfbwEdjd2svVJH9p to learn further information on the array data structure and its operation.
3. Research jagged arrays and create a short summary of your findings. Include examples of jagged arrays in your summary.

4. Click here or https://www.w3schools.com/python/python_datatypes.asp to learn more on array data types.

EXTENDED READING 4.5

1. 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:
   a. Stack, push and pop operations.
   b. Inserting a data item at the beginning of a singly linked list.
   c. Inserting a data item at position N in a singly linked list.
   d. Accessing an item in an array.

EXTENDED READING 4.6

1. Click here or the link below to read more on queues and non-linear data structure.

   https://cse.sc.edu/~bjoshi/csce101/attachments/Stack,%20Queue%20and%20Trees.pdf
2. Find out what is meant by a priority queue and write a short report on your findings.
3. Problem Solving with Algorithms and Data Structures by Brad Miller, David Ranum (2013), pages 61 – 116.
4. Research what is meant by the 'Big O notation' and describe how it links to data structures and algorithms.

EXTENDED READING 4.7

1. Use the link or 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. Research another popular high-level programming language other than Python. Create a table in Word to summarise a comparison between this language and Python, using suitable criteria.
3. Investigate how Python is used in data science and analytics.
4. Click here to download or here to read *Introduction to Python Programming* by OpenStax (2024) retrieved from https://openstax.org/books/introduction-python-programming/pages/1-1-background on 11/09/2024, page 7-37.
5. Click here or https://www.youtube.com/watch?v=_uQrJ0TkZlc to learn more about Python.

EXTENDING READING 4.8

- Research how arrays (lists) can use used in Python to implement the following data structures: a stack, a linear queue and a linked list. Use your research findings to implement one of the listed data structures using an array.
- Click here or https://www.youtube.com/watch?v=6a39OjkCN5I to watch a video on array implementation in Python.
- Chan J., (2014). Learn Python in One Day and Learn It Well Python for Beginners with Hands-on Project. The only book you need to start coding in Python immediately. Page 7 – 54.
- OpenStax (2024) Introduction to Python Programming retrieved from https://openstax.org/books/introduction-python-programming/pages/1-1-background on 11/09/2024, page 7-37.
- Park A. (2022) - Python Programming for Beginners: The Ultimate Crash Course to Learn Python in 7 Days with Step-by-Step Guidance and Hands-On Exercises. ISBN 13: 9798836767464

Review Questions

Review Questions 4.1

1. Imagine you have a routine you follow every school day, from the time you wake up until you arrive at school. Think of it like a series of steps that you follow one after the other. Write pseudocode that describes this school day routine.
2. You have been requested 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:

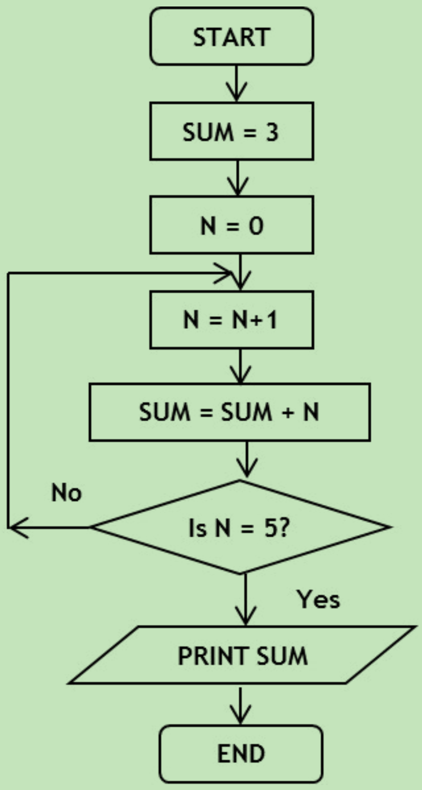
   a. Enter first number, x

   b. Enter second number, x

   c. Let answer = x divided by x

   d. Output answer

   Will this algorithm work? Justify your answer.
3. Interpret the complex flowchart below and state the output by showing your working.



Reviewed Questions 4.2

1. Explain the role of the client at the Analysis stage of SDLC.
2. Some software development models are considered iterative. This means that 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

Review Questions 4.3

1. Explain the difference between static and dynamic data structures and give an example of each.
2. As a member of the computing club in your school, imagine you need to manage an everchanging 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.

Review Questions 4.4

1. A data structure is needed to store the monthly temperatures over a three-month period as shown below. Recommend a data structure to use and give a reason for your choice.

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

2. As a Year 1 Computing learner, you are tasked to store data for the seating plan for your class, write a Python code for the seating plan.

Review Questions 4.5

1. Recommend a data structure to use for implementing an undo feature in a text editor. Give the 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.

Review Questions 4.6

1. Recommend a data structure for each of the following applications. In each instance, give at least one reason for your choice of structure:
   a. CPU task scheduling.
   b. Reversing a list of stored characters.
   c. Storing a song playlist to play previous and next song.
2. Explain why a programmer would choose a contiguous memory data structure to implement a queue rather than a non- contiguous memory structure.

Review Questions 4.7

1. Miss Joyce Sarkpoh 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.

   As a computing student,
   a. State the number of variables Miss Joyce Sarkpoh would require in her program.
   b. State a possible name and data type for each of these variables.
   c. Using these variable names, write the code for the two calculations.
2. 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.
3. Design and code a foreign exchange calculator program for converting GH□ to CFA. The user should enter the amount to be converted and the exchange rate. The result of the conversion should be output to the user.
4. Joycee's Bank lends money to customers on a regular basis. The bank needs a program that can calculate and output simple interest on the loans using the formula

   Where P is the principal amount, R is the rate of interest per year, T is the time in years. The input: P, R, and T should either be floats or integer.

   As a computing student, write this program for the bank to buy.
5. A program is needed to take three numbers as inputs, calculate and output the sum of the numbers and the average of the numbers. As a computing student, write this program.

Review Questions 4.8

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

   [‘Father’, ‘Mother’, ‘Son’, ‘Daughter’]

   a. Write the line of code that must be added to update the first element to ‘Grandpa’.

   b. Write a line of code that will print the third element in the array.

   c. Write one line of code that will sort the family’s array elements into alphabetical order

   d. Test these lines in the Python IDE environment to check your answers.

2. Design a program that will:

   a. create an array called subjects consisting of three school subjects to be inputted by the user.

   b. swap the first and last subject element.

   c. output two versions of the array – the original order and then the order after the swap.

Answers to Review Questions

Answers to Review Questions 4.1

1.
```
BEGIN
    Wake up at 6:00 AM
    Brush teeth
Take a bath
    Dress up in school uniform
    Eat breakfast
Pack school bag with books and supplies
    Leave house at 7:00 AM
    Walk to school
Arrive at school by 7:30 AM
END
```

2. The algorithm will not work correctly due to 2 critical issues:
   a. Variable Naming Conflict: Both the first number and the second number are being entered and stored with the same variable name, x.
   b. Doubt in Calculation: The algorithm specifies "Let answer = x divided by x". Since both numbers are stored as x, this operation will always result in 1 (or an error if zero is entered for the second number).

3. Interpretation
   a. Initialisation:

      The algorithm starts with the `START` symbol.

      The variable `SUM` is set to 3.

      The variable `N` is set to 0.

   b. Iteration (loop):
      The algorithm enters a loop:
      `N` is increased by 1 (`N = N + 1`).
      The current value of `N` is added to `SUM` (`SUM = SUM + N`).
      The algorithm checks if `N` is equal to 5 in the decision symbol
      If `N` is not 5, the loop continues.
      If `N` is 5, the loop ends.
   c. Output:

The final value of `SUM` is printed.

The algorithm ends with the `END` symbol.

d. working:

Iteration	N	SUM
1	1	$3 + 1 = 4$
2	2	$4 + 2 = 6$
3	3	$6 + 3 = 9$
4	4	$9 + 4 = 4$
5	5	$13 + 5 = 18$

Therefore, the output of the algorithm is 18.

Answers to Review Questions 4.2

1. At the Analysis stage of the Software Development Life Cycle (SDLC). The client is used to clearly explain what they want the software to do. They share their needs, goals, and any specific features they want in the software. The development team then use this information to understand what the software should include and how it should work.

2.

   a. Testing and Design: After you have developed your software and started testing it, you might find some problems or areas where the software is not working well. When this happens, you can go back to the "design stage" to fix these issues.

   b. Design and Analysis: When you are developing software, you rely on information gathered during the "analysis stage". But as you design, you might realise that the initial analysis missed something important or did not cover all the details. In this case, you can return to the "analysis stage" to gather more information or make adjustments. This helps to ensure that the design of the software matches exactly what is needed.

Answers to Review Questions 4.3

1.

Static Data Structures:	Dynamic Data Structures:
• **Size Fixed at Compile Time**: The size of a static data structure is determined when the program is compiled and cannot change during runtime	• **Size Can Change at Runtime**: The size of a dynamic data structure can grow or shrink during the execution of the program, depending on the amount of data being stored
• **Memory Allocation**: Memory is allocated in advance, and since the size is fixed, it can sometimes lead to wasted memory if not all allocated space is used.	• **Memory Allocation**: Memory is allocated as needed during runtime, and the structure can adjust its size based on the data being added or removed
Examples	
• **An array** is a classic example of a static data structure	• **A linked list** is an example of a dynamic data structure.

2.

a. The suitable Data Structure is a "Dynamic Data Structure". This is because dynamic data structures, like linked lists, can easily adjust their size during runtime.

b. Two Regularly Performed Operations are

i. Insertion (Add a Member)

ii. Deletion (Remove a Member)

c. Uses of These Operations:

i. Insertion (Add a Member): Whenever a new student joins the computing club, their name needs to be added to the list of members.

ii. Deletion (Remove a Member): If a member leaves the club, his or her name needs to be removed from the list.

Answers To Review Questions 4.4

1. Monthly temperatures over a three-month period.

   A Two-Dimensional Array is best because it provides a clear structure to represent the data, with rows for months and columns for daily temperatures, and individual temperatures can be accessed directly using row and column indices.

2. Sample answer, your solution may vary with numbers or letters.

```
seating_plan = [
    [None, None, None, None],  # Row 1
    [None, None, None, None],  # Row 2
    [None, None, None, None],  # Row 3
    [None, None, None, None]   # Row 4
]
```

Answers to Review Questions 4.5

1. For implementing an undo feature in a text editor, the most appropriate data structure to use is a stack.

   The reason for choosing a stack is because it is ideal for an undo feature. Each action a user performs, such as typing, deleting, or formatting text, is pushed onto the stack as a new entry. When the undo command is executed, the most recent action is popped (removed) from the stack, reverting the editor to its previous state.

2. In a linked list, each piece of data also includes pointers to connect it to the next data item, which uses extra memory. Conversely, arrays store all their data tightly together in one block, using no extra space for pointers. Therefore, linked lists require more memory than arrays due to these additional pointers and the way they are spread out in memory. This makes linked lists more flexible but less memory-efficient compared to arrays, which are better when the amount of data is fixed and known.

3.

   a. Direct Access in Linked Lists: In a linked list, you can directly access any node to add or remove it, thanks to pointers that link each element. This allows for quick changes anywhere in the list.

   b. Limited Access in Stacks: In a stack, you can only add or remove items from the top. If you need to insert or delete somewhere else, you must move all the items above it first, which can take more time.

   c. No Shifting Needed: When you add or remove elements in a linked list, you don't need to shift other elements around; just adjust the pointers. In stacks, adding or removing items affects only the top, making it less flexible.

Answers to Review Questions 4.6

1. Recommended Data Structure for Specific Applications

   a. CPU Task Scheduling

   - Data Structure: Priority Queue
   - Reason: Priority queues facilitate efficient scheduling by always allowing access to the task with the highest priority next, which is essential for managing CPU tasks based on their urgency or importance.

   b. Reversing a List of Stored Characters

   - Data Structure: Stack

  - Reason: A stack operates on a Last In, First Out (LIFO) principle, which is ideal for reversing sequences. Characters pushed onto the stack will be popped in the reverse order of their original arrangement.

c. Storing a Song Playlist to Play Previous and Next Song
  - Data Structure: Doubly Linked List
  - Reason: Doubly linked lists allow for easy traversal both forwards and backwards, which is necessary for navigating between previous and next songs in a playlist.

2. Imagine a line at a bus stop. Each person in the line is like an element in a queue. In computing, we can store these elements in memory.

   There are two ways to store them:

- **Linked List:** This is like a chain of people holding hands. Each person knows the person in front and behind them.
- **Contiguous Memory**: This is like everyone standing side by side in a row.

Why choose contiguous memory for queues?

- **Faster**: It's like jumping directly to a person in the line without having to count from the beginning.
- **Simpler**: It's easier to manage people standing in a row than a chain of people holding hands.
- **Predictable**: You know exactly how many people can fit in the line.
- So, while linked lists are flexible, contiguous memory is often a better choice for queues because it's faster, simpler, and more predictable.

Answers to Review Questions 4.7

1.

a. Miss Joyce Sarkpoh would need five variables for the program:

   Water cost.

   Flavouring cost.

   Labour cost.

   Selling price.

   Profit.

b. Possible Variable Names and Data Types:

   Water cost: water_cost (float)

   Flavouring cost: flavouring_cost (float)

   Labour cost: labour_cost (float)

   Selling price: selling_price (float)

Manufacturing cost: manufacturing_cost (float, calculated)

Profit: profit (float, calculated)

c.

```
# Input values for costs and selling price
water_cost = float(input("Enter the water cost per litre: "))
flavouring_cost = float(input("Enter the flavouring cost per litre: "))
labour_cost = float(input("Enter the labour cost per litre: "))
selling_price = float(input("Enter the selling price per litre: "))
# Calculate manufacturing cost
manufacturing_cost = water_cost + flavouring_cost + labour_cost
# Calculate profit
profit = selling_price - manufacturing_cost
# Output the results
print(f"Manufacturing cost per litre: GH¢ {manufacturing_cost}")
print(f"Profit per litre: GH¢ {profit}")
```

2. Step-by-Step Algorithm:
   - **i.** Start
   - **ii.** Store the value of number1 in a temporary variable, temp.
   - **iii.** Assign the value of number3 to number1.
   - **iv.** Assign the value of number2 to number3.
   - **v.** Assign the value of temp (which holds the original value of number1) to number2.
   - **vi.** End

3. **Steps to Design the Program**:

   The user inputs the amount in Ghana Cedis (GH¢) to be converted.

   The user also inputs the exchange rate (how many CFA one GH¢ is worth).

   The program performs the conversion using the formula: CFA amount = GH¢ amount × exchange rate.

   The program outputs the equivalent amount in CFA.

The code is:

```
# Input the amount in GH¢ to be converted
gh¢_amount = float(input("Enter the amount in GH¢: "))
# Input the exchange rate (1 GH¢ to CFA)
exchange_rate = float(input("Enter the exchange rate (GH¢ to CFA): "))
# Calculate the amount in CFA
cfa_amount = ghc_amount × exchange_rate
# Output the result
print(f"{ghc_amount} GH¢ is equivalent to {cfa_amount} CFA at an exchange
rate of {exchange_rate}.")
```

4. The Simple Interest Code:

```
# Input the principal amount (P)
P = float(input("Enter the principal amount (P): "))
# Input the rate of interest per year (R)
R = float(input("Enter the rate of interest per year (R): "))
# Input the time in years (T)
T = float(input("Enter the time in years (T): "))
# Calculate the simple interest using the formula I = (P * R * T) / 100
I = (P * R * T) / 100
# Output the calculated simple interest
print(f"The simple interest on a loan of GHS {P} at a rate of {R}% for {T} years
is: GHS {I}")
```

5.

```
# Input the three numbers
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
# Calculate the sum of the numbers
sum_of_numbers = num1 + num2 + num3
# Calculate the average of the numbers
average = sum_of_numbers / 3
# Output the sum and the average
print(f"The sum of the numbers is: {sum_of_numbers}")
print(f"The average of the numbers is: {average}")
```

Answers to Review Questions 4.8

1.

   a. Creating the array ‘seasons’

   ```
   family = [‘Father’, ‘Mother’, ‘Son’, ‘Daughter’]
   ```

   b. Updating the first element to ‘Grandpa’

   ```
   family[0] = ‘Grandpa’
   ```

   c. Printing the third element in the array

   ```
   print(family[2])
   ```

   d. Sorting the seasons array in alphabetical order

   ```
   family.sort()
   print(family)
   ```

2. Code

```
subjects = []
for i in range(1,4):
    subject= input(f”Enter subject{i}: “)
    subjects.append(subject)
print(“Original array:”, subjects)
temp = subjects.copy()
subjects[0], subjects[-1] = temp[-1], temp[0]
print(“Array after swapping first and last elements:”, subjects)
```

REFERENCE

- Alan D., Barbara H. W., Roberta M. R., (2012) System Analysis and Design, pages 10 – 17, 5th Edition, VP & Publisher.
- Blogs, B., & Software. (2022, September 8). 4 steps of a successful software development life cycle. Vates. https://www.vates.com/4-steps-of-a-successful-software-development-life-cycle/
- Brad M., David R. (2013) Problem Solving with Algorithms and Data Structures, Release 3.0, pages 3 – 116.
- Chan J., (2014). Learn Python in One Day and Learn It Well Python for Beginners with Hands-on Project. The only book you need to start coding in Python immediately. Page 7 – 54 retrieved from https://www.pdfdrive.com/a-python-book-beginning-python-advanced-python-and-python-d9236005.html on 17/09/2024.
- Click here or https://www.tutorialspoint.com/sdlc/index.htm.
- Computing – Teacher Manual (2024), Year One Book 2 for SHS, SHTS, STEM curriculum – pages 4 – 95.
- Computing – Teacher Manual (2024), Year One Book 2 for SHS, SHTS, STEM curriculum – pages 64 – 95.
- GeeksforGeeks. (n.d.). Linked List Data Structure. Retrieved from https://www.geeksforgeeks.org/linked-list-data-structure/.
- GeeksforGeeks. (n.d.). Stack Data Structure. Retrieved from https://www.geeksforgeeks.org/stack-data-structure/.
- https://www.javatpoint.com/software-development-life-cycle.
- Joshi, B. (n.d.). Stack, Queue and Trees. Retrieved from https://cse.sc.edu/~bjoshi/csce101/attachments/Stack,%20Queue%20and%20Trees.pdf
- NaCCA (2023) Computing curriculum for Secondary Education (SHS 1-3) pages 44.
- NaCCA (2023) Computing curriculum for Secondary Education (SHS1-3) pages 42 – 45.
- NaCCA (2023) Computing curriculum for Secondary Education (SHS1-3) pages 48 – 49.
- Narasimha K. (2016) - Data Structures And Algorithmic Thinking With Python, CareetMonk Publications, retrieved from https://www.pdfdrive.com/data-structures-and-algorithms-in-python-e25119593.html on 22/08/2024, pages 14 - 19.
- OpenStax (2024) Introduction to Python Programming retrieved from https://openstax.org/books/introduction-python-programming/pages/1-1-background on 11/09/2024, page 7-37.
- Park A. (2022) - Python Programming for Beginners: The Ultimate Crash Course to Learn Python in 7 Days with Step-by-Step Guidance and Hands-On Exercises.

ISBN 13: 9798836767464 Publisher: Independently published, retrieved from https://www.amazon.com/Python-Programming-Beginners-Step-Hands/dp/B0B3S3RFQT on 17/09/2024.

- Programiz. (n.d.). Linked List. Retrieved from https://www.programiz.com/dsa/linked-list.
- Programiz. (n.d.). Stack. Retrieved from https://www.programiz.com/dsa/stack.
- Radix Learning Private Limited (2017). Actionable learning, practice & assessment for algorithms, data structures and problem solving.
- Tutorials Point. (n.d.). Linked List Algorithms. Retrieved from https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm.
- Tutorials Point. (n.d.). Stack Algorithm. Retrieved from https://www.tutorialspoint.com/data_structures_algorithms/pdf/stack_algorithm.pdf

ACKNOWLEDGEMENTS

List of Contributors

Name	Institution
Dordoe Senyo Raphael	Ziavi SHTS
Eric Asomani Asante	Naylor SDA School
Miheso Daniel	Wa SHTS
Osei Amankwa Gyampo	Kumasi Wesley Girls High School