UNIT 2 PROGRAMMING

Assessment Objective 1

Demonstrate knowledge and understanding of the key principles of computer science

Assessment Objective 2

Apply knowledge and understanding of key concepts and principles of computer science

Assessment Objective 3

Analyse problems in computational terms:
- to make reasoned judgements
- to design, program, test, evaluate and refine solutions

In this unit you will learn about translating algorithms into programs written in a high-level language using constructs, such as variables and arrays. You will also look at how programs can be structured using subprograms and how data input can be validated to ensure that it is reasonable. Humans are not perfect. Most programs have one or two errors and you will look at how programs can be tested, and errors corrected.

5 DEVELOP CODE

PROGRAMMING LANGUAGES

Throughout the book, we have coloured the different programming languages in order for you to easily find the one you are looking for.

The colours are:

`Pearson Edexcel pseudocode`

`Python`

`Java`

`C#`

Once an algorithm has been developed to solve a particular problem, it has to be coded into the programming language that the developer is using. This usually means that the written descriptions, flowcharts and pseudocode have to be converted into actual programming code.

As you will be expected to understand, use and edit Pearson Edexcel `pseudocode` in the examination, examples will be given in the pseudocode as well as in `Python`, `Java` and `C#`.

LEARNING OBJECTIVES

- Explain the difference between algorithms and programs
- Code an algorithm in pseudocode and a high-level programming language
- Describe the characteristics of data types and select appropriate data types for variables
- Use sequencing, selection and iteration constructs in your programs

ALGORITHMS AND PROGRAMS

As you learnt in Unit 1, an algorithm is a precise method of solving a problem. It consists of a sequence of unambiguous, step-by-step instructions. A program is an algorithm that has been converted into program code so that it can be **executed** by a computer. A well-written algorithm should be free of logic errors and easy to code in any high-level language.

As part of this course you will learn to write programs in a high-level programming language. All high-level programming languages are like natural human languages, which makes them easier for humans to read and write but impossible for computers to understand without the help of a translator. You will learn more about how a program written in a high-level language is translated into machine code – the language of computers – in Unit 4. The aim of this unit is to develop your programming skills.

SUBJECT VOCABULARY

execution the process by which a computer carries out the instructions of a computer program

▶ A computer programmer at work writing code

DATA TYPES

GENERAL VOCABULARY

determine to discover facts about something

As you learnt in Unit 1, algorithms use variables (named memory locations) to store values. Variables have a variety of uses. For example, controlling the number of times a loop is executed, **determining** which branch of an IF statement is taken, keeping running totals and holding user input.

When algorithms are converted into programs, the computer needs to be told what type of data is stored in each variable. Every programming language has a number of built-in data types.

DATA TYPE	DESCRIPTION	EXAMPLE	EXAMPLES OF USE
integer	Used to store whole numbers without a fractional part	30	age = 30 number = 5
real or float	Used to store numbers with a fractional part (decimal place). Real numbers are sometimes referred to as floats (short for floating point)	25.5	weight = 25.5 price = 12.55
Boolean	Only has two possible values: True or False	False	correct = False lightOn = True
character*	A character can be a single letter, a symbol, a number or even a space. It is one of the four basic data types	'm'	gender = 'm' char = ':'
string	A set of characters which can include spaces and numbers and are treated as text rather than numbers	'the computer'	name = 'Catherine' type = 'liquid'

*Python does not have a character data type.

▲ **Table 2.1** Common data types

GENERAL VOCABULARY

assign to give somebody a particular task

When writing pseudocode, you don't have to specify the data types of variables. However, data types become much more important once you start programming in a high-level language. This is because the data type of a variable determines the operations that can be performed on it.

For example, the result of multiplying a value by 5 differs according to its data type.

integer	8 * 5 = 40
real	8.0 * 5 = 40.0
character	'8' * 5 = '88888'

HINT

Although you don't have to declare the data types of variables in pseudocode, it is a good idea to do so. You can simply list the variables and their data types at the start of an algorithm, for example:

```
INTEGER age
REAL weight
BOOLEAN correct
CHARACTER gender
```

You can also specify the data type of a variable in the RECEIVE statement, for example:

```
RECEIVE age FROM (INTEGER) KEYBOARD
RECEIVE price FROM (REAL) KEYBOARD
```

The method of declaring variables differs between programming languages. Some languages, such as Python, automatically select the appropriate data type for a variable based on the data **assigned** to it. Others, such as Java and C#, require the data type of variables to be declared before the variables can be used.

VARIABLE INITIALISATION

When a variable is declared, the computer gives it a location in its memory. Initially, this location is empty, so before a variable can be used it has to be given a value.

SUBJECT VOCABULARY

initialise to set variables to their starting values at the beginning of a program or subprogram

initialisation the process of assigning an initial value to a variable

assignment statement the SET…TO command is used to initialise variables in pseudocode, for example:

```
SET anotherGo TO 0
SET correct TO False
```

You can put an initial value into a variable by:

- initialising it when the program is run (e.g. `SET total TO 0`, in pseudocode)
- reading a value from a keyboard or other device (e.g. `RECEIVE admissionCharge FROM (INTEGER) KEYBOARD`).

Once a variable has been **initialised** an **assignment statement** is used to change its value (e.g. `SET total TO total + admissionCharge`).

In Python this would be:

```
total = 0
total = total + admissionCharge
```

In Java this would be:

```
Scanner scan = new Scanner(System.in);
int admissionCharge = scan.nextInt();
int total = 0;
total = total + admissionCharge;
```

In C#, variables must be declared before use. When you declare a variable, you need to state the data type that the variable will store, for example:

```
// declare a variable called total that will be used to store floating point numbers and assign the value 0.0 to it
float total = 0.0;
// add the value stored in the variable admissionsCharge to the total
total = total + admissionsCharge;
```

GENERAL VOCABULARY

intended designed for a certain purpose

If a variable, such as a loop counter, is **intended** to hold a running total, then it should always be initialised to a starting value. Some programming languages won't execute if the programmer fails to do this; others will do so but may well produce some unexpected results.

SKILLS CRITICAL THINKING, DECISION MAKING

ACTIVITY 1

DATA TYPES

1 Investigate what data types are available in the high-level language you are studying. Produce a table similar to Table 2.1 to give a summary of your findings.

2 What do you think is an appropriate data type for each of these items?
   a the test score of an individual learner
   b the average score for a group of learners
   c whether or not the pass mark for the test has been achieved.

3 Look back over the algorithms you wrote in Unit 1 and find instances of variable initialisation.

SUBJECT VOCABULARY

type coercion the process of converting the value stored in a variable from one data type to another

TYPE COERCION

Sometimes the data type of a variable gets changed during program execution. This is known as **type coercion**. For example, if an integer value and a real value are used in an arithmetic operation, the result will always be a real.

In Python, type coercion is done automatically, for example, the output from the following program is 3.25.

```
x = 1
y = 2.25
z = x + y
print(z)
```

Type coercion is also automatic in Java:

```
int x = 1;
double y = 2.25;
double z = x + y;
System.out.print(z);
```

In C# this is generally referred to as casting. As with Python this can be automatic for many data type conversions and is known as implicit casting. For more complicated conversions you may need to research explicit casting.

One frequently used type of explicit casting is converting from a string to a numeric value. This is used when getting input from a user as `Console.Readline`. For example, the following C# code will output 3.25 and demonstrates automatic type coercion.

```
int x;                  // declares a variable called x that will store integers
double y, z;            // declares two variables called y and z that will store
                        floating point numbers

x = 1;
y = 2.25;
z = x + y;
Console.WriteLine(z);
```

SKILLS CRITICAL THINKING, PROBLEM SOLVING, DECISION MAKING

ACTIVITY 2

MONITORING VISITOR NUMBERS

A theme park uses a program to monitor the number of people entering and exiting the park. The maximum number of visitors at any one time must not exceed 10 000. When the number of people in the park reaches the maximum, a 'Park Full' message is displayed at the entrance gate. Children can visit the park free of charge. Adults must pay AED 125 admission. The program records the amount of money collected at the gate.

1 What are the variables needed in the program?

2 Select an appropriate data type for each variable and constant.

COMMAND SEQUENCE, SELECTION AND ITERATION

In Unit 1 you learnt that the three key building blocks of algorithms are command sequence, selection and iteration. In this unit, you will have the opportunity to implement these constructs in the high-level programming language you are studying.

SKILLS CRITICAL THINKING, PROBLEM SOLVING, CREATIVITY

ACTIVITY 3

UNDERSTANDING ALGORITHMS

Read the following algorithm written in pseudocode and then answer the questions below.

```
RECEIVE number1 FROM (INTEGER) KEYBOARD
RECEIVE number2 FROM (INTEGER) KEYBOARD
SET result1 TO number1 / number2
SEND result1 TO DISPLAY
SET result2 TO number1 MOD number2
SEND result2 TO DISPLAY
SET result3 TO number1 DIV number2
SEND result3 TO DISPLAY
```

1 What does this algorithm do?

2 What is the output of the algorithm, given the following inputs:
   a 4, 2
   b 10, 3
   c 20, 6?

Implement this algorithm in the high-level language you are studying.

SELECTION

The selection construct is used to create a branch in a program. The computer selects which branch to follow based on the outcome of a condition, using an `IF…THEN…ELSE` statement. For example, in pseudocode:

```
IF day = 'Saturday' OR day = 'Sunday' THEN
  SET alarm TO 11
ELSE
  SET alarm TO 8
END IF
```

SUBJECT VOCABULARY

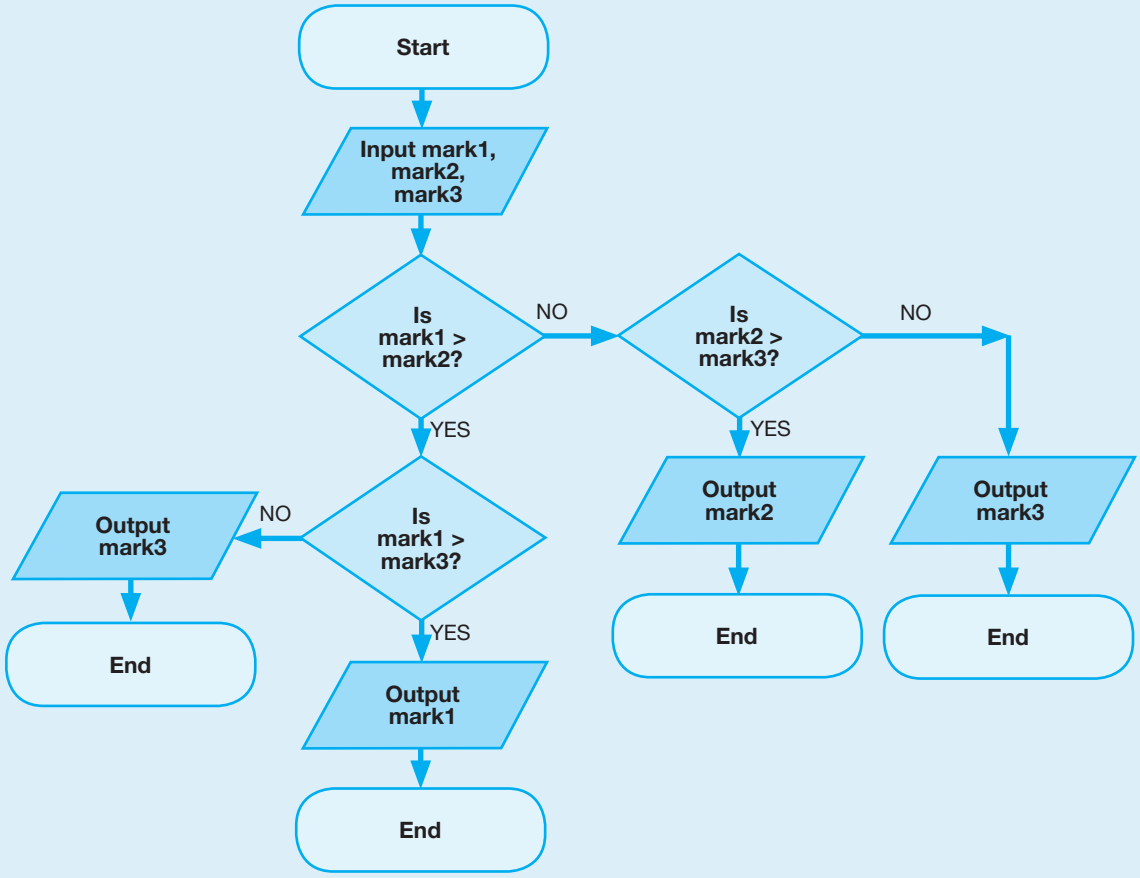
nested IF statement a nested IF statement consists of one or more IF statements placed inside each other. A nested IF is used where there are more than two possible courses of action

A standard `IF…THEN…ELSE` statement provides two alternatives. If there are more than two, then in Pearson Edexcel pseudocode a nested `IF` must be used. However, many high-level programming languages have an additional built-in selection construct that does away with the need for a **nested IF statement**.

WORKED EXAMPLE

A learner handed in three homework assignments, which were each given a mark out of 10. All the marks were different. Write an algorithm that would print out the highest mark.

Figure 2.1 shows the algorithm expressed as a flowchart.



▲ **Figure 2.1** Flowchart of an algorithm to print out the highest homework mark

HINT

When you are creating nested `IF` statements, you have to ensure that each one is completed with an `END IF` statement at the correct indentation level. Some programming languages do not need an `END IF` statement and just use the indentation levels to indicate when statements are grouped.

In Pearson Edexcel pseudocode, this algorithm could be expressed as:

```
RECEIVE mark1 FROM KEYBOARD
RECEIVE mark2 FROM KEYBOARD
RECEIVE mark3 FROM KEYBOARD
IF mark1 > mark2 AND mark1 > mark3 THEN
    SEND mark1 TO DISPLAY
ELSE
    IF mark2 > mark1 AND mark2 > mark3 THEN
        SEND mark2 TO DISPLAY
    ELSE
        IF mark3 > mark1 AND mark3 > mark2 THEN
            Send mark3 TO DISPLAY
        END IF
    END IF
END IF
```

In Python, Java and C# this does not have to be done as they have an 'else if' statement.

In Python the 'else if' statement is `elif` and the algorithm above could be:

```
mark1 = input('Please enter the first mark')
mark2 = input('Please enter the second mark')
mark3 = input('Please enter the third mark')

if mark1 > mark2 and mark1 > mark3:
    print(mark1)
elif mark2 > mark1 and mark2 > mark3:
    print(mark2)
elif mark3 > mark1 and mark3 > mark2:
    print(mark3)
```

Here is the algorithm in Java:

```
import java.util.*;
class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Please enter the first mark: ");
    int mark1 = scanner.nextInt();
    System.out.print("Please enter the second mark: ");
    int mark2 = scanner.nextInt();
    System.out.print("Please enter the third mark: ");
    int mark3 = scanner.nextInt();
    scanner.close();

    if (mark1 > mark2 && mark1 > mark3) {
      System.out.print(mark1);
    } else if (mark2 > mark1 && mark2 > mark3) {
      System.out.print(mark2);
    } else if (mark3 > mark1 && mark3 > mark2) {
      System.out.print(mark3);
    }
  }
}
```

And in C#:

```
string mark1String, mark2String, mark3String;
int mark1, mark2, mark3;

Console.WriteLine("Please enter the first mark");
mark1String = Console.ReadLine();    // read input into string
mark1 = int.Parse(mark1String);      // convert string to integer

Console.WriteLine("Please enter the second mark");
mark2String = Console.ReadLine();    // read input into string
mark2 = int.Parse(mark2String);      // convert string to integer
```

```
Console.WriteLine("Please enter the third mark");
mark3String = Console.ReadLine();    // read input into string
mark3 = int.Parse(mark3String);      // convert string to integer

if (mark1 > mark2 && mark1 > mark3)
{
    Console.WriteLine(mark1);
}
else if (mark2 > mark1  && mark2 > mark3)
{
   Console.WriteLine(mark2);
}
else if (mark3 > mark1 && mark3 > mark2)
{
   Console.WriteLine(mark3);
}
```

These examples use **relational** and **logical operators**.

SUBJECT VOCABULARY

relational operator an operator that tests the relationship between two entities

logical operator a Boolean operator using AND, OR and NOT

RELATIONAL OPERATORS

The relational operators are used to compare two values and in Python, Java and C# are all the same.

RELATIONAL OPERATOR	PYTHON, JAVA, C#
Equal to	==
Greater than	>
Greater than or equal to	>=
Less than	<
Less than or equal to	<=
Not equal to	!=

ACTIVITY 4

Look at the following algorithm and answer the questions.

```
IF score <= highScore THEN
    SEND 'You haven't beaten your high score.' TO DISPLAY
ELSE
    SEND 'You've exceeded your high score!' TO DISPLAY
END IF
```

What is the output of the algorithm when

- `score` = 5 and `highScore` = 10?
- `score` = 20 and `highScore` = 10?
- `score` = 15 and `highScore` = 15?

LOGICAL OPERATORS

AND

If two conditions are joined by the `AND` operator, then they must both be true for the whole statement to be true.

OR

If two conditions are joined by the `OR` operator, then either one must be true for the whole statement to be true.

NOT

The `NOT` operator reverses the logic of the `AND` and `OR` statements. The statement `IF A = 3 AND B = 6` will be true only if the conditions are met, i.e. A and B are both equal to the values stated.

The statement `IF NOT (A = 3 AND B = 6)` will be true whenever both A and B are NOT equal to the values stated, i.e. either or both are not equal to those values.

Logical operators in high-level languages.

LOGICAL OPERATOR	PYTHON	JAVA	C#
AND	and	&	&&
OR	or	|	||
NOT	not	!	!

ACTIVITY 5

A driving school uses this rule to estimate how many lessons a learner will require.

- Every learner requires at least 20 lessons.
- Learners over the age of 18 require more lessons (two additional lessons for each year over 18).

Create a program in a high-level language that inputs a learner's age and calculates the number of driving lessons they will need.

LOOPS

A loop is another name for an iteration. Loops are used to make a computer repeat a set of instructions more than once. There are two types of loop: definite and indefinite.

A definite loop is used when you know in advance how often the instructions in the body of the loop are to be repeated. For example, if you want the computer to display a character on the screen for a fixed amount of time and then remove it.

An indefinite loop is used when the number of times a loop will need to be repeated is not known in advance. For example, if you want to give a user the option of playing a game as often as they want. Indefinite loops are repeated until a specified condition is reached.

Every programming language has a number of built-in loop constructs. You will need to explore the ones provided in the language you are studying.

DEFINITE ITERATION

This is used when the number of iterations, or turns of the loop, is known in advance.

In the Pearson Edexcel pseudocode there are two ways of doing this using `REPEAT…END REPEAT` and `FOR…END FOR`.

An example of a `REPEAT` loop is shown below.

```
REPEAT 50 TIMES
  SEND '*' TO DISPLAY
END REPEAT
```

Using a `FOR` loop, this would be:

```
FOR times FROM 1 TO 100 DO
    SEND '*' TO DISPLAY
END FOR
```

`FOR` loops can also include a step so that the counting is not consecutive. A step is included in the following pseudocode example.

```
FOR times FROM 0 TO 100 STEP 25 DO
    SEND times TO DISPLAY
END FOR
```

The output would be 0, 25, 50, 75.

USE	EXAMPLE	RESULT
Using the 'range' command	`for x in range(6):` `  print(x)`	0 1 2 3 4 5
Stipulating a start number so that it does not begin at 0	`for x in range(2,6):` `  print(x)`	2 3 4 5

Using a step by adding a third number into the brackets	`for x in range(0,100,25):`
	`  print(x)`
	
	To also print 100 you would use:
	`for x in range(0,101,25):`
		0
		25
		50
		75

Printing items from a list	`aList = ['red', 'blue', 'green', 'yellow', 'purple', 'orange']`
	`for colour in aList:`
	`  print (colour)`
	
	The number of iterations does not have to be given as the length of the list is used.
		red
		blue
		green
		yellow
		purple
		orange

(continued)

(continued)

USE	EXAMPLE	RESULT

Leaving out items from the list	`aList = ['red', 'blue', 'green', 'yellow', 'purple', 'orange']`
	`for colour in aList:`
	`  if colour == 'yellow':`
	`  continue`
	`  print (colour)`
		red
		blue
		green
		purple
		orange

▲ **Table 2.2** Python

USE	EXAMPLE	RESULT

Using a `for` loop	`for(int x = 0; x < 6; x++) {`
	`  System.out.println(x);`
	`}`
		0
		1
		2
		3
		4
		5

Stipulating a start number so that it does not begin at 0	`for(int x = 2; x < 6; x++) {`
	`  System.out.println(x);`
	`}`
		2
		3
		4
		5

Using a step by adding an amount to increase by	`for(int x = 0; x < 100; x+=25) {`
	`  System.out.println(x);`
	`}`
	To also print 100 you would use:
	`for(int x = 0; x <= 100; x+=25) {`
	`  System.out.println(x);`
	`}`
		0
		25
		50
		75

Printing items from a list	`String[] aList = {"red", "blue", "green", "yellow", "purple", "orange"};`
	`for(String colour: aList) {`
	`  System.out.println(colour);`
	`}`
		red
		blue
		green
		yellow
		purple
		orange

Leaving out items from the list	`String[] aList = {"red", "blue", "green", "yellow", "purple", "orange"};`
	`for(String colour: aList) {`
	`  if(colour == "yellow") {`
	`  continue;`
	`  }`
	`  System.out.println(colour);`
	`}`
		red
		blue
		green
		purple
		orange

▲ **Table 2.3** Java

USE	EXAMPLE	RESULT

Using a `for` loop	`for (int i = 0; i <= 5; i++)`
	`{`
	`  Console.WriteLine(i);`
	`}`
		0
		1
		2
		3
		4
		5

Stipulating a start number so that it does not begin at 0	`for (int i = 2; i <= 5; i++)`
	`{`
	`  Console.WriteLine(i);`
	`}`
		2
		3
		4
		5

Using a step by altering the increment value of a `for` loop	`for (int i = 0; i < 100; i+= 25)`
	`{`
	`  Console.WriteLine(i);`
	`}`
		0
		25
		50
		75

Printing items from an array*	`string[] colours = {“red”, “blue”, “green”, “yellow”, “purple”, “orange”};`
	
	`foreach (string i in colours)`
	`{`
	`  Console.WriteLine(i);`
	`}`
	
	The number of iterations does not have to be given as the length of the array is used.
		red
		blue
		green
		yellow
		purple
		orange

Leaving out items from the array*	`string[] colours = {“red”, “blue”, “green”, “yellow”, “purple”, “orange”};`
	
	`foreach (string i in colours)`
	`{`
	`  if (i == “yellow”)`
	`  {`
	`  continue;`
	`  }`
	`  else`
	`  {`
	`  Console.WriteLine(i);`
	`  }`
		red
		blue
		green
		purple
		orange

*See pages 62–68 for more information on arrays.

▲ **Table 2.4** C#

SKILLS PROBLEM SOLVING

ACTIVITY 6

Produce a program in a high-level language that asks a user to enter a start number and an end number and then outputs the total of all the numbers in the range. For example, if the start number was 1 and the end number was 10, the total would be 55 (10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1).

NESTED LOOPS

A **nested loop** is made of a loop within a loop. When one loop is nested within another, each iteration of the outer loop causes the inner loop to be executed until completion.

ACTIVITY 7

HINT

You should initialise the variable total to zero before the start of the loop.

1 Python

Look at the following program and then answer the questions below.

```
for student in range(1, 21):
    sum = 0
    for mark in range(1, 6):
        nextMark = int(input('Please enter a mark'))
        sum = sum + nextMark
    averageMark = sum/5
    print(averageMark)
```

a What is the purpose of this program?

b Why is `int` used in the line `nextMark = int(input('Please enter a mark'))`?

2 Java

Look at the following program and then answer the questions below.

```
import java.util.*;
class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    for(int student = 1; student < 21; student++) {
      int sum = 0;
      for(int mark = 1; mark < 6; mark++) {
        System.out.print("Please enter a mark: ");
        int nextMark = scanner.nextInt();
        sum = sum + nextMark;
      }
      double averageMark = sum / 5;
      System.out.println(averageMark);
    }
    scanner.close();
  }
}
```

a What is the purpose of this program?

3 C#

```
int nextMark, sum, averageMark;
string markString;
sum = 0;
for (int mark = 1; mark <= 5; mark++)
{
    Console.WriteLine("Please enter a mark");
    markString = Console.ReadLine();
    nextMark = int.Parse(markString);
            sum = sum + nextMark;
      }

      averageMark = sum / 5;
      Console.WriteLine(averageMark);
```

a What is the purpose of the program?

b Why is `int.Parse` used in the line `nextMark = int.Parse(markString);`?

SKILLS PROBLEM SOLVING

ACTIVITY 8

Produce an algorithm that will print out the times tables (up to 12 times) for the numbers 2 to 12.

INDEFINITE ITERATION

An indefinite loop is used when the number of times a loop will need to be repeated is not known in advance. For example, if you want to give a user the option of playing a game as often as they want. Indefinite loops are repeated until a specified condition is reached.

Python

For indefinite iteration, Python uses the 'while' loop – something is done while a condition is met.

The following program asks a user to enter a number while the number is less than 20.

```
number = 1
while number <= 20:
    number = int(input('Please enter a number'))
print('You entered a number greater than 20')
```

As soon as the number is greater than 20, the program breaks out of the loop and prints a message for the user.

Java

For indefinite iteration, Java uses a 'while' loop – instructions are repeated while a condition is met.

The following program asks a user to enter a number while the number is less than 20.

```
import java.util.*;
class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int number = 1;
    while(number <= 20) {
      System.out.print("Please enter a number: ")
      number = scanner.nextInt();
    }
    System.out.println("You entered a number greater than 20")
    scanner.close();
  }
}
```

C#

```
string numberString;
int number = 1;
while (number <=20)
{
    Console.WriteLine("Please enter a number");
    numberString = Console.ReadLine();
    number = int.Parse(numberString);
}

Console.WriteLine("You entered a number greater than 20");
```

As soon as the number is greater than 20, the program breaks out of the loop and prints a message for the user.

ACTIVITY 9

What do these two algorithms do? Implement them in the high-level programming language you are studying.

a Algorithm A

```
FOR index FROM 1 TO 10 DO
   SEND index * index * index TO DISPLAY
END FOR
```

b Algorithm B

```
SET counter TO 10
WHILE counter > 0 DO
```

```
    SEND counter TO DISPLAY
    SET counter TO counter - 1
END WHILE
```

RANDOM NUMBERS

Random numbers are commonly used in games of chance such as flipping a coin or rolling a dice. The aim is to make an event random.

All high-level programming languages have functions to create random numbers.

Python

Python has a random module. The following code will generate a random number between 1 and 10.

```
import random
x = random.randint(1, 11)
print(x)
```

The 'import' command is necessary so that the 'random' module can be used.

Java

Java can generate a random integer using `System.util.Random`. The following code will generate a random number between 1 and 10.

```
import java.util.*;
class Main {
  public static void main(String[] args) {
    Random r = new Random();
    int x = r.nextInt(10) + 1;
    System.out.println(x);
  }
}
```

Alternatively, you can generate a random real value using `Math.random()`.

```
class Main {
  public static void main(String[] args) {
    int x = (int)(Math.random() * 10) + 1;
    System.out.println(x);
  }
}
```

C#

C# can generate random numbers using the Random Class. The following code will generate a random number between 1 and 10 and display on screen.

```
Random rand = new Random();
randomNumber = rand.Next(1, 11);
Console.WriteLine(randomNumber);
```

ACTIVITY 10

Create a guessing-game program with the following specification.

- The computer generates a random number between 1 and 20.
- The user is asked to enter a number until they enter this random number.
- If their guess is too low or too high they are told.
- They are told when their guess is correct.
- They are asked if they want to play another game until their answer is 'NO'.

CHECKPOINT

Strengthen

SKILLS CRITICAL THINKING

S1 Why are variables needed?

SKILLS REASONING

S2 Provide examples of the four data types.

SKILLS CRITICAL THINKING

S3 How are selection and iteration implemented in the high-level language you are studying?

Challenge

SKILLS PROBLEM SOLVING

C1 Outline the following structural components of a program: variable and type declarations, command sequences, selection and iteration constructs.

How confident do you feel about your answers to these questions? If you're not sure you answered them well, try redoing the activities in this section.

SUMMARY

- A program is an algorithm that has been converted into program code.
- Pseudocode is far more forgiving than program code.
- The four basic data types are integer, float/real, Boolean and character.
- The data type of a variable determines the operations that can be performed on it.
- Data types don't have to be declared in pseudocode but it's a good idea to do so.
- Variable and type declarations, command sequences, selection and iteration are four of the structural components of a program.

6 MAKING PROGRAMS EASY TO READ

Developers usually work in teams and it is important that they understand how each other's code and programs work, especially if there are errors that need correcting. Code should be written in standard ways and be explained using comments.

LEARNING OBJECTIVES

- Explain the benefit of producing programs that are easy to read
- Use techniques to improve the readability of code and describe how code works

CODE READABILITY

You should always try to ensure that any code you write is easy to read and understand. We refer to this as 'readability'. This benefits you and anyone else who needs to understand how your programs work.

It is surprising how quickly you forget. Try revisiting the programs you have already written in this unit and make sure it is still clear to you what they do and how they work. Imagine how much more difficult it would be to make sense of a complex program with lots of variables, subprograms, nested loops and multiple selection statements.

Programming is not a solo activity. Programmers usually work in teams, with each programmer developing a different part of the program. This only works if they all adopt a standard approach to writing readable code.

▶ **Figure 2.2** An example of Python code

```
Python 3.4.1: example2.py - C:/Python/example2.py
File  Edit  Format  Run  Options  Windows  Help
import random
game=True
while game==True:
    Num1=int(input("Choose one of the following: 1 Rock, 2 Paper, or 3 Scissors:"))
    Num2=random.randint(1,3)
    if Num2==1:
        print("Computer chooses Rock")
    elif Num2==2:
        print("Computer chooses Paper")
    else:
        print("Computer chooses Scissors")
    if Num1==Num2:
        print("It's a draw")
    elif Num1==1 and Num2==3:
        print("You win!")
    elif Num1==2 and Num2==1:
        print("You win!")
    elif Num1==3 and Num2==2:
        print("You win!")
    else:
        print("Computer wins")
    text=input("Play again? Y/N:")
    if text!="Y":
        game=False
                                                                    Ln: 26  Col: 0
```

GENERAL VOCABULARY

practice the way that you do something

SUBJECT VOCABULARY

block of code a grouping of two or more code statements

The programmer who produced the program in Figure 2.2 did not follow good **practice**. There are a number of ways that the readability of this code could be improved.

- Use descriptive names for variables (e.g. `userChoice` instead of `Num1`).
- Add blank lines between different **blocks of code** to make them stand out.
- Add comments that explain what each part of the code does.