



2.2.1 PROGRAMMING FUNDAMENTALS

The use of variables, constants, operators, inputs, outputs and assignments

The use of the three basic programming constructs used to control the flow of a program:

- ☐ Sequence
- ☐ Selection
- ☐ Iteration (count- and condition-controlled loops)

The common arithmetic operators

The common Boolean operators AND, OR and NOT

There are three main “constructs” used in high level language programming – **SEQUENCING**, **SELECTION** and **ITERATION**. **SEQUENCING** involves a block of code that executes line after line (in sequence) :

```
print("Good morning")
name = input("What is your name?")
print("Hello",name)
age = int(input("How old are you?"))
print(age,"is a very good age!")
```

SELECTION involves the use of **IF** statements to evaluate the contents of a variable - program will execute different code depending on the value of the variable

```
question = input("Do you enjoy programming?")
if answer == "yes":
    print("Awesome!")
```

ITERATION is used to repeat (loop) a block of code. This is a more efficient way of programming then to add the same code multiple times. There are two types of iteration; a **count controlled loop** runs a block of code a SET number of times;

```
for count in range (1,10):
    print("I have counted to", count)
```

a **condition controlled loop** runs a block of code until a specific condition is met – for example, a program could ask for a password until it is entered correctly.

```
correct = False
while correct == False
    password = input("Enter your password")
    if password == correctpassword:
        correct = True
```

2.2.2 DATA TYPES

The use of data types:

- ☐ Integer
- ☐ Real
- ☐ Boolean
- ☐ Character and string
- ☐ Casting

Constants and variables can be stored as a range of **DATA TYPES**. It is also possible to use **CASTING** to convert data from one type to another:

```
NumberString = "42"
Number = int(NumberString)
pi = 3.141
pi = int(pi)
print(pi)
3 >_
```

FUNDAMENTALS

A **VARIABLE** is a memory location used to store data. Programmers can label a variable using an **IDENTIFIER**. The contents of the memory location (and the value of the variable) can be changed by the programmer.

Giving a value to a variable is called **ASSIGNING**. Variables can be assigned a value directly by the programmer or **INPUT** by the user when running the program.

A print statement can be used to **OUTPUT** data – a print statement can be used to display specific text or the contents of a variable.

```
name = input ("Please enter your name")
print ("Hello,"name)
```

CONSTANTS are similar in principle to variables, but their value does not change throughout the program

```
const Pi = 3.142
```

MATHEMATICAL OPERATORS allow calculations to be performed using variables and constants

+	Addition
-	Subtraction
/	Division
*	Multiplication
DIV	Integer division
MOD	Modulus (remainder)
^	Exponent

BOOLEAN OPERATORS are used when making logical comparisons (i.e. when using IF statements)

NOT	Addition
AND	Subtraction
OR	Division
!=	Not equal to
==	Equal
<	Less than
>	Greater than
<=	Less or = to
>=	Greater than or = to

DATA TYPE	EXPLANATION	EXAMPLE
Integer	Whole number	HIGH_SCORE = 100000 RANK = 10
Float/ Real	A "fractional" number	PI = 3.141 TEMPERATURE = 21.5
Character	A single character (letter, number, symbol)	INITIAL = "J" GRADE = "A"
String	Zero or more characters	NAME = "Arthur Dent" PASSWORD = "FISH42*"
Boolean	Can be either TRUE or FALSE	PERMISSION = True CORRECT = False

2.2.3 ADDITIONAL PROGRAMMING TECHNIQUES

The use of basic string manipulation

The use of basic file handling operations:

- ☐ Open
- ☐ Read
- ☐ Write
- ☐ Close

The use of records to store data

The use of SQL to search for data

The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays

How to use sub programs (functions and procedures) to produce structured code

Random number generation

Data can be imported to/exported from programs using **FILES**. This means that a program can keep its data, even when it is closed and reopened. A range of **FILE HANDLING OPERATIONS** are possible..

open	Prepares the file ready for use
close	Close access to the file when it is no longer needed
read	Retrieve data from the file
write	Overwrite the file with new data
append	Save new data onto the end of the file

STRING MANIPULATION

Many programming languages (including Python) have built-in functions allow programmers to manipulate strings.

Description	Example	Result
Length	<code>length = len(name)</code>	17
Convert to upper case	<code>capitals = name.upper()</code>	ZAPHOD BEEBLEBROX
Convert to lower case	<code>small = name.lower()</code>	zaphod beebblebrox
Return a substring	<code>name.substring(0,2)</code>	Zap

There are a wide number of ways in which strings can be manipulated – a few are examples are given in the table for this example: `name = "Zaphod Beeblebrox"`

DATABASES are used to organise and structure data. In a database, data is stored in on a table – each row holds a **RECORD** and each column (**FIELD**) refers to different aspect of the data. **SQL (STRUCTURED QUERY LANGUAGE)** is a language used to build, edit and interrogate databases.

```
SELECT Pupil_ID, FirstName, Surname
FROM Pupil
WHERE grade > 7
```

REVISION NOTE

Python does not use **ARRAYS** – make sure you are clear about how they differ from **LISTS**

PUPIL

Pupil_ID	First Name	Surname	Mentor	Mark	Grade
1012	Ford	Prefect	HG5	80	8
0981	Tricia	McMillan	HG7	95	9
1422	Arthur	Dent	HG1	55	6

1012 Ford Prefect
0981 Tricia McMillan

LIST ARRAY 2 Dimensional Array

	LIST	ARRAY	2 Dimensional Array
Data Structure	✓	✓	✓
Can contain mixed data types	✓	✗	✗
Size can be changed after it has been defined	✓	✗	✗
Arranges data in row and columns	✗	✗	✓

While variables store individual pieces of data **ARRAYS** are data structures which store related items of data.

```
PupilName = {"Ford",
"Tricia", "Arthur"}
NameAndMark = {"Ford", 80,
"Tricia", 95, "Arthur", 6}
ClassTests [20,10]
```

SUBPROGRAMS

are "programs within programs" and perform a specific function within a larger program. Using subprograms allows larger programs to be broken down into smaller parts making them easier to design, test and understand.

Description	PROCEDURE	FUNCTION
Example of a subprogram	✓	✓
Needs to be called from the main program	✓	✓
Can have parameters passed into it	✓	✓
Can return values back out to the main program	✗	✓

Programming languages have built-in functions that can be used to generate "RANDOM" numbers. `dice_roll = random (1,6)`