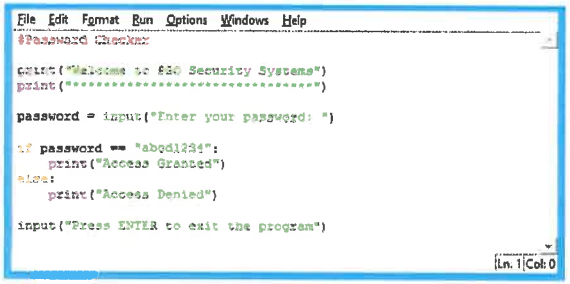
Year 9 Python Knowledge Organiser

Programming with Python



Python's Development Environment

Called IDLE – Integrated Development Environment

Two Modes:
Interactive Mode lets you see your results as you type them.
Script Mode lets you save your program and run it again later.

Writing error-free code

When writing **programs**, **code** should be as legible and error free as possible. **Debugging** helps keep **code** free of **errors** and documenting helps keep **code** clear enough to read.

Syntax errors

Syntax is the spelling and grammar of a **programming language**. In **programming**, a **syntax error** occurs when:

- there is a **spelling mistake.**
- there is a **grammatical mistake.**

Data Types

String - holds alphanumeric data as text
Integer - holds whole numbers
Float - holds numbers with a decimal point
Boolean - holds either 'True' or 'False'

Defining Variable Data Types

Python automatically assigns a data type to a variable. You can override this to manually define or change the data type using:

str() , **int()** or **float()**

Selection

When designing **programs**, there are often points where a **decision** must be made. This **decision** is known as **selection** and is implemented in **programming** using **IF statements**.

Operator	Meaning	Example	Evaluates to
==	equal to	7==7	True
!=	not equal to	6!=7	True
>	Greater than	7>6	True
<	Less than	5<8	True
>=	Greater than or equal to	6>=8	False
<=	Less than or equal to	7<=7	True

Procedures

A **procedure** is a small section of a **program** that performs a specific task. **Procedures** can be used repeatedly throughout a **program**. **Procedures** can make **code** shorter, simpler, and easier to write. Writing a **procedure** is extremely simple. Every **procedure** needs:

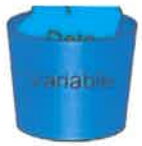
1. *A name*
2. *The **program** code to perform the task*

Variables

A **variable** is a location in **memory** in which you can temporarily store text or numbers. It is used like an empty box or the Memory function on a calculator. You can choose a name for the box (the "**variable name**") and change its contents in your **program.**

Using a Variable (firstname)

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



Functions

Functions are special keywords that do a specific job. **Functions** appear in purple.

print() and **input()** are examples of functions

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

Adding Comments

Comments are useful to help understand your **code.** They will not affect the way a **program** runs. **Comments** appear in red and have a preceding # symbol.

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

Year 9 Python Knowledge Organiser

Iteration

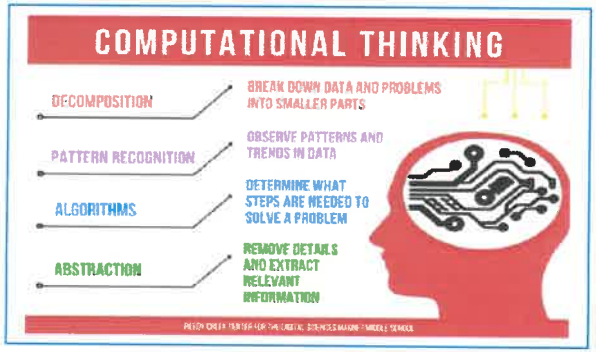
Algorithms consist of steps that are carried out (performed) one after another. Sometimes an **algorithm** needs to **repeat** certain steps until told to stop or until a particular condition has been met. ***Iteration is the process of repeating steps.***

Iteration allows us to **simplify** our **algorithm** by stating that we will **repeat** certain **steps** until told otherwise. **Iteration** is implemented in **programming** using **FOR** and **WHILE** statements.

There are **two** ways in which **programs** can **iterate** or **'loop'**:

- count-controlled loops
  - Sometimes it is necessary for **steps** to **iterate** a **specific number** of times.
- condition-controlled loops
  - **iteration** continues **while**, or **until**, a **condition** is met.

Each type of **loop** works in a slightly different way and produces different results.

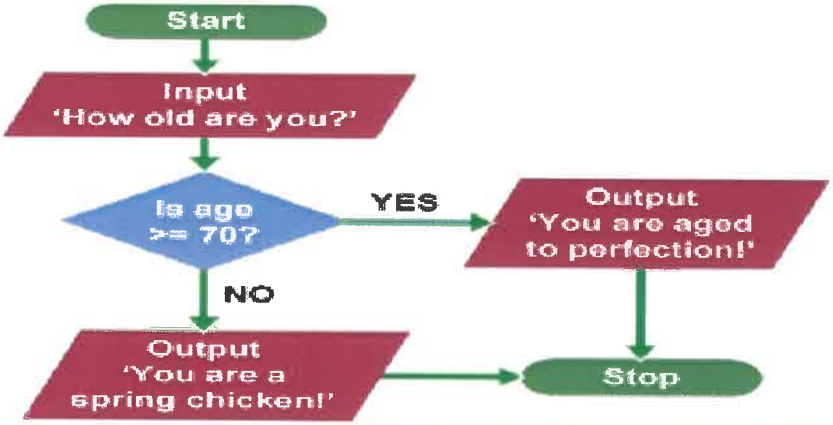


IF Statements

When designing **programs**, there are often points where a **decision** must be made. This **decision** is known as **selection** and is implemented in **programming** using **IF** statements. In **programming**, **selection** is usually represented by the statements **IF** and **ELSE**.
For **selection**, Python uses the statements **if** and **else** (note the lowercase **syntax** that **Python** uses):

Consider the age-related **algorithm** using **Python**. The steps are:

- Ask how old you are
- if you are 70 or older, say "You are aged to perfection!"
- else say "You are a spring chicken!"



The above algorithm would be written in Python (3.x) as:

```
age = int(input("How old are you?"))

if age >= 70:

        print("You are aged to perfection!")

else:

        print("You are a spring chicken!")
```

Arrays

An **array** is a series of **memory** locations – or **'boxes'** – each of which holds a single item of **data**, but with each box sharing the same name. All **data** in an **array** must be of the same **data type**.

Arrays are named like **variables**. The number in brackets determines how many **data** items the **array** can hold. The array **score(9)** would allow ten data items to be stored.



Any **facility** that holds more than one item of **data** is known as a **data structure**. Therefore, an **array** is a **data structure**.

Lists are **data structures** similar to **arrays** that allow **data** of more than one **data type**.

Functions

A **function** is also a small section of a **program** that performs a specific task that can be used repeatedly throughout a **program**, but the task is usually a **calculation**. **Functions** perform the task and return a value to the main **program**.

Every **function** needs:

1. A name
2. The values that it needs to use for calculation
3. The **program** code to perform the task
4. A value to return to the main program

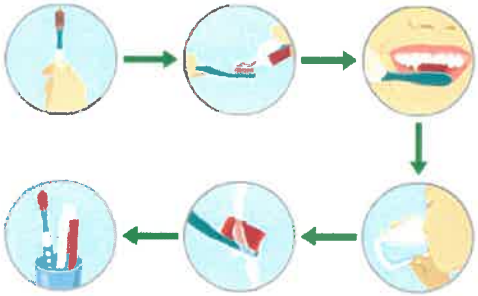
Knowledge Organiser: Programming — Procedures and Functions

Procedures and Functions

Summary—When writing programs, we should avoid long, repetitive code. Procedures and functions help to keep our programs simple and short.

A **procedure** is a small section of a program that performs a specific task. Procedures can be used repeatedly throughout a program.

A **function** is also a small section of a program that performs a specific task that can be used repeatedly throughout a program, but the task is usually a calculation. Functions perform the task and return a value to the main program.



What is a Procedure

A **procedure** is a small section of a program that performs a specific task. Procedures can be used repeatedly throughout a program.

Why use a Procedure

Program code is easier to read and understand when it is broken up into smaller sections. By breaking a program up into these sections, or procedures, code can be made shorter and simpler.

Writing a Procedure

Writing a **procedure** is extremely simple. Every procedure needs:
a name

the **program** code to perform the task

Running a Procedure

Once a **procedure** is named and written, it can be **called** at any point in the **program**.

To call a procedure in **Python**, simply use its name (include the brackets)

What is a Function

A **function** is also a small section of a program that performs a specific task that can be used repeatedly throughout a program, but the task is usually a calculation. Functions perform the task and return a value to the main program.

Both functions and procedures are small sections of code that can be repeated through a program. The difference between them is that functions return a value to the program where procedures perform a specific task.

Why do we use Functions

Program code is easier to read and understand when it is broken up into smaller sections. Functions and procedures can make code shorter, simpler and easier to write.

A function is just like a procedure, except that you wait to see what the result is.

Running a Function

Once a **function** is named and written, it can be **called** at any point in the **program**. 'Calling' a function means running it.

To call a function in **Python**, simply use its name (include the brackets) and the value it needs to use for calculation, plus a variable that the returned value will be stored in.

Key Vocabulary

Algorithm	A sequence of logical instructions for carrying out a task. In computing, algorithms are needed to design computer programs.
Calling	Starting or running a function or procedure.
Code	Instructions in a computer program.
Debug	The process of finding and correcting programming errors.
Function	A section of code that, when programming, can be called by another part of the program with the purpose of returning one single value.
Procedure	A section of computer code that performs a specific task.
Program	Sequences of instructions for a computer.
Python	A high-level programming language.
Statement	The smallest element of a programming language which expresses an action to be carried out.



Some images © 2016 BBC

Knowledge Organiser: Programming — Writing Error Free Code

Writing Error Free Code

Summary
When writing programs, code should be as legible and error free as possible. **Debugging** helps keep code free of errors and documenting helps keep code clear enough to read.

When programs are written, it is likely that at least some errors will creep in. Errors in programs are often referred to as **bugs**.

It is vital that programs are as free of errors as possible. Errors can cause a program to produce unexpected results, or **crash**.



Syntax Error

Syntax errors will cause a program to crash or not run at all. The **program** may run until it encounters a syntax

Types of Syntax Error

More than one type of syntax error may exist. There may be:
incorrectly spelled **statements**
incorrectly spelled **variables**
missing punctuation (quotes, brackets, etc)

Any one or more of these errors may exist in a program, and each will cause the program to crash or not run at all.

Fixing syntax errors is simple. Spotting them is more difficult. Some of the most common syntax errors are **things that come in pairs**. Always check for speech marks and brackets first if your program

Fixing syntax errors is simple.
Spotting them is more difficult.

Logic Errors

Logic errors occur when there is a **fault** in the **logic** or **structure** of the problem. Logic errors do not usually cause a program to **crash.** However, logic errors can cause a program to produce **unexpected results.**

Types of Logic Error

More than one type of logic error may exist. Parts of the program may:
be in the wrong **sequence**
have the wrong **Boolean** expression
use the wrong **data type**
be missing altogether

Any one or more of these errors may exist in a pro-

Documenting Code

Code that is hard to read makes it difficult to understand what the **program** is trying to do. It also makes it difficult to understand the purpose of
any **variables, procedures** and **functions**. Errors are easier to fix when we understand the code we are reading through.

When we **document** code, we make it easier to read and understand. We also document code in case:
we need to come back to it at a later date

Key Vocabulary

Boolean	A data type in computing which only has two possible values, true or false.
Bug	An error in a program.
Code	Instructions in a computer program.
Commenting	Adding one or more sentences to explain the purpose of a section of code.
Crash	When an application or operating system no longer responds. Crashes can be caused by software and hardware.
Data Type	The format in which a variable or constant holds data, such as 'integer' or 'string'.
Debug	The process of finding and correcting programming errors.
Documenting	Making code easier to understand by giving meaningful names for variables, procedures and functions and by adding comments to explain the purpose of instructions.
Execute	To run a computer program.
Function	A section of code that, when programming, can be called by another part of the program with the purpose of returning one single value.
Instruction	A single action that can be performed by a computer processor.
Logic error	A fault in the logic or structure of the problem.
Procedure	A section of computer code that performs a specific task.
Program	Sequences of instructions for a computer.
Programming Language	A language used by a programmer to write a piece of software.
Python	A high-level programming language.
Sequence	In computer programming, this is a set of instructions that follow on one from another.
Statement	The smallest element of a programming language which expresses an action to be carried out.

Some images is copyright © 2016 BBC.

Knowledge Organiser: Representing Text Images and Sound

Representing Data

All data inside a computer is transmitted as a series of electrical signals that are either **on** or **off**. Therefore, in order for a computer to be able to process any kind of data, including text, images and sound, they must be converted into **binary** form. If the data is not converted into binary – a series of 1s and 0s – the computer will simply not understand it or be able to process it.

Representing Text

A code where each number represents a character can be used to convert text into binary. One code we can use for this is called **ASCII**.

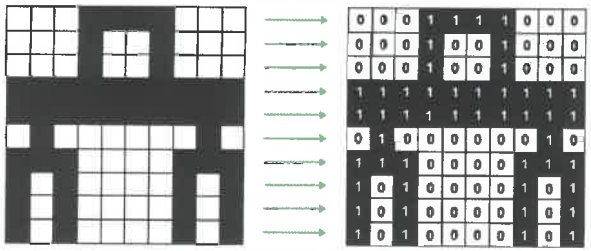
ASCII code can only store 128 characters, which is enough for most words in English but not enough for other languages.
If you want to use accents in European languages or larger alphabets such as Cyrillic (the Russian alphabet) and Chinese Mandarin then more characters are needed.

Therefore another code, called **Unicode**, was created. This meant that computers could be used by people using

Representing Images—Black and White

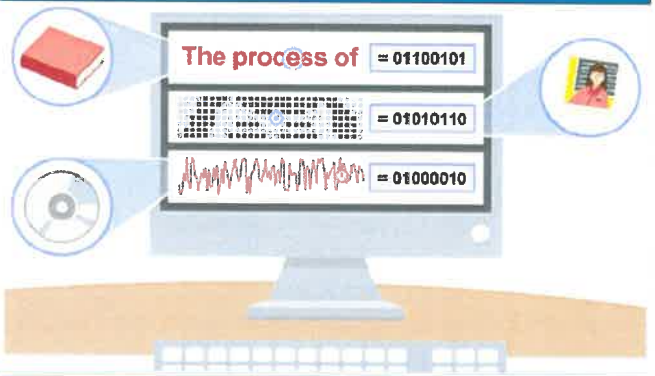
Images also need to be converted into **binary** in order for a computer to process them so that they can be seen on our screen. Digital images are made up of **pixels**. Each pixel in an image is made up of binary numbers.

If we say that 1 is black (or on) and 0 is white (or off), then a simple black and white picture can be created using binary.



Some images is copyright of their own.

Text Images and Sound



Representing Images—Colour

Adding another binary digit will double the number of colours that are available:

- 1 bit per pixel (0 or 1): two possible colours
- 2 bits per pixel (00 to 11): four possible colours
- 3 bits per pixel (000 to 111): eight possible colours
- 4 bits per pixel (0000 – 1111): 16 possible colours
- ...

16 bits per pixel (0000 0000 0000 0000 – 1111 1111 1111 1111): over 65 000 possible colours

The number of bits used to store each pixel is called the **colour depth**. Images with more colours need more pixels to store each available colour. This means that images that use lots of colours are stored in larger files.

Image Quality

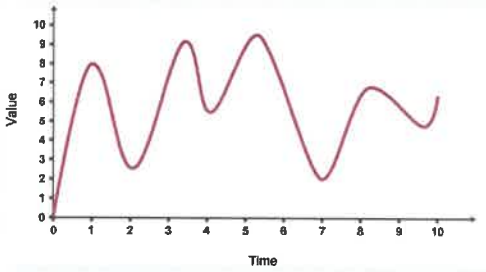
Image quality is affected by the **resolution** of the image. The resolution of an image is a way of describing how tightly packed the pixels are.

In a low-resolution image, the pixels are larger so fewer are needed to fill the space. This results in images that look blocky or **pixelated**. An image with a high resolution has more pixels, so it looks a lot better when you zoom in or stretch it. The downside of having more pixels is that the file size will be bigger.

Representing Sound

Sound needs to be converted into **binary** for computers to be able to process it. To do this, sound is captured - usually by a microphone - and then converted into a **digital** signal.

An **analogue** to digital converter will sample a sound wave at regular time intervals. For example, a sound wave like this can be sampled at each time sample point:

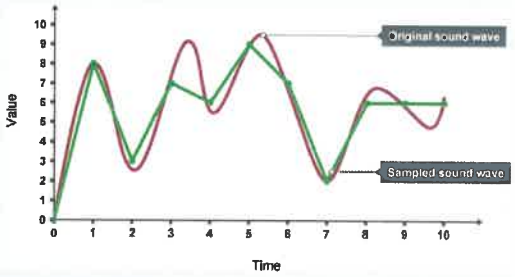


The samples can then be converted to binary. They will be recorded to the nearest whole number.

If the time samples are then plotted back onto the same graph, it can be seen that the sound wave now looks different. This is because sampling does not take into account what the sound wave is doing in between each time sample.

Sound loses quality as data has been lost between the time samples

The way to increase the quality is to have more time samples that are closer together.



The frequency at which samples are taken is called the **sample rate**, and is measured in Hertz (Hz).

Compression

Lossy compression
Lossy compression removes some of a file's original data in order to reduce the file size.

Lossless compression
Lossless compression doesn't reduce the quality of the file at all. There are various algorithms for doing this. The space savings of lossless compression are not as good as they are with **lossy** compression.

Cyber security - revision on a page

CYBER SECURITY

What is cyber security?

Protecting networks, computers, programs and data from attack, damage or unauthorised access through the use of technologies, processes and practices.

Cyber security in depth

1. Social engineering

Manipulating individuals so they give away personal information (e.g. bank account).

Blagging - inventing a scenario to target someone into divulging info. Companies can give employees security training.

Phishing - fraudulently obtaining personal info (using e.g. email or SMS). Beware of links in emails!

Pharming - cyber attack to redirect a website's traffic to another, fake site. Check the http address has http**'s'**

Shouldering - observing a person's private info over their shoulder (e.g. ATM)



2. Malicious code

Malware - umbrella term to describe a variety of hostile or intrusive s/w: -

- **Computer virus** - installed on your computer without your permission with the intention to do harm. Viruses spread through email attachments or IM services OR through files/programs downloaded
- **Trojan** - pretends to have a legitimate purpose. Spread by email
- **Spyware** - gathers info without user knowing (e.g. bank account details).
- **Adware** - internet usage analysed and then advertising targeted.

Other Cyber security threats

3. Weak and default passwords	Using weak or default passwords.
4. Misconfigured access rights	Weakness from giving wrong network access.
5. Removable media 	Allowing the use of removable media (USB stick) in an organisation risks data theft & viruses.
6. Unpatched and/or outdated s/w	Vulnerabilities caused by out of date software.

Methods to detect and prevent cyber security threats

Penetration testing - used to find security weaknesses by attempting to gain access to resources without the knowledge of usernames, passwords etc. *White box test* to simulate a malicious insider who has basic network knowledge. How much damage can they do? *Black box test* to simulate an external hacking or cyber warfare attack.

Biometric measures - E.g. fingerprint access to smartphone, retinal scan to gain access to computer room.

Password systems

CAPTCHA - are you a robot?

Email verification - e.g. sign up to new web service.

Automatic software updates.

Anti-malware software.