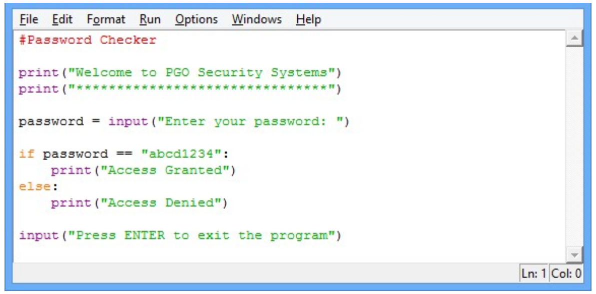
Computing Knowledge Organiser

# Year 9 Python Knowledge Organiser

## Programming with Python



```
#Password Checker

print("Welcome to PGO Security Systems")
print("*******************************")

password = input("Enter your password: ")

if password == "abcd1234":
    print("Access Granted")
else:
    print("Access Denied")

input("Press ENTER to exit the program")
```

## 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

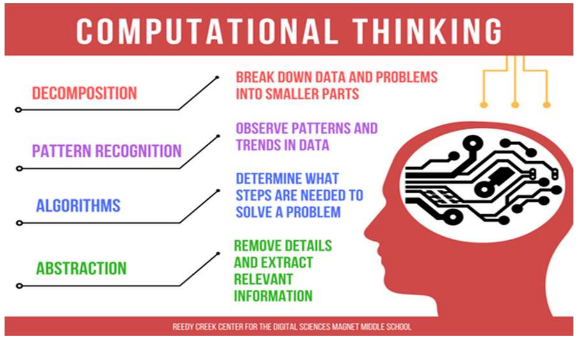
**Algorithms** consist of steps that are carried out (performed) one after another. Sometimes an **algorithm** needs to **repeat** certain steps until told to stop or until a particular condition has been met. ***Iteration is the process of repeating steps.***

**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.

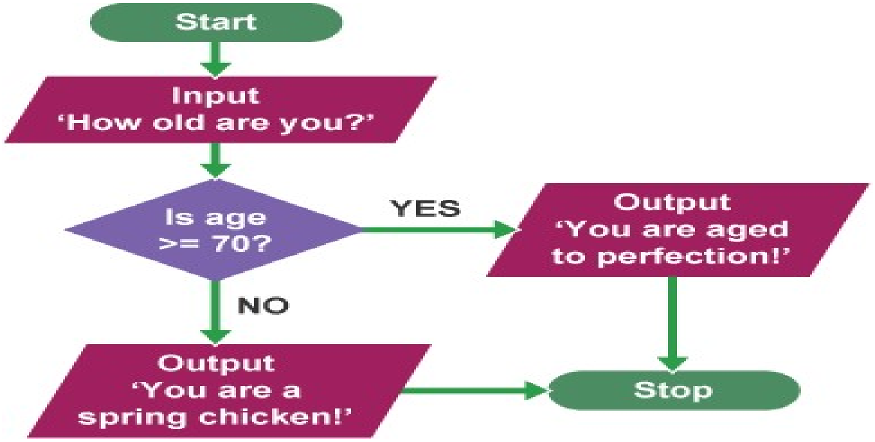

COMPUTATIONAL THINKING
DECOMPOSITION
BREAK DOWN DATA AND PROBLEMS INTO SMALLER PARTS
PATTERN RECOGNITION
OBSERVE PATTERNS AND TRENDS IN DATA
ALGORITHMS
DETERMINE WHAT STEPS ARE NEEDED TO SOLVE A PROBLEM
ABSTRACTION
REMOVE DETAILS AND EXTRACT RELEVANT INFORMATION
REEDY CREEK CENTER FOR THE DIGITAL SCIENCES MAGNET MIDDLE SCHOOL

## 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!"


Start
Input 'How old are you?'
Is age >= 70?
YES
Output 'You are aged to perfection!'
NO
Output 'You are a spring chicken!'
Stop

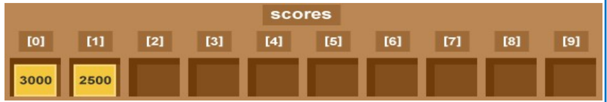
**The above algorithm would be written in Python (3.x) as:**

```
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