# Detailed guide for learning to program in Python 3

This is a general guide to assist in learning Python 3. Not all the components here are necessary for teaching or learning programming for Edexcel GCSE (9-1) Computer Science, for example Turtle graphics knowledge is not required. Teachers should refer to the specification for the content which will be assessed.

## Python data types

| Data type | Python Abbreviation | Explanation | Example |
|---|---|---|---|
| integer | int | A whole number. | 45 |
| string | str | A sequence of characters that can include letters, spaces and other characters. | "Have a nice day!" |
| float | float | A number with a fractional part. Also known as a real number. | 16.76 |
| Boolean | bool | Boolean or logical data that can only have one of two values: True or False. | True<br>False |

## Built-in functions

| Syntax | Description | Example |
|---|---|---|
| len() | Calculates the length of a string. | >>> ans=len("my string")<br>>>> ans<br>9 |
| print() | Displays information on the screen. | >>> print("Hello world") |
| type() | Displays the type (int, bool, str or float) of a variable or value. | >>> ans=7.8<br>>>> type(ans)<br><class 'float'> |
| int() | Converts a string or float value into an integer number.<br>Often used in conjunction with the input function, e.g.<br>number = int(input("Please enter the number of T-shirts you require:")) | >>> ans=7.8<br>>>> int(ans)<br>7 |
| input("prompt") | Prompts for input from the user. The data entered is assigned to a variable. | >>> reply=input("Enter your name: ")<br>Enter your name: Fred<br>>>> reply<br>'Fred' |

| Built-in functions | | |
|---|---|---|
| **Syntax** | **Description** | **Example** |
| range() | Creates a list of numbers.<br>Often used with the for loop, e.g.<br>range(start number, end number, in steps of). End number is the number after the last number required. | >>> for next in range(1,4):<br>print(next)<br>1<br>2<br>3 |
| max() | Returns the largest of a set of numbers. | >>>max(12,16, 33)<br>33 |

| Variables and lists (arrays) | | |
|---|---|---|
| **Syntax** | **Description** | **Example** |
| variableName = <value> | Assigns a value to a variable. | myString="hello world"<br>myNumber= 89<br>myAnswer=True |
| variableName = <expression> | Computes the value of an expression and assigns it to a variable. | number= 7 * 8<br>answer= len("this is the age") |
| listName = [value,value,value] | Assigns a set of values to a list. | myList= ["apple","oranges",8] |
| listName[index] | Identifies an element of a list by reference to its position in the list, where index is an integer value starting at 0. | myList[2] |
| listName[row][column] | Identifies an element of a nested list (equivalent to a two dimensional array). | myList[0][6] |

| Nested lists in Python (two-dimensional arrays) | |
|---|---|
| **How to…** | **Example** |
| initialise a two-dimensional array | rowLength=4<br>columnLength=6<br>myArray=[[0 for row in range(rowLength)] for column in range(columnLength)] |
| address an element in a two- | [row][column] |

| | |
|---|---|
| dimensional array | |
| assign a value to an element in a two-dimensional array | myArray[0][4] = 99<br>myArray[2][3] = 74 |
| print the contents of a two-dimensional array, a row at a time | for row in range(rowLength):<br>    print(myArray[row]) |

| Selection constructs | | |
|---|---|---|
| **Syntax** | **Description** | **Example** |
| if <expression>:<br>  <commands> | If <expression> is true then commands are executed. | if colour == "green":<br>  print("It is safe for you to cross.") |
| if <expression>:<br>  <commands1><br>else:<br>  <commands2> | If<expression> is true then<commands1> are executed otherwise <commands2> are executed | if colour == "green":<br>  print("It is safe for your to cross.")<br>else:<br>  print("STOP! It is not safe to cross.") |
| if <expressionA>:<br>  <commands1><br>elif <expressionB>:<br>  <commands2><br>elif <expressionC>:<br>  <commands3> | If <expressionA> is true then <commands1>are executed, else if <expressionB> is true then <commands2>are executed, etc. | if answer == 1:<br>  print("You will make a new friend this week.")<br>elif answer == 2:<br>  print("You will do well in your GCSEs.")<br>elif answer == 3:<br>  print("You will find something you thought you'd lost.") |
| try:<br>  <commands1><br>except:<br>  <commands2><br>else:<br>  <commands3> | If <commands1> cause an error, then <commands2> are executed. If <commands1> execute successfully, then <commands3>are executed. | try:<br>  ans=numOne/numTwo<br>except ZeroDivisionError:<br>  print("Second number cannot be zero!")<br>else:<br>  print("The answer is: ", ans) |

| Iteration constructs | | |
|---|---|---|
| **Syntax** | **Description** | **Example** |
| for variable in <expression>:<br>  <commands> | Executes <commands> for a fixed number of times, given by <expression>. | myList=["cat","dog","cow","donkey","rabbit","canary"]<br>  for next in myList:<br>  print(next) |
| while <condition>:<br>  <commands> | Executes <commands whilst <condition> is true. This is a pre-condition loop. | answer="N"<br>counter=0<br>while answer != "Y":<br>  print("Are you hungry? You have been asked {0} times.".format(counter))<br>  answer = input("Please respond Y or N:")<br>  counter = counter + 1<br>print("Please get something to eat!") |

| File handling | | |
|---|---|---|
| **Syntax** | **Description** | **Example** |
| <file identifier variable> = open(<filename>, "flag") | Opens a file. The flag is: reading (r), writing (w) appending (a). This opens the file and creates a file object. | myFile=open("file.txt","r")<br>myFile=open("file.txt","w") |
| <file identifier variable>.close() | Closes the file. | myFile.close() |
| <file identifier variable>.write(<string>) | Writes string to a file. Use "\n" at the end of each record. | myFile.write("this is a record \n") |
| Variable = <file identifier variable>.readline() | Reads a line from a file and assigns to a variable. | record = myFile.readline() |

| How to... | Explanation |
|---|---|
| start the debugger | In IDLE shell choose Debug/Debugger |
| save a file | File > Save<br>*Hint: file type is always.py* |
| run a program | Run > Run module<br>or F5 |
| display the last command entered in the shell | alt p |
| repeat the last command entered in the shell | alt n |
| indent and dedent blocks of code | Select and use Format > Indent and Format > Dedent |
| interrupt a program that is running | control z or control c |

| Escape sequence | Effect |
|---|---|
| \t | tab |
| \n | new line |
| \\ | displays \ |
| \’ | displays ‘ |
| \” | displays “ |

| Precedence |
|---|
| Parentheses control the order in which expressions are calculated. Anything in parentheses is evaluated first.<br>The precedence order is: parenthesis (round brackets), exponential, division and multiplication, subtract and add.<br>B E D M A S |

| Python errors | Description |
|---|---|
| TypeError | When an operation is attempted that is invalid for that type of data. |
| RuntimeError | An error that occurs when the program is running. |
| NameError | When a name is used that is not known about (often a misspelt variable name). |
| ZeroDivisionError | Dividing a number by zero. |
| KeyBoardInterrupt | When a program is interrupted from the keyboard by pressing control+c |

| Rules for variable names |
|---|
| Must begin with a letter (upper or lower case) followed by zero or more other letters or numbers. |
| Cannot have spaces in the name. |
| Can include "_", e.g. My_variable. |
| Cannot use reserved Python command words. |

| Reserved Python command words | | |
|---|---|---|
| and | exec | not |
| assert | finally | or |
| break | for | pass |
| class | from | print |
| continue | global | raise |
| def | if | return |
| del | import | try |
| elif | in | while |
| else | is | with |
| except | lambda | yield |

| Mathematical operator symbol | Operation |
|---|---|
| / | divide |
| * | multiply |
| ** | exponential |
| + | add |
| - | subtract |
| // | integer division |
| % | modulus (remainder after the division) |

| Operator meaning | Operator | Sample condition | Evaluates to |
|---|---|---|---|
| Equal to | == | "fred" == "sid" | False |
| Not equal to | != | 8 != 8 | False |
| Greater than | > | 10 > 2 | True |
| Greater than or equal to | >= | 5 >= 5 | True |
| Less than | < | 40 < 34 | False |
| Less than or equal to | <= | 2 <= 109 | True |

| Symbol | Description |
|---|---|
| AND | Returns true if both conditions are true. |
| OR | Returns true if one of the conditions is true. |
| NOT | Reverses the outcome of the expression; true becomes false, false becomes true. |

| Python command colour coding | What does it show | Example |
|---|---|---|
| green | strings | “hello” |
| purple | functions | print() |
| black | variables and data | myName |
| orange | key commands | if |
| red | comments | # This is a comment |

## Examples of modules

### math module

| Function name | Function explanation | Example |
|---|---|---|
| math.pi | Displays the pi constant. | >>>math.pi<br>3.141592653589793 |
| math.pow() | Displays the first number raised to the power of the second number. | >>> ans=math.pow(2,3)<br>>>>ans<br>8.0 |
| math.sqrt() | Displays the square root of a number. | >>> ans=math.sqrt(1024)<br>>>>ans<br>32.0 |

### random module

| Function name | Function explanation | Example |
|---|---|---|
| random() | Returns a random number. | >>> import random<br>>>> ans = random.randint(1,6)<br>>>>ans<br>3 |

### winsound module

| Function name | Function explanation | Example |
|---|---|---|
| winsound.Beep(frequency,duration) | Makes a sound in the range frequency (hertz) for the duration (milliseconds). | import winsound<br>frequency = 100<br>for i in range (40):<br>  winsound.Beep(frequency,200)<br>  frequency = frequency + 50 |
| winsound.PlaySound(filename, flags) | Plays the sound file identified by file name using the flags to indicate type. | winsound.PlaySound("SystemExit", winsound.SND_ALIAS) |

### webbrowser module

| Function name | Function explanation | Example |
|---|---|---|
| webbrowser.open("URL") | Open the URL given as a string in the default web | webbrowser.open("http://www.bbc.co.uk") |

| | | |
|---|---|---|
| | browser. | |

| Keyboard short cut | Explanation |
|---|---|
| Control c | copy selected text |
| Control v | paste |
| Control a | select all |
| Control x | cut selected text |
| Control f | find |
| Control n | open a new window |
| Control p | print |
| Control s | save |
| Control z | undo |

| Turtle graphics (using x,y cartesian co-ordinates) | | |
|---|---|---|
| **Turtle command** | **Explanation** | **example** |
| import turtle | Imports module for turtle graphics. | import turtle |
| t = turtle.Pen() | Makes a turtle. | t = turtle.Pen() |
| turtle.shape("shape") | Sets shape of turtle ("arrow", "turtle", "circle", "square"). | t = turtle.shape("turtle") |
| turtle.color(colour) | Changes turtle colour. | t = turtle.color("red") |
| turtle.forward(distance)<br>turtle.backward(distance) | Moves turtle forward/backward by distance given in direction it is heading. | t = turtle.forward("100") |
| turtle.right(angle)<br>turtle.left(angle) | Turtle turns right or left by angle given. | t = turtle.right("90") |
| turtle.goto() | Turtle goes to x,y position. | t = turtle.goto(200,-200) |
| turtle.setheading() | Sets the direction the turtle will move (in degrees) 0=north, 90 = east, 180=south, 270=west. | t = setheading(180) |
| turtle.speed() | Sets turtle speed (1 to 10). | t = turtle.speed(10) |
| turtle.position() | Reports current position as x,y co-ordinates. | t.position() |
| turtle.heading() | Reports current heading. | t.heading() |
| turtle.circle(radius) | Draws a circle of a given radius. | t = turtle.circle(15) |
| turtle.stamp() | Draws a picture of itself on the canvas. | t = turtle.stamp() |
| turtle.penup()<br>turtle.pendown() | Puts the pen up or down. | t = turtle.penup() |
| turtle.pensize(width) | Changes pen size. | t = turtle.pensize(5) |
| turtle.pencolor(colour) | Changes pen colour. | t = turtle.pencolor("orange") |
| turtle.hideturtle()<br>turtle.showturtle() | Hides and shows turtle. | t = turtle.hideturtle() |
| turtle.clear() | Deletes all turtle drawings. | t = turtle.clear() |
| turtle.reset() | Deletes drawing, re-centres turtle and sets variables to default values. (To reset everything close Python and start again.) | t = turtle.reset() |

| | | |
|---|---|---|
| turtle.undo(number) | Undoes the last command. | t = turtle.undo() |
| turtle.delay(number) | Inserts a delay before next command. | t = turtle.delay(50) |

# Glossary

| Term | Definition |
|---|---|
| Algorithm | A sequence of steps to perform a task. |
| Flowchart | A graphical representation of an algorithm that uses flow lines and shapes to represent the operations. |
| Pseudocode | Pseudocode looks a bit like a programming language but, unlike a real programming language, it does not require a strict syntax. This makes it particularly useful for designing programs.<br>Because it is not an actual programming language, pseudocode cannot be compiled into executable code. |
| Structured English | Similar to pseudocode, structured English is a way of describing an algorithm using English words to describe each step of the process. |
| Program code | The set of instructions that form a program written in a particular programming language. |
| | |
| Data type | The classification of data as being a string, integer, float or Boolean (logical). |
| Boolean | Boolean or logical data that can only have one of two values, i.e. True or False (1 = True and 0 = False). |
| Float | A number with a fractional part, e.g. 34.67. Also known as a real number. |
| Integer | A whole number, e.g. 45. |
| String | A sequence of characters that can include letters, spaces, symbols, e.g. "This is a name £££46 " |
| | |
| Variable | A named location in a computer's memory where data is stored. |
| Global variable | A variable assignment that is available throughout the whole program. |
| Local variable | A variable assignment that is only available within a subprogram. |
| Type declaration | When a variable is allocated a particular data type (integer, string, Boolean, float). |
| Variable declaration | When a value is assigned to a variable. |
| Constant | A variable assignment that stays the same throughout a program. |
| Data structure | A group of related data items. Examples include strings, arrays and lists. |
| | |
| Command sequence | A block of program commands that are executed sequentially, i.e. one after the other. |
| Loop | A piece of code that keeps repeating until a certain condition is reached. |

| Term | Definition |
|---|---|
| Output | Data that is sent out from a program to a screen, a file or a printer. |
| Indentation | Positioning a block of code, e.g. a for loop or an if statement, further from the margin than the main program.<br>The use of indentation makes programs easier to read.<br>In Python correct indentation is very important, since indenting starts a block and unindenting ends it. |
| Comment | Text included in code that is for the human reader and is ignored by the program.<br>In Python, comments are indicated by the # symbol at the start of a line. |
| Structural components | The parts of a program such as variable declarations, data structures or subprograms. |
| | |
| Subprogram | A small computer program that runs within another computer program. Subprograms are used to split up a program into a number of smaller programs, with each subprogram performing a specific function. Subprograms can be called in any order any number of times. |
| Function | A subprogram that returns a value. |
| Procedure | A subprogram that does not return a value. |
| Return value | The value returned by a subprogram. |
| Parameter | The names that appear in a function definition when passing data to a function. |
| Argument | A piece of information/value that is required by a function to perform a task, e.g. function(argument1, argument2) |
| Built-in subprograms | Pre-existing libraries of subprograms that are built into the programming language. |
| Library subprograms | Pre-existing libraries of sub-programs that can be imported and used in the programming language. |
| | |
| Mathematical operators | Mathematical operators take two operands and perform a calculation on them.<br>Examples of mathematical operators are: '+' (addition), '-' (subtraction), '*' (multiplication), '/' (division). |
| Precedence | The order in which values are calculated. |
| Integer division | Division in which the remainder is discarded, e.g. 10//3 = 3. |
| Modulus | The absolute (non-negative) value of a number. The modulus of -3 is 3. |

| Term | Definition |
|---|---|
| Relational operators | Relational operators take two operands and compare them.<br>Examples of relational operators are: < (less than) < (greater than), == (equal to), != (not equal to).<br>Relational operators are used in conditional statements, especially in loops, where the result of the comparison decides whether execution should proceed. |
| Logical operators | Logical operators take two operands and compare them, returning True or False depending on the outcome.<br>Examples of logical operators are: AND, OR, NOT. |
| | |
| Logic error | An error in the design of a program, such as the use of a wrong programming control structure or the wrong logic in a condition statement.<br>This type of error may not produce an error message, just the wrong outcome. |
| Runtime error | An error detected during program execution, often due to a mistake in the algorithm or in the type of data used. |
| Syntax error | An error that occurs when a program statement cannot be understood because it does not follow the rules of the programming language. |
| Trace table | A technique used to test for logical errors in a program. Each column of the table is used to represent a variable and each row to show the values of the variables at different stages of the program. |
| | |
| Break point | A point in the code when a program is halted so that the programmer can investigate the values of variables to help locate errors. |
| Bug | An error in the code that prevents a program from running properly, or from running at all. |
| Debugger | A tool that helps to detect, locate and correct faults (or bugs) in programs. |
| Single step | When a program is executed one statement at a time under user control. |
| Watchers | Allow the programmer to watch for certain events, such as variables changing as the program is running. |
| | |
| Test plan | A list of tests to be carried out, designed to test a program in the widest possible range of situations to make sure it is working. |
| Test data | Data used in a test plan to test what happens when the input data is valid, invalid or extreme. |
| Valid test data | Test data that is within the normal range and should be accepted by the program. |

| Term | Definition |
|---|---|
| Invalid test data | Test data that is outside the normal range and should be rejected by the program. |
| Extreme test data | Test data that is still valid, but is on the absolute limits of the normal range and should be accepted by the program. |
| | |
| Textual user interface | A type of interface in which the user interacts with the computer program by using a keyboard to enter commands and make selections. |
| Graphical User Interface (GUI) | A type of user interface that uses windows, icons, buttons and menus. A pointer is used to make selections. |
| Validation | Validation is designed to ensure that a program only operates on appropriate data, e.g. that it has an appropriate format or value. It involves checking the data that a user enters or that is read in from a file and rejecting data that does not meet specified requirements.<br>However, validation can only prove that the data entered is appropriate, not that it is correct. |
| | |
| Integrated development environment (IDE) | A text editor with useful built in tools to help programmers develop programs, e.g. IDLE. |
| | |
| Cartesian co-ordinates | A co-ordinate system that specifies a point uniquely using (x,y) co-ordinates. |