GCSE (9–1)
COMPUTER SCIENCE

OCR
Oxford Cambridge and RSA

FIRST ASSESSMENT
SUMMER 2022

J277 Guide to programming techniques: Python

Introduction

This guide is designed to support candidates' learning about how to use Python and how it relates to the OCR Exam Reference Language.

Please refer to the J277 Specification, Section 2.2 for a full list of skills/techniques that candidates must be familiar with.

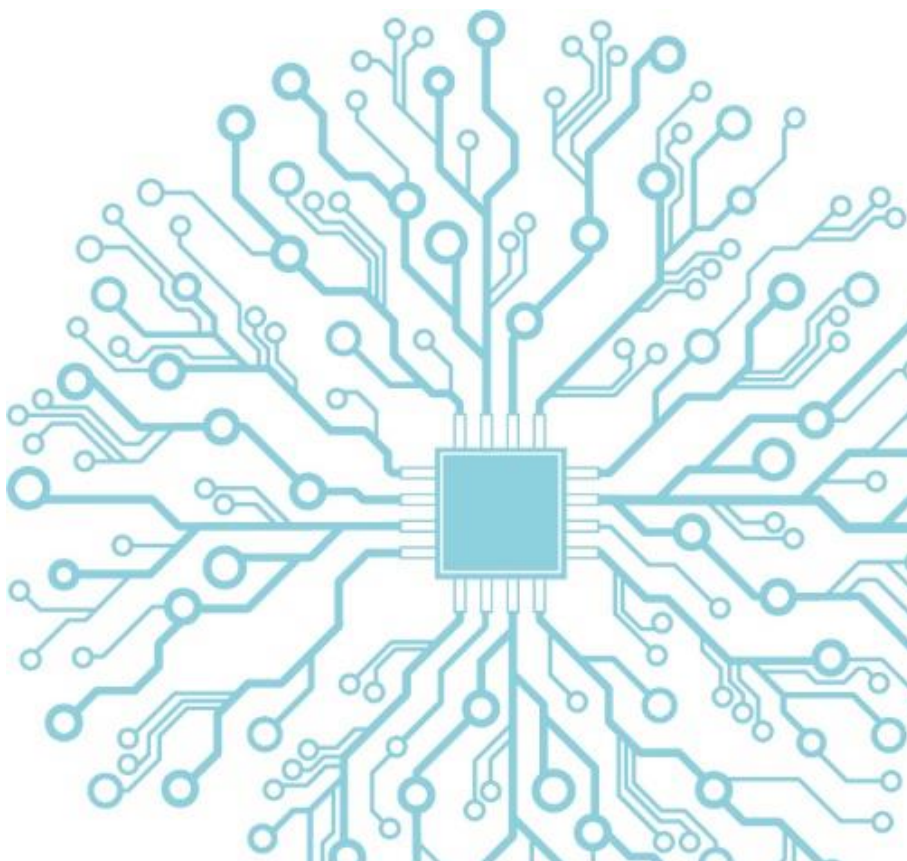
Disclaimer: Please note that this is not a complete guide to Python and only explores some of the ways to use Python to express the techniques in the specification.

Using the guide

This guide uses Python 3.

>>> this denotes the use of the interpreter (shell) and not a saved .py file.

If you are copying and pasting the code below, sometimes you will need to change the quote marks (") in your chosen IDE as sometimes the formatting means the IDE doesn't recognise them.



Version 1

Contents

The use of variables

OCR Exam Reference Language

`x = 3 name = "Bob"`	Variables are assigned using the = operator.
`const pi = 3.14`	Variables in the main program can be made a constant with the keyword const.
`global userID = 123`	Variables in the main program can be made global with the keyword global.

Python

`>>> count = 5`
 	
`>>> count`
 	
`5`
 	
`>>> total = 2`
 	
`>>> count + total`
 	
`7`
 	
`>>> count = count + 2`
 	
`>>> count`
 	
`7`
	A variable is initialised (created) as soon as a value is stored in it. The variable count is assigned the value 5. When count is called it returns the value 5.

	Once assigned you can use the variable with other values or variables such as count + total evaluating to 7 (5+2).
	A variable can be overwritten with a new value at any time.

`>>> count = "it is a silly place"`
 	
`>>> count`
 	
`"it is a silly place"`
	You can assign other data types to variables. Here we assign the letters "it is a silly place" to spam.

`>>> pi = 3.14`
 	
`>>> pi`
 	
`3.14`
	There are no constants in Python, instead use a variable and simply don't change it.
	
	In Python you simply document that it should not be changed. .

`someGlobal = 10`
 	
`def func1():`
`    someGlobal = 20`
	You may think this will print 20 but it prints 10, In Python the scope of a variable lies within a function. If there is not a name assigned within the function it looks outside of it, but not in

`def func2():`
`    print(someGlobal)`
 	
`func1()`
`func2()`
	other functions. If you want a variable in function to be treated as a global variable, then you can use the global keyword as below:
	
	`def func1():`
	`    global someGlobal`
	`    myGlobal = 20`

There are some rules with variable names in Python:

- they can only be one word
- they can only use letters, numbers and underscores (_)
- hyphens are not allowed (-)
- spaces are not allowed
- they can't begin with a number
- special characters are not allowed such as $ or "

Please remember:

- variable names are case sensitive, COUNT and count are different variables
- it is convention in Python to use all lower case letters for variable name, using underscore_separators or CamelCase
- a good variable name describes the data it contains

Operators

OCR Exam Reference Language

Comparison operators

`AND`	`Logical AND`
`OR`	`Logical OR`
`NOT`	`Logical NOT`

Comparison operators

`==`	Equal to
`!=`	Not equal to
`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to

Arithmetic operators

`+`	Addition e.g. `x=6+5` gives 11
`-`	Subtraction e.g. `x=6-5` gives 1
`*`	Multiplication e.g. `x=12*2` gives 24
`/`	Division e.g. `x=12/2` gives 6
`MOD`	Modulus e.g. `12MOD5` gives 2
`DIV`	Quotient e.g. `17DIV5` gives 3
`^`	Exponentiation e.g. `3^4` gives 81

Python

Comparison operators

`>>> cats = 9`
`>>> cats == 9`
 	
`True`
 	
`examBoard = "OCR"`
`print ("My exam board is OCR")`
`print (examBoard == "OCR")`
 	
`True`
	We set `examBoard` to `OCR` then test whether they are equivalent which returns as `True`.

`>>> 5 != 9`
 	
`True`
 	
`parrots = 1`
`if parrots != 2:`
`    print ("squawk")`
 	
`squawk`
	Five is not equal to nine.
	
	`parrots` is equal to 1, if `parrots` does not equal 2 then it prints `squawk`.

`>>> 6 > 6`
 	
`False`
 	
`>>> (1>2) and (9>=1)`
 	
`False`
	Six is not greater than six.
	
	One is not greater than two (False), nine is not greater than or equal to one (False), so False AND False evaluates to False.

`>>> 7 <= 7`
 	
`True`
 	
`>>> (6 < 4) or (4 != 3)`
 	
`True`
	Seven is less than or equal to seven.
	
	Six is not less than 4 (False), 4 is not equal to 3 (True), so False OR True evaluates to True.

`>>> 8 > 2`
 	
`True`
 	
`>>> (1>2) and (2>4)`
 	
`False`
	Eight is greater than 2.
	
	1 is greater than 2 (True), 2 is greater than 4 (False). True AND False evaluates to False.

`>>> 9 >= 3`
 	
`True`
 	
`>>> (10 >= 1) and (1 < 2)`
 	
`True`
	Nine is greater than or equal to 3.
	
	Ten is greater than or equal to 1 (True) and 1 is less than 2 (True). True AND True evaluates to True.

Arithmetic operators

`>>> 1 + 1` `2`	One add one equals 2.
`>>> 8 – 10` `-2`	Eight take away ten evaluates to negative two.
`>>> 2 * 6` `12`	Two multiplied by six evaluates to twelve.
`>>> 6 / 2` `3`	Six divided by two evaluates to three.
`>>> 4 % 3` `1`	Four MOD three evaluates to 1
`>>> 9 // 2` `4`	Nine DIV two evaluates to four.
`>>> 4 ** 4` `256`	Four ^ (exponent) four evaluates to two hundred and fifty-six.

Inputs

OCR Exam Reference Language

`myName = input("Please enter your name")`	Here we declare a variable myName and assign the input to it. We also prompt the user as to what to input.

Python

`>>> print("What is your favourite colour?")` `>>> favColour = input()` `print(favColour)`	You don't have to prompt the user in Python but it usually helps. Inputs can be stored as a variable so they can be used later.
`>>> favColour = input("What is your favourite colour?")`	You can also combine the message as an argument.

Outputs and assignments

OCR Exam Reference Language

print(string)
print(variable)
	Outputs the argument (string or variable) to the screen.

Python

`>>> print("The parrot is no more")`
 	
`The parrot is no more`
 	
`>>> number = 66`
`>>> print(number)`
 	
`66`
	The `print` function takes an argument that is then printed to the screen.

Sequence

OCR Exam Reference Language

`x = 3`
`y = 2`
`x = x + y`
`print(x)`
	`x` is assigned the value of 3, `y` is assigned the value of 2. `x` is then re-assigned to be the value of 3 plus 2 which evaluates to 5 and is printed to the screen.
	
	It should be noted that that value of `x` changes in sequence, line by line as it is interpreted, at the start of line 3 (`x=x+y`) `x` still has a value of 3 but once that line is run it then changes to be `x+y` or 5.

Python

`>>> numberOne = 2`
`>>> numberTwo = 2`
`>>> print(numberOne)`
 	
`2`
 	
`>>> numberOne = numberOne +`
`numberTwo`
`>>> print(numberOne)`
 	
`4`
	`numberOne` is assigned the value 2.
	`numberTwo` is also assigned the value of 2.
	
	`numberOne` is then re-assigned to be `numberOne` (2 as it is currently) plus `numberTwo`, which evaluates to 4.
	
	Similarly in this example the value of `numberOne` is 2 until the line `numberOne = numberOne + numberTwo` is interpreted which results in `numberOne` now has a value of 4.

Selection

It helps to think of selection as a test of a condition such as:

if some condition is met:
    do something

OCR Exam Reference Language

```
if entry == "a" then
    print("You selected A")
elseif entry=="b" then
    print("You selected B")
else
    print("Unrecognised ")
endif

switch entry:
    case "A":
        print("You selected A")
    case "B":
        print("You selected B")
    default:
        print("Unrecognised")
endswitch
```
	Selection will be carried out with if/else and switch/case. In the example the OCR Exam Reference Language is checking the input and returning a message based upon the specific input required, the else block is used as a catch for any unexpected input which allows the code to degrade gracefully.
	
	The switch/case method works in the same way.

Python

```
airSpeedVelocity = 11
if airSpeedVelocity <= 11:
    print ("European")
else:
    print ("African")
```
	The `airSpeedVelocity` has a value of 20 the if statement is used to test whether the value of `airSpeedVelocity` is greater than or equal to 22. If it evaluates to True then it prints "European" otherwise it prints "African". The else block is only executed if the conditional test returns False. This is great for situation where there are only two outcomes.

```
points  = 4
if points == 4:
    print("Max Score")
elif points > 0 <4:
    print("You have
scored",points)
else:
    print("No points scored")

ni = ["shrubbery", "slightly
higher", "a little path"]

if "shrubbery" in ni:
    print ("Ekky ekky")
```
	We can go further and add in more options by using an `elif` that allows more conditional tests. Note that the `elif` has 2 conditional tests, greater than 0 AND less than 4.
	
	You can use multiple elif statements if necessary.
	
	Sometimes there are multiple conditions that could be True and in this case you should use the `in` operator to do a membership test in a

`if "slightly higher" in ni:`
`    print ("kerplang")`
`if "a little path" in ni:`
`    print ("zoot boing")`
	sequence of accepted elements in a list for example.

Python does **not** support **switch/case** statements

Iteration (count-controlled loops)

OCR Exam Reference Language

`for i=0 to 7`
`   print ("Hello")`
`next i`
	Will print "Hello" 8 times (0-7 inclusive). Note that the count starts at 0.

`for i=0 to 7 step 2`
`   print ("Hello")`
`next i`
	Will print "Hello" 4 times. The `step 2` command increases the counter (`i`) by 2 each time.

Python

`print("Here are 5 Knights")`
 	
`for i in range(5):`
 	
`    print("Knight ("+str(i)+")")`
 	
`Knight (0)`
`Knight (1)`
`Knight (2)`
`Knight (3)`
`Knight (4)`
	The `for` loop will loop for a set number of times as defined by the `range()` function. In this example we print a string then print 5 times the string `"Knight"` followed by the value for `i`.

`guess = 0`
`for num in range(101):`
`    guess = guess + num`
`print(guess)`
 	
`5050`
	In this example we are adding up all the numbers from 0 to 100 using a `for` loop. This shows how useful they can be.

`for i in range(0,10,3):`
`    print(i)`
 	
`0`
`3`
`6`
`9`
	You can also use three arguments in the range function `range(start_value, stop_value, step_value)`. The step value is the value by which the variable is increased by each iteration.

Iteration (condition controlled loops)

OCR Exam Reference Language

`while answer!= "x"`
`   answer = input("Press any key to continue or x to quit")`
`endwhile`
	Condition-controlled loop, this will loop until the user inputs "x". It will check the condition before entering the loop.

`do`
`   answer = input("New answer")`
`until answer != "Correct"`
	The loop iterates once before the check is carried out.
	
	Note that the 'until' means that the logic of the loop has now changed. **Be careful** when writing this in Pseudocode!

Python

`coconut = 0`
`while coconut < 3:`
`    print("clip clop")`
`    coconut = coconut + 1`
 	
`clip clop`
`clip clop`
`clip clop`
	A while statement is a condition controlled loop. The indented code will be repeated WHILE the condition is met

`while 1 == 1:`
`    print ("lol")`
 	
`***infinite lols***`
	One thing to be careful of is creating an infinite loop. In the example the while loop checks whether 1 is equal to 1 and then prints "`lol`" so it will print "`lol`" for ever.

`troll = 0`
`while troll <1:`
`    print ("lol ")`
`    troll = troll + 1`
`    break`
`print("phew ")`
 	
`phew`
	You can use a `break` statement to jump out of a loop. In Python you will not need this if you use the loop properly.

`for letter in "Python":`
 	
`   if letter == "h":`
 	
`      continue`
 	
`   print ("Current Letter :", letter)`
 	
`Current Letter : P`
`Current Letter : y`
`Current Letter : t`
`Current Letter : o`
`Current Letter : n`
	You can also use a `continue` statement that when reached will jump back to the start of the loop and re-evaluate the loop's condition just as when the loop reaches the end of the loop. In this example the continue statement rejects the remaining statement in the current iteration of the loop and moves the control back to the top of the loop.

The use of basic string manipulation

OCR Exam Reference Language

`stringname.length`
`subject = "Computer Science"`
`subject.length`
	This gets the length of a string.
	`subject.length` will return 15

`stringname.subString(startingPosition, numberOfCharacters)`
 	
`subject.substring(3,5)`
`subject.left(4)`
`subject.right(3)`
	This gets a substring but the string will start at the 0th character.
	
	`subject.substring(3,5)` will return `"puter"`
	
	`subject.left(4)` will return `"Comp"`
	`subject.right(3)` will return `"nce"`

`stringname.upper`
`stringname.lower`
 	
`subject.upper`
`subject.lower`
	This converts the case of the string to either upper or lower case.
	
	`subject.upper` will return `"COMPUTER SCIENCE"`
	`subject.lower` will return `"computer science"`
	
	.

`ASC(character)`
`CHR(asciinumber)`
 	
`ASC(A)`
`CHR(97)`
	This converts to and from ASCII.
	
	`ASC(A)` will return `65 (numerical)`
	`CHR(97)` will return `"a" (char)`
	
	`Uppercase letters and lowercase letters have different ASCII values as does numbers represented in a string.`

`someText="Computer Science"`
`print(someText.length)`
`print(someText.substring(3,3))`
 	
`16`
`put`
	Here length of the variable is printed along with the 3 characters 3 character in for 3 characters.

Python

`>>> food = "eggs"`
`>>> print(len(food))`
 	
`4`
 	
`>>> food = ["eggs","oranges","apples"]`
`>>> print(len(food))`
 	
`3`
	Here we define a variable as the string `"eggs"` and then print the length of the string using the `len` function.
	
	This can also be done with a list where the number of values in the list is returned.

`>>> animal = "It\’s only a bunny"`
`>>> print(animal[0:5])`
 	
`It’s`
 	
`>>> food = ["eggs","oranges","apples"]`
 	
`>>> print(food[:2])`
 	
`["eggs","oranges"]`
 	
`>>> print(food[2:])`
 	
`["apples"]`
	Note the \’ that escapes (ignores) the ‘ for it’s. The substring consists of the start position and the end position of the characters. Also note its starts from 0.
	
	This can also be done with a list where the list value is returned.

`>>> fruit = "Fruit is tasty"`
 	
`>>> print(fruit.upper())`
 	
`FRUIT IS TASTY`
 	
`>>> print(fruit.lower())`
 	
`fruit is tasty`
 	
`favColour = input("What is your favorite colour?").lower()`
 	
`if favColour = "blue":`
`    print ("aaarrrrghghg")`
 	
`else:`
`    print ("no, yellow!")`
	We can use the .uppper and .lower methods to change the case of a string.
	
	Changing the case to all upper or lower makes checking the input easier as you don’t need to worry about the case.

`>>> ord("b")`
 	
`98`
 	
`>>> chr(13)`
 	
`\r`
	The `ord` function gives the integer value of a character.
	
	The `chr` function returns an integer into ascii.

`>>> tennis = "tennis"`
`>>> tennis += " ball"`
`>>> print (tennis)`
 	
`tennis ball`
	There are other interesting things you can do by using augmented assignments. The `+=` assignment for example concatenates strings.

`>>> "tennis" in "tennis ball"`
 	
`True`
 	
`>>> "gord" in "brave sir Robin"`
 	
`False`
	You can also perform logical tests on strings using `in` and `not`.

Open

OCR Exam Reference Language

`myFile = open("sample.txt")`
`x = myFile.readLine()`
`myFile.close()`
	To open a file to read from `open` is used and `readLine` to return a line of text from the file.

Python

`>>> myFile = open("myFilename")`	The first line opens a file (`myFile`) in read only by default.

Read

OCR Exam Reference Language

`myFile = open("sample.txt")`
`while NOT myFile.endOfFile()`
`      print(myFile.readLine())`
`endwhile`
`myFile.close()`
	`readLine` is used to return a line of text from the file. `endOfFile()` is used to determine the end of the file. The example will print out the contents of sample.txt

Python

`>>> myFile =`
`open("myFilename","r")`
 	
`>>> myFile.read()`
 	
`>>> for line in myFile:`
`    print (line, end = " ")`
	The first line opens a file (`myFile`) and sets the mode to read only (`"r"`). Please note that "`myfilename`" will be looked for in the same folder as the .py file unless otherwise stated.
	
	The `.read` method with no arguments will read the entire file.
	
	You can also loop through the file object line by line. The loop ends when it reaches the end of the file.

Write

OCR Exam Reference Language

`myFile = open("sample.txt")`
`myFile.writeLine("Hello World")`
`myFile.close()`
	To open a file to write to, `open` is used and `writeLine` to add a line of text to the file. In the example, `Hello world` is made the contents of sample.txt (any previous contents are overwritten).

Python

`>>> myFile.open("myFilename","w")`	In this example a variable (`myFile`) is created and then open is used to create a file object with 2 arguments. The first is a string with the filename and the second is the mode to be used. This can be: `r` – (default if not specified) read only `w` - write `a` – open for appending only `r+` - read and write

Close

OCR Exam Reference Language

`myFile.close()`	This closes the file.

Python

`myFile.close()`	When you are done with a file close it using the `.close` method to free up system resources.

The use of records to store data

OCR Exam Reference Language

`array people[5]`
`people[0]="Sir Robin"`
`people[1]="Brave"`
`people[2]="chicken"`
`people[3]="ran away"`
	Arrays will be 0 based and declared with the keyword *array*.

Python

`>>> spam = ["Sir Robin", "Brave", "chicken ", "ran away"]`
 	
`>>> print(spam[0])`
 	
`Sir Robin`
	In Python we can store records using lists or dictionaries. The record "spam" has four properties that can be indexed by position in the list.

The use of SQL to search for data

OCR Exam Reference Language

`SELECT`
 	
`FROM`
 	
`WHERE`
	SELECT LastName
	
	FROM Customers
	
	WHERE LastName = "Smith";

SQL

This example assumes there is a database created called "`Customers`" with columns called:

- CustomerID
- CustomerName
- ContactName
- Address
- City
- Country

`SELECT CustomerID FROM Customers`	This selects the CustomerID field from the `Customers` database.

`SELECT ContactName,Address`
`FROM Customers`
`WHERE ContactName = "Mr Creosote";`
	This selects the `ContactName` and `Address` columns from the `Customers` table and then specifically looks for a Mr Creosote in the `ContactName` field.

The use of arrays

OCR Exam Reference Language

`array names[5]`
`names[0]="Ahmad"`
`names[1]="Ben"`
`names[2]="Catherine"`
`names[3]="Dana"`
`names[4]="Elijah"`
 	
`print(names[3])`
	Arrays will be 0 based and declared with the keyword *array*.

`array board[8,8]`
`board[0,0]="rook"`
	Example of a 2D array:

Python

`>>> spam = ["Sir Robin", "Brave",`
`"chicken", "ran away"]`
 	
`>>> print(spam[0])`
 	
`Sir Robin`
	In this example we create a list called spam and then print the first element (0).

`>>> lol = [`
`       [1,2,3,4]`
`       [2,3,4,5]`
`       [3,4,5,6]`
`       [4,5,6,7]`
`]`
	Here we have a nested list of 3 lists of length 4.

`list_of_lists = []`	In this example we create a list of lists,

`a_list = []`
`for i in range(0,10):`
`    a_list.append(i)`
`    if len(a_list) > 3:`
`        a_list.remove(a_list[0])`
 	
`list_of_lists.append((list(a_list), a_list[0]))`
`print(list_of_lists)`
 	
`[([1, 2, 3], 1), ([2, 3, 4], 2), ([3, 4, 5], 3), ([4, 5, 6], 4), ([5, 6, 7], 5), ([6, 7, 8], 6), ([7, 8, 9], 7)]`
	the first, `[:]`, is creating a slice (normally often used for getting just part of a list), which happens to contain the entire list, and so is effectively a copy of the list.
	
	The second, `list()` is using the actual list type constructor to create a new list which has contents equal to the first list.

`breakfast = ["spam ", "eggs ", "beans ", "toast "]`
`breakfast.sort()`
`print(breakfast)`
 	
`["beans", "eggs", "spam", "toast"]`
 	
`breakfast.sort(reverse = True)`
`print(breakfast)`
 	
`["toast", "spam", "eggs", "beans"]`
 	
`lunch = ["spam ", "eggs ", "beans ", "more spam "]`
`print(sorted(lunch))`
 	
`["beans", "eggs", "more spam", "spam"]`
 	
`lunch.reverse()`
`print(lunch)`
 	
`["more spam", "beans", "eggs", "spam"]`
	Sorting lists is usually useful and you can do this by using the `.sort` method for permanent sorting or the `sorted()` function for temporary sorting of lists.
	
	You can also use arguments to reverse the order of the sort or you could use the `.reverse` method.

`#Make an empty list for storing cheese.`
`cheese = []`
 	
`#make 10 cheeses`
`for cheeseNumber in range(10):`
`    newCheese = {"type":"Cheddar","smell":"Strong", "Colour":"Yellow"}`
`    cheese.append(newCheese)`
 	
`#Show the first 2 cheeses`
`for ch in cheese[:3]:`
`    print(ch)`
 	
`{"type": "Cheddar", "smell": "Strong", "Colour": "Yellow"}`
	You can also create lists of dictionaries to make use of immutable features of a dictionary. Even though the output shows 3 dictionaries with the same information in them, Python treats each one as a separate object.

`{"type": "Cheddar", "smell": "Strong", "Colour": "Yellow"}` `{"type": "Cheddar", "smell": "Strong", "Colour": "Yellow"}`	

How to use sub programs (functions and procedures)

OCR Exam Reference Language

`function triple(number)`
`      cubedNumber=number*3`
`       return cubedNumber endfunction`
 	
`y= triple(7)`
 	
`procedure greeting(name)`
`       print("hello"+name)`
`endprocedure`
 	
`greeting("Gemma")`
	Here we define a function with a name that takes an argument `(number)`. The calculation is then performed and the function is ended.
	
	Here we can see the argument for the procedure called from main program to print a string including the argument.

Python

`def addNum(x):`
`    return(x+1)`
`y = addNum(3)        #call it`
`print(y)           #print it`
	A function is like a mini program within your program. In the example we define a function (`addNum`) and it takes an argument, 3 in the example and then assigns that to a variable and then prints it
	
	You can then call the function to carry out its function. See the 'Combinations of techniques' section below to see more functions with other techniques within them.

Integer

OCR Exam Reference Language

`int("3")`
 	
`3`
	The `int` casts the 3 as an integer.

Python

`>>> int('100')`
 	
`100`
	The `int` function is used to typecast a string into an integer.

Real

OCR Exam Reference Language

`float("3.14")`
 	
`3.14`
	The `float` casts the 3.14 into a real number.

Python

`>>> float('100')`
 	
`100.0`
	The `float` function converts from a string to a float. You can tell by the outputs .0 at the end that it is a float/real number.

Character and string

OCR Exam Reference Language

`str(3)`
 	
`"3"`
	The `str` casts the 3 into a string.

Python

`>>> string = "always look on the bright side of life"`
`>>> print(string)`
 	
`always look on the bright side of life`
 	
`>>> number = "1234"`
	Python will recognise a string as such and will automatically assign what it thinks is the correct data type. You can of course set/change the data type to typecast your variables.

`>>> num = int(number)`
`>>> num`
 	
`1234`
	Here we declare a variable with a number (1234) Python will treat this as a string unless we tell it otherwise.

Casting

OCR Exam Reference Language

`str(3)  returns "3"`
 	
`int("3") returns 3`
 	
`float("3.14") returns 3.14`
 	
`real("3.14") returns 3.14`
 	
`bool("True") return True`
	Variables can be typecast using the int str, real and bool float functions.

Python

`>>> str(100)`
 	
`'100'`
	Converts from a numeric type to a string.

`>>> int('100')`
 	
`100`
	Converts from a string to an integer.

`>>> float('100')`
 	
`100.0`
	Converts from a string to a float.

Random

OCR Exam Reference Language

`number = random(1,6)`
 	
`number = random(-1.0, 10.0)`
	Creates a random number between 1 and 6 inclusive.
	
	Creates a random real number between -1.0 and 10 inclusive

Python

`>>> import random`
`>>> number=random.randint(1,6)`
`>>> print(number)`
 	
`4`
 	
`>>> import random`
`>>> number=random.randint(-1,10)`
`>>> print(number)`
 	
`0`
	Import random imports the set of functions to use the random number generator.
	`random.randint(1,6)` creates a random number between 1 and 6 inclusive.
	
	`random.randint(-1,10)` creates a random number between -1 and 10 inclusive

Combinations of techniques

Inputs, variables, random integers and outputs in a function

Python

```
import random
def findName(name):
    print('Hello ' + name)
    print('What is your favorite
colour?')
    colour = input()
    if colour == 'yellow':
        print('You shall pass')
    else:
        num = random.randint(0,99)
        while num < 99:
            print('aaarrrghghgh')
            num = num + 1
        print('Splat, you are
splatted ' + name)
name = input('What is your name?')
findName(name)
```

This example starts by importing the `random` set of functions that we will use to generate a random number. We then create a function called `findName` that's expects an argument called `name`. The argument is provided by the input and variable (`name`). The user is then asked what their favorite colour is and a logical test is performed where if they type yellow they get one answer and if they type anything else they get a random amount of 'aaaargh' generated by the `random.randint` and this is used to print the string a random amount of times depending on whether it is less than 99 or not using a while loop. Note how nothing actually happens until the last two lines are interpreted where the input for `name` is taken and the then the `findName` function is called.

```
import random
```

Here is another example where a user is prompted to make a choice. Note the use of !=

```
def intro():
    print('You find yourself in a room for a red and blue door')
    print('On the wall it says \"One door leads to cake the other to certain death\"')

def choice():
    door = ''
    while door != '1' and door != '2':
        print('Which door do you choose?(1 or 2)')
        door = input()

        return door

def checkDoor(chosenDoor):
    print('you turn the handle and the door opens...')
    print('The light in the room turns on and you see...')

    niceRoom = random.randint(1,2)

    if chosenDoor == str(niceRoom):
        print('an empty plate, the cake was a lie!')
    else:
        print('a wafer thin mint...noooooo')

intro()
doorNumber = choice()
checkDoor(doorNumber)
```
	in `choice` (not equal to). Also note how all the functions refer to each other in the correct order and separate out the process sensibly.

Looping through lists

OCR Exam Reference Language

```
array names[5]
names[0]="Ahmad"
names[1]="Ben"
names[2]="Catherine"
names[3]="Dana"
names[4]="Elijah"

for i=0 to 4
    print ("Hello" + i)
```

Python

```
py_chars = ["The Announcer", "Mr
Badger", "Arthur Nudge", "Spiny
Norman", "Eric Praline"]
for chars in py_chars:
    print(chars)

The Announcer
Mr Badger
Arthur Nudge
Spiny Norman
Eric Praline
```

In this example we define a list of Monty Python characters and then loop through the list printing each one.

```
py_chars = ["The Announcer", "Mr
Badger", "Arthur Nudge", "Spiny
Norman", "Eric Praline"]
for chars in py_chars:
    print("I love " + chars +
'.\n")
print("And now for something
completely different")

I love The Announcer.

I love Mr Badger.

I love Arthur Nudge.

I love Spiny Norman.

I love Eric Praline.

And now for something completely
different
```

You can add other things to your loops such as strings, spacing between lines (+"\n').

```
pyChars = ["The Announcer", "Mr
Badger", "Arthur Nudge", "Spiny
Norman", "Eric Praline"]
newChar = "ken shabby"
if newChar not in pyChars:
    print(newChar.title() + " is
not in the list")

Ken Shabby is not in the list
```

In this example we define a new variable with a string of a new character, we want to check if the character is in the list so we loop through the list using `not in` operators. Note also the `.title` method used to capitalise the output string.

Read from a file and write back to it

OCR Exam Reference Language

`myFile = open("sample.txt")`
 	
`myFile.writeLine("Hello World")`
 	
`myFile.close()`
	The file is opened and then a string is added and the file is closed.

Python

The example below requires you to have created a .txt file with some text in it in the Python folder.

`>>> import os`
`>>> os.getcwd()`
`""`
`"C:\\Python34\NEA.py"`
	To work with files it is useful to know the current working directory (cwd) as it is assumed you are using the cwd unless otherwise specified.

`>>> a_file =`
`open("C:\\Python\NEA.txt")`
 	
`>>> a_file_content = a_file.read()`
`>>> a_file_content`
 	
`Waitress: Well, there's egg and bacon; egg sausage and bacon; egg and spam; egg bacon and spam; egg bacon sausage and spam; spam bacon sausage and spam; spam egg spam spam bacon and spam; spam sausage spam spam bacon spam tomato and spam; or Lobster Thermidor au Crevette with a Mornay sauce served in a Provencale manner with shallots and aubergines garnished with truffle pate, brandy and with a fried egg on top and spam.`
	Note I have used an absolute path, you can use a relative path if the file is in the cwd (`open("NEA.txt")`).

`#!/usr/bin/python`
`# -*- coding: utf-8 -*-`
 	
`another_file = open("Ni.txt","w")`
`another_file.write("We are the Knights who say…\n")`
`another_file.close()`
 	
`another_file = open("Ni.txt","a")`
`another_file.write("Ni!")`
`print(another_file)`
`another_file.close()`
	As we are creating text we need tell Python which encoding to use. As I am on a Windows PC I define it as UTF-8. In this example we open a file called Ni.txt which doesn’t exist so Python creates is open opens it in the write mode and then adds a string and then closes it.
	
	Here we open the same file in append mode and then append another string and close it.

We value your feedback

We'd like to know your view on the resources we produce. By clicking on the icon above you will help us to ensure that our resources work for you.