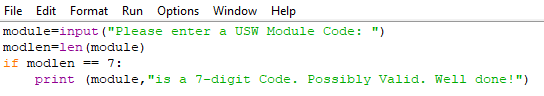
**PYTHON Programming Exercises 3**

In today’s exercises, we will consider some of the essential components that make good computer programs: conditional statements.

**Task 1: IF Condition Statement**

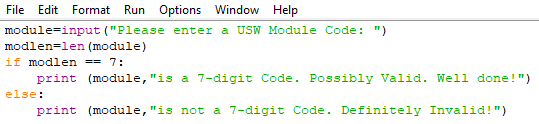
Programming often involves examining a set of conditions and deciding which action to take based on those conditions. Python’s if statement allows you to examine the current state of a program and respond appropriately to that state.

We want to input a University module code as input. We know that these are made up of 7 digits (e.g. IS4S761). From the Python IDLE, create a new file as follows and save as **if1.py**

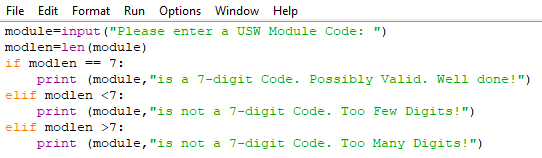


Remember the double equal sign (==) means check if something is equal to something else, i.e. if there are 7 characters input, then print out the statement. The colon at the end of the if statement tells Python that if this statement is true then to do everything after the colon that’s indented. Run this program with a number of different module codes (of 7 or less characters).

We really need to have a condition which takes account of us not entering 7 digits. Update the program to:



And run again with some other module codes. We can also extend the code to consider further conditions with the ELIF condition (else if):



And so on. This is known as the IF-ELIF-ELSE chain (as the last elif could be an else as we have exhausted all possibilities).

**EXERCISE 1:** Adapt this program to test if the module code is an Information Systems module (beginning with the IS prefix). If it is, then print out a message to say so.

**EXERCISE 2:** Write a program to input a person’s age and correctly calculate the Ticket Fare from Treforest to Cardiff (where children under 2 travel free; children under 5 pay £1; children under 16 pay £2; OAPs aged 60 and over pay £3; and everyone else pays £5).

**EXERCISE 3:** Write a Program to input the number of hours worked by an employee, their rate of pay and then calculate and output the total pay. For any additional hours worked over and above 40 hours, pay is calculated at time and a half.

**EXERCISE 4:** Write a Program to find the largest of 3 inputted numbers (x1, x2, x3) using either the IF, IF-ELSE, or IF-ELIF-ELSE statement. There’s a number of ways of doing this, so think about it first.

x1, x2, x3 = eval(input("Please enter three values: "))

# missing code sets max to the value of the largest

print("The largest value is", maxval)

For the record, Python also has a built-in function **max** which calculates the largest of its parameters:

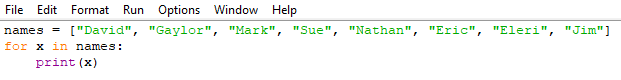
print("The largest value is", max(x1, x2, x3))

but it’s more of a logical challenge using IF.

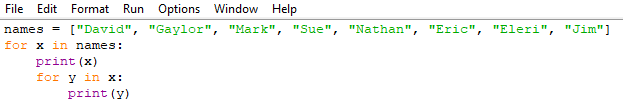
**Task 2: FOR Loops**

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

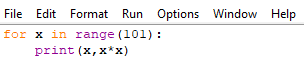
From the Python IDLE, create a new file as follows and save as **for1.py**



This should allow us to print out each name (x) within the list. We can extend this program further to not only print out each name (x), but also print out each character (y) in each name:



To loop through a set of code a specified number of times, we can use the **range()** function. The **range()** function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number. Let’s print out the numbers from 1 to 100 and their squares:



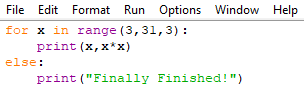
Even though the **range()** function defaults from 0, we can set it to start at other values, and also to increment by different amounts. Run the following:



And



The **else** keyword in a for loop specifies a block of code to be executed when the loop is finished:



**EXERCISE 5:** Write a program to calculate the SUM of all numbers between 1 and 1 million. There’s no need to print out the sum as you go along, just the total at the end. How long did it take to run?

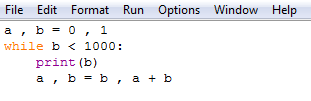
**EXERCISE 6:** A number raised to the third power is called a cube. For example, the cube of 2 is written as 2\*\*3 in Python. Make a list of the first 10 cubes (that is, the cube of each integer from 1 through 10), and use a for loop to print out the value of each cube.

**EXERCISE 7:** Write a Program to allow the user to input ten numbers (one at a time) and calculate and output the average of them (using a FOR loop).

**Task 3: WHILE Loops**

We know that a loop is a piece of code that automatically repeats. The length of the loop is controlled by a conditional test made within the loop. While the tested expression is found to be TRUE the loop will continue. The FOR loop will run a set number of times, so we have more control over it. The WHILE loop can operate in a similar way to a FOR loop (by executing some code a set number of times), or can be set up to run until the condition becomes FALSE and drops out of the loop. It offers more flexibility in our programming than a FOR loop.

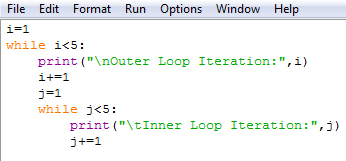
Let’s start by creating a WHILE loop to produce a Fibonacci sequence of numbers (The Fibonacci sequence is a series of numbers where a number is found by adding up the two numbers before it. Starting with 0 and 1, the sequence goes 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so forth). From the Python IDLE, create a new file as follows and save as **fib1.py**



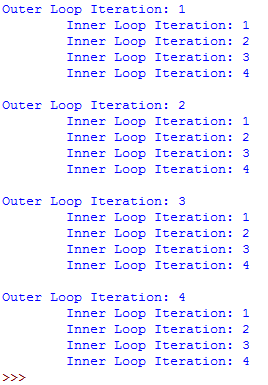
What do you think a,b=0,1 and a,b=b,a+b mean and do? (Multiple variables can be defined with the same or differing values in a single statement using comma separators). Sit down with a pen and paper and work out what the values of a and b will be for the first few passes through the loop.

If you had to write this code using separate assignment statements for a and b, what would your code look like?

As with the FOR loop above (for printing names and individual characters), we can also nest WHILE loops, one within the other. Start a new Python script as follows, save and run:



Which should generate something like this:



Can you work out what each statement means?

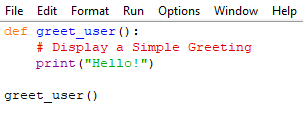
i+=1 and j+=1 are equivalent to i=i+1 and j=j+1 respectively. In the print functions, \n and \t correspond to New Line and Tab respectively. These are useful for formatting output to the screen to make it look better. Try running the code again without the \n and \t parts?

**EXERCISE 8:** As for Exercise 7 above, Write a Program to allow the user to input ten numbers (one at a time) and calculate and output the average of them (using a WHILE loop).

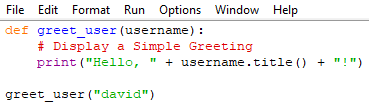
**Task 4: Creating Functions**

A function is a command that you enter into Python to do something. It’s a little piece of self-contained code that takes data, works on it and then returns the result. We’ve already used some of Python’s functions, such as **len(), list(), print()** and **type()**. Basically, a function is a short word followed by brackets which enclose the data for the function. Let’s create some of our own:

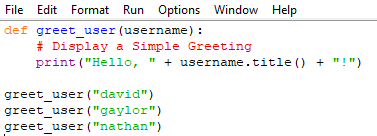
From the Python IDLE, create a new file as follows and save as **greet.py** and then Run



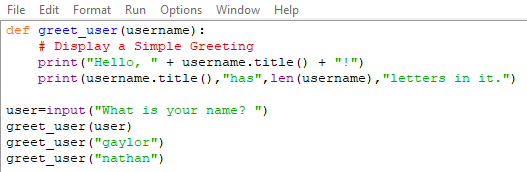
Now edit the above program and save as **greet1.py** and then Run:



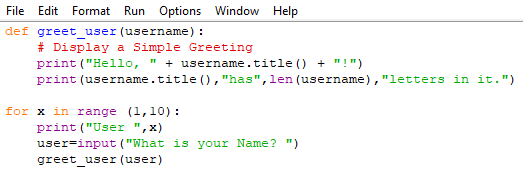
The function can be used over and over. Save and Run the following as **greet2.py**:



And can be extended as we see fit. Edit, Save and Run the following as **greet3.py**:

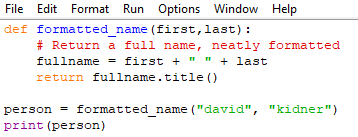


Finally, we can embed the function call inside a loop. Edit, Save and Run the following as **greet4.py**:

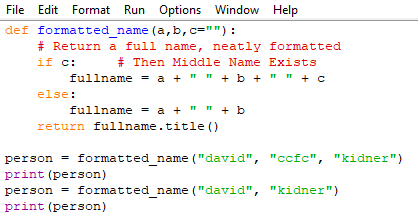


**Exercise 9:** Write a function called **describe\_city()** that accepts the name of a city and its country. The function should print a simple sentence, such as Reykjavik is in Iceland.

We can also use a function to return a value as a variable. In this example, we want to take as input a first name and a last name and output a neatly formatted full name. Create and run the following program with your own names and save as **name.py**:



What if we want to include our Middle Name? What if we haven’t got a Middle Name? Think about how we can adapt name.py to include an optional middle name? In a function call, we can make an argument optional. If we assume that the user will either enter two or three names, then the third input is optional. We can do this by accepting that the third input could be an empty string, and then use the if condition on this value to test whether it has been entered or not. Input and save the following as **name1.py**:



**Exercise 10:** Write a function called **greet\_name()** that accepts as input a list of names (see below) and prints a simple greeting message to them each in turn (and specifies the number of letters in each of their names.

names = [“David”, “Gaylor”, “Mark”, “Sue”, “Nathan”, “Eric”, “Eleri”, “Jim”]