**Introduction**

As we start writing larger programs with more and more lines of code, finding errors and editing the code can become more and more difficult. We can get lost in the hundreds of lines of code.

To overcome this, ‘sensible programmers’ make use of subroutines. Using subroutines provides a structured approach to programming, where code is modularised, allowing for easier debugging and improving the overall organisation of the program’s logic.

**Subroutines**

A subroutine is a block of code, that has been given a unique name and that will only execute (be run) when it is called to do so.

The diagram below attempts to demonstrate this. The subroutine named ‘timestable’, is only executed after it has been called. When it is called, the main program pauses whilst the subroutine runs. Then, after the subroutine has finished executing its code, the main program resumes.



**Subroutine Types**

There are two types of subroutine:

**Procedures** & **Functions**

The main difference is that a function will return values back to the main program after it has been called, whereas a procedure will not.

However, this may make little sense until we explore both procedures and functions in greater depth…

…so, let’s now take a look at how to program these two types of subroutine in python.

**Setting up a Procedure**

Procedures sit at the top of our code.

They are declared using the ‘**def’** command.

They are then given a name.

Notice the brackets and colon!

E.G.: **def procedure1():**

The code for the procedure is then written ‘indented’ underneath.

**An Example of a Procedure**

When programs with procedures are run, the procedures are ignored and the main program is executed first. When a procedure is called in the main program, the main program code is paused and the procedure is executed. After the lines of code in the procedure are executed, the main program code execution is resumed.



**Local and Global Variables**

In programming there are two main classifications of variables:

**Global** & **Local**

A global variable is one which is defined in the main program.

A local variable is one which is defined in the subroutine.

Here, the local variable is being printed first, due to the subroutine call occurring before the print() statement.

Then, the global variable is being printed second, due to the print statement occurring after the subroutine call.

If a subroutine doesn’t assign its own variable, the subroutine will use the global variable.

Here a global variable is first being assigned, as the assignment is the first line of the main program.

Then, the subroutine is called.

And because the subroutine doesn’t assign its own variable, it uses the global variable ‘var’.

**Parameter Passing** *(passing values into subroutines)*

As we have already seen, we can use subroutines to better organise/structure our code, so that finding errors and editing the code can become easier. This is because major sections of our code can become self-contained and separate from one another.

However, the way we have written subroutines so far has meant that any data that it is to process, needs to be created inside the subroutine.

But this is not always ideal. Often, we will have data in our main program, that we want a subroutine to process! So how can we move data from the main program to a subroutine for processing? Let’s take a look.

If we wanted to write a program, which printed the 10 times table, we might write a procedure like this:



But what if we wanted this program to allow the user to select a times table of their choice? Using our current knowledge, we would have to create a procedure for each times table and then ask the user to select their choice!

Here, you can see that the use of procedures written in this way has a drawback.

Although it is great that our code is self-contained, the issue is that the code is being duplicated (with only minor changes)

Programmers like efficiency, and duplicating code is not efficient!



In order to be more efficient, what we really want is to have one procedure to do the multiplication and then PASS the user’s timestable choice into the procedure.



This procedure demonstrates this improved logic. As you can see, the procedure above now takes on the value passed to it from the main program. A value is being passed into it for processing. The variable concerning the passed value is known as a parameter and this process is called parameter passing.

The value in choice is being passed into a variable/parameter called number, in the timestable function.

*It is important to realise that the variable name of the value being passed, doesn’t have to match the destination variable name.*

**Introduction to Functions**

All subroutines that we can studied so far have been procedures, but we will now turn our attention to functions. A function is a subroutine that can return data back into the main program. This can be highly useful in programming!

Coding functions is almost identical to programming procedures, but with 2 major differences:

1. A function call must be assigned to a variable
2. The function must have a return command

**Example of parameter passing and returning values – *tracing the data flow*:**

Here the arrows show how data is being passed from the main program into the function’s parameter ‘x’. number is as a parameter and contains an argument (an item of data), which is passed to the function’s parameter (x).

When the function is finished executing its commands it returns the value back into the main program. The value is returned into the variable that the function call is assigned to (in this case it is ‘answer’).

**Passing more than one value (argument) and Returning a value (argument):**

Here the arrows show how 2 parameters (each containing arguments) are being passed from the main program into the function’s parameters ‘x’ and ‘y’.

It is important to realise that the names of the parameters passing the values and the parameter names of the function, do not have to match. However, the order in which they are sent is the order in which they are received (number1 is sent to x, number2 is sent to y).

**Passing and Returning more than one value (argument)**

Here, the arrow highlighting the ‘return’ is showing how 2 pieces of data can be returned into the main program. Of course, many more could be returned in one go in this same manner.

As these values are being returned to one identifier ‘answer’, it automatically becomes a tuple, which is a data type that can hold a number of different values (like a ‘list’, but tuples are not editable (immutable)).

**PRIMM TASKS**

**1: Predict**

|  |  |
| --- | --- |
| **Code** |  |
| **Predictions** | **What do you think the code will do?** |
|  |
| **Write the program’s output exactly as you think it will appear.** |
| **>>>** |

**2: Run**

|  |  |  |
| --- | --- | --- |
| **Now type in the code and run the program.**  **Show a screenshot of the actual output below.** | | |
|  | | |
| **Was it the same as your prediction?** | Yes | No |
| **Were there differences? If so, show or describe these below.** | | |
|  | | |

**3: Investigate**

|  |  |
| --- | --- |
| **What happens if the code is altered to the following?**    **Explain result.** |  |

**4: Modify**

|  |  |
| --- | --- |
| **Modify the original code so that it loops in the reverse order (descending).**  **Place a screen shot of your code opposite and explain what the code is doing.** |  |

**5: Make**

**Create a program in python for each point below and add a screenshot of your code in the boxes provided:**

**1. Create a program which when run, calls a procedure that prints a statement of your choice to the screen.**

|  |
| --- |
|  |

**2. Create a program which has a menu system that gives the option of selecting one of three procedures:**

* **When this procedure is called, the procedure prints the 10 times table**
* **When called this procedure tells you how you are feeling by picking a statement (from a set that you have added) at random**
* **When called this procedure will display your name and address (try programming this with both local and global variables)**

|  |
| --- |
|  |

**3. Create a program which has two functions which can each be chosen from a menu by the user:**

* **The first will receive the user’s age and calculate how old they will be in 50 years’ time and then pass this data back into the main program for printing;**
* **The second will receive the user’s annual wage and the user’s savings, convert wages into euros and savings into US dollars and then return the two calculated values back into the main program for printing.**

|  |
| --- |
|  |