In this tutorial, we are going to develop a simple program in Python 3 using the IDLE (Integrated Development and Learning Environment) interface. This tutorial assumes that you already have the IDLE packages previously installed. To start, open IDLE on your computer using the shortcut created upon installation, or if you are using Windows, you can simply open the search menu and open up the program.

This is the IDLE interface. It is used to run Python programs and comes with various editing and debugging features. For this exercise we will be developing a simple program that accepts a user’s name, their age, and outputs their estimated age after a 5 year period. To start, select the [File] button in the top left corner of the toolbar and click [New File]. This will open a new window where we can begin writing our Python script.

Our first line in the script will be utilizing the ‘print’ keyword, a function that outputs information to the screen. It should be noted that in this tutorial, anything referred to as ‘input’ pertains to information the user enters from the keyboard, and anything referred to as ‘output’ pertains to anything the program displays when it runs. The ‘print’ keyword is one function that handles output. To denote the start of our script, we will type “PROGRAM STARTED” as a parameter to be fed into the ‘print’ function.

print("---PROGRAM STARTED---")

This will display the text in the quotations to the screen.

Next, we write our ‘input’ function. In contrast to the ‘print’ function, the ‘input’ function is used for gathering user input, whereas the ‘print’ function handles program output. Since our program is going to accept a user’s name and age, we need to write two ‘input’ functions each.

input("Please enter your name: ")

input("Please enter your age: ")

The statements in the quotations will be displayed onscreen, however, the ‘input’ function will wait for the user to enter their response on the keyboard. So looking at these simple function calls, the phrase, “Please enter your name:” will be displayed first, and then the program will wait for the user to enter their name before displaying the next prompt, “Please enter your age:”.

Now that we have our input, we also need a way to store that information in memory so that the program can access it later when we eventually write our output. To do this, we define a couple of variables. Variables are placeholders used to represent specific values, and these values can range from being numbers to a set of characters or strings. For this exercise, our variables are going to represent string values, which essentially mean that they will contain words, which makes sense in the context a person’s name. An age should be represented as a number, but we will get into more of that soon. For now, we define two variables, ‘name’ and ‘age’, and assign them to each of our previous ‘input’ functions respectively.

name = input("Please enter your name: ")

age = input("Please enter your age: ")

This will ensure that the user’s inputted name will be stored in our ‘name’ variable, and the user’s age will be stored in our ‘age’ variable.

Our goal is to calculate the user’s age after a 5 year period. Now that we have their age stored in memory, we can now access it and modify that value before assigning it to a new storage location. Our new storage location will be represented as another variable denoted as ‘newAge.’ To calculate the user’s age after 5 years, we simply have to take their inputted age and add the value 5 to it. We will store this modified value in ‘newAge.’

newAge = age + 5

This line is not yet complete however, as this statement will not execute without a compiler warning. This is because our ‘age’ variable that we got from our ‘input’ function is stored as a string and not an integer value (a number). The compiler will not understand what we are trying to do if we take the integer 5 and add it to a string (a series of characters). Thus, we must convert ‘age’ into an integer in order for the mathematical operation to be valid.

newAge = int(age) + 5

The modified age is now stored in ‘newAge’. From here, all we have to do is output the new age value using the ‘print’ function. In addition to handling specific output, the ‘print’ function can also accept variables and output their values to the screen as well. We do this by putting a comma after our explicit phrase, followed by our variable whose value we wish to output.

print("Hello", name)

This line will output the phrase “Hello [user’s name]” to the screen.

We can do the same thing with the user’s initial age.

print("You are", age, "year(s) old.")

And then finally, the user’s modified age.

print("In 5 years you will be", newAge, "years old.")

So these three lines, when executed, should output the user’s name, age, and age after 5 years to the screen in sequential order.

Lastly, to signify the end of our script, we write a simple ‘print’ function to denote our closing similarly to our very first ‘print’ statement.

print("---PROGRAM END---")

These ‘print’ statements denoting the start and end of our program are not necessary, however, they will make it easier to identify when and where our program begins and finishes when the script is run.

To test our program, click on the [Run] button on the toolbar near the top of the window, and then click [Run Module]. Alternatively, you can just press the F5 key on your keyboard. Now you can test your program by providing some input through the keyboard and awaiting its output.

To summarize, in this tutorial we developed a small program that calculates a user’s inputted age after 5 years. Our program utilizes the ‘print’ and ‘input’ functions for handling output and input respectively. The user’s input is stored in local variables and modified in order to arrive at our solution. Our solution is then finally outputted to the screen. That concludes this introductory Python tutorial.