In computer programming, the standard method of communication between the computer and the real world is known as input/output, also abbreviated as IO. Input/output allows users to communicate with computers, as well as observe results that the computer may try to display to them. In this exercise, we will cover two basic input/output functions built into Python 3, as well as explore some of their uses.

Our first function is known as print(), and simply displays whatever value is passed into it to the screen. This value can be either numerical, or a string literal. Some examples include:

Print(“Hello world”)

Print(5)

Print(“5”)

Print(“Test print”, 4)

Print(“Hello {} my old friend”.format(“darkness”))

Print is the standard output function, so named because of how it can be used to display values or results to the screen.

Out other function, this time for handling input, is known appropriately as input(). Despite its name, the input() function does in fact utilize a form of output in its usage. Similarly to print(), input() accepts any value that is numerical or a string literal. This value is then outputted to the screen, again much like print(). The difference between the two however, is that after outputting, the input() function will also wait for the user to provide a form of input through the computer keyboard. The function will not complete or return a proper value until an input is read. Some examples as follows:

Input(“Please enter your name:”)

John

Input(“Please enter your age:”)

40

As you can see, the input() function outputs the string values we pass into it. The value that is entered after is arbitrarily up to the user, and defines the value that is returned by the function upon completion. We can take this value and store it in a local variable that can be used at other points in our program.

Name = input(“Please enter your name:”)

John

Print(“Hello”,name)

Age = input(“Please enter your age:”)

40

Print(“You are”,age,”years old.”)

So in the input() function, a three main steps occur in this order:

First, the function will display the value we pass into it to the screen.

Second, it will then wait for the user to enter a value through the keyboard.

Finally, if possible, the value that the user enters will be stored in a variable that we assign it to. Here’s one more example to demonstrate this:

Day = input(“What day is it today?”)

The function displays our question to the screen.

Now the function waits for us to enter a value.

Monday

This value is then assigned to the variable we gave it. In this case, our *day* variable.

Print(day)

An important thing to note about the input() function is that values that are returned from it are treated as string literals. This means that doing this:

Age = input(“Please enter your age:”)

40

nextAge = age + 1

Is invalid.

Even though we entered a numeric value, we are not allowed to perform mathematical operations on it, for our value is actually a string value. Instead of 40, the value stored in age is really “40” with quotations. If we wanted to perform mathematical operations on a numeric value entered by the user, we would have to take advantage of one more function—not related to input/output, but nonetheless useful for problems like this.

The int() function attempts to convert anything we pass into it into an integer. The new integer value is then returned by the function. Some examples are:

Int(4.2)

Int(“6”)

int(2.9)

Note that the int() function does not round the integer value. Any decimals that are in the pass parameter is simply truncated instead. This technique of converting one datatype value to another datatype value while at the same time preserving the data is also known as “casting.”

We can use the int() function in conjunction with the input() function to cast our age result from before into a proper numeric value.

Age = input(“Please enter your age:”)

40

Age = int(age)

nextAge = age + 1

print(nextAge)

Now our age is treated as an integer, and so we can perform mathematical operations on it. We can also shorten this segment of code by combining the int() and input() functions into a single line like so:

Age = int(input(“Please enter your age:”))

40

nextAge = age + 1

print(nextAge)

In this example, out input() function resolves before our int() function due to the nature of the bracket structure. Thus, our input() function will run through all three steps like earlier before taking the value and passing it into our int() function. Our int() function will then attempt to convert out value into an integer datatype before finally assigning it to our specific variable.

To recap, input/output is used as a way of communication between a computer and the real world. Basically, input comes in the form of users typing in values that the computer must be able to read and understand. Likewise, output is defined as anything that the program displays in order to communicate a message or result. The two main functions for handling basic input and output are known as input() and print() respectively. The input() function utilizes three main steps in its execution. First, it displays the values that are passed into it (normally a message prompt, but this is up to the programmer). Next, it will wait for the user to enter a value. Finally, this value is returned by the function and can be assigned to a variable. This value is always of the string datatype, but can be casted to an integer by using the int() function. Lastly, the int() function and the input() function can both be used on the same line by wrapping the entire input() function and its parameters into the int() function. The value returned from this will always result in an integer datatype.