Aaron Eberly

CNIT 15501

Week 6 deliverable

Functions

If you find yourself repeating a lot of code in a program, it is probably more convenient to use a function. You can define a function, which is a block of code that you can call. Whenever you call that function, it will execute all the code inside of that function.

To create a function, use the following line:

def functionName():

Note that you can name the function whatever you want (within python’s naming restrictions)

Make sure to indent all of the code within the function.

Observe the following code:

def summer(N):

    sum = 0

    for num in range(1,N+1):

        sum += num

    print(sum)

summer(7)

In this code, the line summer(7) is where the summer function is called.

Notice the N inside of the parenthesis on that first line. This is called an argument or parameter. A parameter will be defined when the function is called. In this example, since summer(7) is called, the value of N would be 7.

If I instead called summer(9), the value of N would be 9.

Note that a function can have multiple parameters. When a function is called, ALL parameters must be defined.

def summer(N, step):

    sum = 0

   for num in range(1,N+1, step):

        sum += num

  print(sum)

summer(7, 2)

In the above code, 7 corresponds to N, and 2 corresponds to step.

When there are multiple parameters, they must be listed in the correct order.

Returning Values from a Function

In a function, you may return a value. The syntax to return the number 4 would be as follows:

return 4

Alternatively you can include parenthesis:

return(4)

When you return a value, the function immediately ends. Then, the function evaluates to the thing you returned. You can return many things, including integers, strings, doubles, tuples, lists, and more.

You can also return a variable.

Here is an example of returning:

def isBelowTen(num):

    if(num < 10):

        return True

    else:

        return False

num = 7

returnVal = isBelowTen(num)

print(returnVal)

The code will print True. Since the fuction returned as true, the variable returnVal = True.

If num had instead been set to 11, False would have been printed instead.

Global vs Local Variables

When you define a variable outside of a function, it is a global variable. These variables can be used inside any function. However, if you define a variable inside of a function, then you can only use it in that function. This is called a local variable. If you want to change a global variable while inside of a function, you must first use the global keyword:

global num

In the above case, num is a variable name. Now, if you change the variable num in that function, it will change it globally. If you do not use the global keyword before changing the variable, the variable will only be changed for that function.

def bonusCalc():

    amount += 10

amount = 20

bonusCalc()

print(amount)

The above example will result in an error. Since amount is not a global variable, it can’t be changed within the bonusCalc function.

The following example will make amount a global variable:

def bonusCalc():

    global amount

    amount += 10

amount = 20

bonusCalc()

print(amount)

The code will print 30, as expected. Since amount was declared to be global, it’s value can be changed in the function.