**Creating Functions in Python**

A ***function***is *a section of code that is written to accomplish a specific task*. You have used a few Python functions already:

print() # sends output to the monitor

input() # gets input from the keyboard

int() # converts a value to integer

float() # converts a value to floating point decimal

str() # converts a value to string

As you can see, all functions end with round brackets. The brackets are an important part of function notation. If you have not learned about function notation in math class, you will see the meaning of the brackets soon.

In math class we learn about the sine function:

y = sin(30)

This means “put the value 30 into the function and take the sine of that value. Put the answer into the variable y.”

The value 30 is called the ***function*** ***argument***. It is the value that you put into the function.

The value that the function calculates (i.e. “the answer”) is called the ***return*** ***value*** of the function. In our example the return value is saved in the variable y.

The sine function does not have to take a value of 30, but it must take *some* value. The general form is:

y = sin (x)

where x is some number (a float will take care of all situations). The variable x is called the ***parameter*** of the function. As you may know from math already, a function may have many parameters, but only one return value.

We can make our own functions using the ***def*** keyword (short for ***define***):

def double(x): # define a function called “double”

return 2\*x

This is about as simple a function can be. All it does is double the number. How do you use it?

Here’s how we define and call our new function:

#------------ define our function -----------------------#

def double(x): # define a function called “double”

return 2\*x # return the answer

#------------- our main program ----------------#

y = double(25) # call the function and put the answer in y

print (y) # print the answer

It’s pretty basic as functions go, but it works! What is the:

* function name
* parameter name
* function argument:
* return value

Answers: The function is named double. The parameter is x. The argument was 25. The function return value was 50, in this case.

Here are some more function calls:

def double(x): # define a function called “double”

return 2\*x

num = 33

p = double(num) # call the function

print (p)

print (double(12)) # call the function again

In the first call the argument is 33 and the return value is 66. In the second call, the argument is 12 and the return value is 24.

**Functions with No Arguments or No Return Value**

A function does not need to return a value or take an argument necessarily. Here is an example:

print()

This example takes no argument (the brackets are empty) and returns nothing. But it still does a job - it prints a blank line.

Here is an example of our own that has no parameters and no return value:

def happy():

print ("Let's be happy!")

return # no return value

happy() # call the function

happy() # call the function again

happy() # one more time!

Notice that the brackets are still required, even though they are empty. The brackets lets the interpreter know we are referring to a function called happy(), not a variable called happy.

A function that returns no value is called a **void function** or a **None function**.

Remember that a function needs to be defined, and then called at some point. The following example doesn't do anything. Can you see why?

def sad():

print ("Boo Hoo!")

return

To make a function do its job, you need to call it, like this:

sad() # call the sad() function

**Functions with Multiple Arguments**

You can create a function that takes multiple arguments. Here is an example:

def sum(a,b,c):

return a+b+c # return the answer

s = sum(2,3,4) # call the function with 3 arguments

print (s)

What are the:

* arguments:
* parameters:
* return value (in this example):

Answer: In this case, the arguments are 2, 3 and 4. The parameters are a, b and c. The return value is 9.

A function definition must be placed before the function call. What happens in this example?

y = add(3,4)

print (y)

def add(a,b): # function definition begins here

return a+b

Answer: This generates the error code:

NameError: name 'add' is not defined

Functions need to be defined *before* you call them. Can you fix the problem?

**Exercises**

13.0 Consider the following code. Do not run it. The line numbers are for reference:

1 def stuff(a,b,c)

2 x = a + b \* c

3 return x

4 m = 3

5 n = 2

6 p = 5

7 y = stuff (m,n,p)

8 print (y)

In the above example:

1. What line number does the function begin on?
2. What line number does the function end on?
3. What are the arguments to the function?
4. What are the parameters of the function?
5. What is the name of the function?
6. What type of data does the function return (boolean, float etc)?
7. What line number is the function call?
8. Create a function called **cube()** that returns the cube of a number. The function does not print anything – if you want to print the value, you have to do it outside of the function. Test your function with these function calls:

x = cube(10)

print (x) # should print 1000

print (cube(45)) # 45 cubed is 91125

t = 2.5

print (cube(t))

1. Create a function called **printMoney()** that takes an argument and prints its value with a dollar sign in front and rounded to two decimal places. The function returns no value. Here’s how the call would look:

x = 3.56859

printMoney(x)

>>> $3.57

Note: You’ll need to review this clip from lesson 3 to remind you how to format a printout (see below):

c = 4.777777

print ("$%.2f" % c) # puts a dollar sign and rounds to 2 decimal places

Test your function with the following calls:

printMoney(4.10)

>>> $4.10

printMoney(0.5\*5)

>>> $2.50

printMoney(10/3)

>>> $4.10

printMoney(8)

>>> $8.00

1. Create a function called ***hypo*()** that calculates the hypotenuse of a right triangle given *two* arguments – the lengths of the two sides – and then returns the answer. Have the main code ask for two side lengths and then print out the value of the hypotenuse as shown below. Format your printout so it rounds to one decimal place (look at previous question.

Note: You will need to import the math module so you can take the square root to get the hypotenuse. Put this at the top of your code:

import math

and then use the square root function, like this:

blah = math.sqrt(16) # calculates the square root of 16 and stores in blah

Enter the length of one side: <7>

Enter the length of the other side: <12

The hypotenuse is 13.9

1. In the previous example, the function returns the answer, but does not print anything. Why is a function that prints out the answer not as useful as a function that just returns the answer and doesn’t print anything?

13.5 In the same vein as the previous question: A rule of thumb for functions is “One function, one task”. Why is it a bad idea to make a function do too many things?

Keywords: ***function, function call, parameter, argument, return value, void, none***