Lab 5 Fruitful Functions

February 27, 2019

1 ME 30 Lab 5 - Functions and Style

Description and Summary: >A function in programming is a block of code that performs some task. Functions are a way to organize and simplify your program to allow for easy debugging and reading later on. For instance, print() is a built-in Python function that allows the programmer to easily print something to the screen without writing extra code. You may find it easier to think of functions like in math; for example, y = mx + b. The equation of a line is a function of x, where x and y are inputs and outputs, respectively. Programmers are encouraged to make their own functions on top of the built-in functions already provided by Python.

It is best to read chapters 3 - 5 in *Think Python* before starting this lab.

1.1 Anatomy of a Function

Consider the linear equation function mentioned earlier. This function can be implemented in Python like:

```
In [ ]: # Linear equation function definition:
        def y(m, b, x):
            y = m * x + b
            return y
```

Copy the code above to a code cell in your Lab Report notebook and press Shift-Return. Use the function by placing the following lines of code in a new code cell and executing the cell by pressing Shift-Return.

```
In [ ]: # Use the function to calculate y at x = 10 with m = 2 and b = 0:
```

Note the following things in the function definition:

1. The definition begins with the keyword `def`
2. The name of the function is 'y'. Be aware of naming conventions: https://www.python.org/dev/peps/pep-0008/#naming-conventions
3. The function has three parameters (i.e., *references* to the pieces of data that are input to the function) that are enclosed in parentheses.
4. The first line ends with a colon

5. The second line forms an expression using the parameters
6. The last line returns the value of y when the function is used

Note the following things when the function is used:

1. The function is executed ('called') by typing its name and passing it three arguments (actual values corresponding to the parameters)
2. The calculated value of y = 10 is returned.

1.2 0. Warmup Example

The code cell below implements a function to add two variables, but contains lines that refer to functions Subtract, Multiply, and Divide that have not yet been defined. Run the code and select choice 1 to verify that the addition function operates properly.

```
In [ ]: #Example:
        #From: https://www.programiz.com/python-programming/examples/calculator
        #This function adds two numbers
        def add(x, y):  #parameters = x, y
           return x + y

        print("Select operation.")
        print("1.Add")
        print("2.Subtract")
        print("3.Multiply")
        print("4.Divide")

        # Take input from the user
        choice = input("Enter choice(1/2/3/4):")

        #make sure input is an integer only
        num1 = int(input("Enter first number: "))
        num2 = int(input("Enter second number: "))

        #if the user entered 1:
        if choice == '1':
           print(num1,"+",num2,"=", add(num1,num2)) #this is called a 'function call for add f
        elif choice == '2':
           print(num1,"-",num2,"=", subtract(num1,num2)) #subtract function call with argument
        elif choice == '3':
           print(num1,"*",num2,"=", multiply(num1,num2))
        elif choice == '4':
           print(num1,"/",num2,"=", divide(num1,num2))
        else:
           print("Invalid input") #built-in function = print(). argument = "Invalid input"
```

1.3 Exercise 1

Following the pattern given in the warmup example above, create functions, 'Subtract', 'Multiply', and 'Divide'. Run your code and verify that it works. For example:

```
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4):1
Enter first number: 1
Enter second number: 2
1 + 2 = 3
```

When making a function, it is not required to include paramters, for instance:

```
def no_paramters():
    """This function will print numbers to screen when called"""
    print(5*5)

no_paramters() #this is a function call
```

Notice that the function runs even though we did not include any arguments between the paranthesis. **Parameters** refer to the variables inside paranthesis that is in the function definition. **Arguments** refer to the input when calling a function in a function call. **Docstrings** are another important part of functions, in between triple quotes (""" """) type what the function does. This is good programming practice to follow as well as commenting your code.

```
In [ ]: def user_defined():
            """this is a doctsring, explains what the function does"""
            print('hello')
            print(' everyone')

        user_defined()
```

Python has its own built-in functions for doing certain tasks. For example, if we want to collect input from the user, we would type:

```
x = input('How old are you?')
print(x)
```

Type and run the above statements of code in a separate code cell.

2 Exercise 2

Make a simple function that asks the user for input when it is called. Exercise requirements:

1. Make a function that asks the user to enter a number
2. The function will return the type of the number the user enters when it is called.

3 Exercise 3

Make a function with two input parameters of integers. The function will compare the two parameters and return the number that is larger.

Now that we have seen that each function performs some task, lets look at a group of similar functions called, **modules**. Chapter 3 in *Think Python* has already introduced the **math** module and basic functions inside it. Lets look at other Python 3 built-in modules.

```
import statistics

statistics.mean([1, 2, 3, 4])
```

or

```
from statistics import *

mean([1, 2, 3, 4])
```

To learn more type:

```
dir(statistics)
```

Functions can also be called within other functions, *Think Python* already explains this concept in chapter 3. It is expected that you have read Chapter 3 in *Think Python* before working on this assignment.

```
In [ ]: def hello(name):
            """This function will print hello name"""
            print('hello', name) #built-in print function call inside our own function

        hello('ME 30') #
```

4 Exercise 4 - Personalized Calculator

Make a program that consists of five functions. Each function should relate to the courses you are taking this semester, all five functions can relate to only one course as well. Each function will perform some operation and return a value based on input arguments. Look at the examples below to get an idea of the functions you can make. Import modules (i.e., math, statistics, etc.), try Googling to find one that works for your case.

Practice calling functions inside other functions (further explained in chapter 3).

```
In [ ]: def circle_area(radius):
            """"this function find the area of circle based on radius, r.
            circle_area(radius, pi=3.14)"""
            pi = 3.14
            y = pi * (radius**2)
            return(y)
        circle_area(5)
```

```
In [ ]: #Physics - Classical Mechanics example
        def final_velocity(initial_velocity, acceleration, time):
            """final_velocity = initial_velocity + acceleration * time"""
            y = initial_velocity + acceleration * time
            return(y)

        final_velocity(0, 3, 2)
```

5 Summary:

User-defined functions include:

1. keyword **def**
2. begin with letter (follow PEP8 coding style: https://gist.github.com/sloria/7001839)
3. Open paranthesis, include parameters, close paranthesis
4. End function definition with a **colon (:)**
5. Press return, notice the indentation (tab or 4 spaces)
6. Include a **docstring** that states what the function does.
7. **return** statement to end function block

5.1 References

1. Python Functions Tutorial: www.datacamp.com/community/tutorials/functions-python-tutorial
2. Python Functions: https://www.w3schools.com/python/python_functions.asp
3. PEP8: https://gist.github.com/sloria/7001839
4. Python Book: *Illustrated Guide to Python 3* by Matt Harrison
5. Python Book: *Think Python* by Allen Downey