Lecture 7: Introduction to Computer Programming Course - CS1010

DEPARTMENT OF COMPUTER SCIENCE | 08/24/2019

Rensselaer

Announcements

- Late Homework Policy: There are 6 late days in all. Use these wisely, you might need them later in the semester.
- Homework 3 is posted.
- Due next week on Tuesday.
- Homework 2 is due tonight
- Exam 1 next week (October 1st): Will be a 90 minutes exam
- Will be in-class
- Next class is a review class
- Back Exam is posted

Goals for Today

- Functions
  - Built-in functions
  - User-defined functions
  - Problems
- Methods

Functions

- Definition: A function is a set of statements that
  - takes input,
  - does some specific computation and
  - produces output.
- The reason behind creating a function is to put the code for some task that ‘repeats’ in one place, instead of writing the same code again and again for different inputs.
- Once you have function all you need to do is call it!
- Using a function is called ‘calling’ it.
  - Syntax: function()
- Within the parenthesis: Pass data/values called ‘Arguments’ (can have none, one or more)
  - Syntax: function(argument1, argument2…)

Built in functions

- **Real world analogy**: I bought a car that came with a radio installed in it. I can use the radio (I know how to play it/or use it) very well. I don't need to know how it works (internal technical details).

- We know that Python has a number of built-in operators (such as + for addition, - for subtraction, * for multiplication, and so on), it also comes with a number of *built-in functions*.

- We have used a number of these built-in functions in our code:
  - print()
  - input()
  - float()
  - int()
  - str()
  - bool()
  - len()

User Defined Functions

- **Real World Analogy**: You want to learn to use a hammer. You can break down your learning into the following steps:
  - Grip the hammer by the handle,
  - Hold the nail perpendicular to the surface,
  - Tap the nail with the head of the hammer to get it started,
  - Then hit the nail harder, and so forth.
- Once you understand the steps involved in using a hammer, you can apply your hammer skills any time a set of instructions calls for you to use one, without having to worry about the details.
- Creating detailed low-level descriptions of steps (like how to use a hammer) is very similar to the way functions are embedded in a code.

Definition of Function

- **Definition :** A *function* is a series of related steps (statements) that make up a larger task, which is often called from multiple places in a program.
- Here is the generic form of a function in Python:
  - def functionName(optionalParameters):

    # the 'body' of the function
  - notice the parentheses and the ending colon indented statement(s)
- Let’s build our first function in Spyder that prints our grocery list!

Calling a user defined function

- Similar to built-in functions, when you want to use a user-defined function, you call a user-defined function by specifying the name of the function, followed by a set of parentheses, and include any data (arguments) that you want the function to act on, as follows:
- functionName(argument1, argument2, ...)
- Because our getGroceries function does not utilize any data, therefore to call the getGroceries function, we only need to specify its name, followed by an empty set of parentheses:
- getGroceries() # calling the function, must have parentheses, even if there are no arguments

Data in a User-Defined Function

- Our getGroceries function is a good example of what a user-defined function looks like, but it's not very useful.
- Every time you call getGroceries, it does the exact same thing:
  - this is an example of what is known as *hard-coding*.
- Let’s modify the getGroceries function to use one parameter.
- Instead of always printing *milk* as the first item in our grocery list, we want to allow the caller to call getGroceries and pass in one item to get.
  - Whatever the caller passes in should be printed as the first item.
  - Can further modify to include more than one user provided arguments.

User defined function

- The order of the arguments and the parameters is important.
  - Value of the first argument is given to the first parameter, the value of the second argument is given to the second parameter, and so on.
- Number of arguments in a call must match the number of parameters in the called function.
  - If these don't match in number, Python will generate an error message.

Build functions

- Let's build a slightly more useful example.
- We will build a function whose purpose is to accept a numeric parameter, add two to it, and print the result.
- Keywords
  - For returning output from a function: Can use the keyword ‘return’
  - The special syntax **args* in function definitions is used to pass a variable number of arguments to a function. It is used to pass a non-keyworded, variable-length argument list.

More problems

- 1. Write a Python function that gives the sum of two numbers.
-

Problem 2

- 2.Write a Python function to check if a number is odd or even.
- # A simple Python function to check

Problem 3

- 3.Write a Python function that finds the maximum of 2 numbers.
- Do not use the max function.

Problem 4

- 4. Write a Python Function that finds the maximum of 3 numbers.
- ##Max of 3 numbers

Problem 5

- 5. Write a function that checks whether a number is in a given range (inclusive of high and low)

Methods

- Method is called by its name, but it is **associated to an object** (dependent).
- A method is **implicitly passed on the object** on which it is invoked.
- It **may or may not return any data.**
- A method **can operate on the data (instance variables) that is contained by the corresponding class.**

Syntax

```
# Basic Python method
class class_name
    def method_name () :
        ......
        # method body
        ......
```

- ##User defined method:
- class XYZ:
- def method_xyz:
- print(‘I am in xyz’)

- Call:
- Class_ref=XYZ()
- Class_ref.method_xyz

Python on Built-In Method

- Python comes with built in methods (math, strings)
- import math
- ceil_val = math.ceil(15.25)
- print("Ceiling value of 15.25 is : ", ceil_val)
- Will learn more built in methods with Lists (object type)

Additional Resources

- https://docs.python.org/3/library/math.html
- https://docs.python.org/3/library/string.html

Next Class

- Scope of variables
- Practice more problems :
  - Functions and Methods