CSE 1321L: Programming and Problem Solving I Lab

Lab 6

Methods

What students will learn:

* Writing their own methods
* Calling methods
* Using built-in methods

Sometimes called “functions” or “procedures”, methods are blocks of code that have been given a name for the sake of convenience. This is done usually for clarity or to reduce the overall amount of code being written. Whenever we wish to execute the code inside that method, we “call” the method.

All methods you write in this class have a “template” that looks like:

def <name>(<parameters>):

<body>

In the template above, <name> is the name of the method (which can be any name except reserved keywords), the parameter list in parentheses (which are the method’s inputs), and the body of the method (the code that will be executed once the method is called).

After a method has been defined, we can call it from anywhere else in the code. Once the computer reaches a line of code containing a method, it will halt the execution of the code that is currently running and execute the code inside the method before resuming. Methods reduce the amount of code we have to write: instead of writing the same block of code over and over across our entire program, we can simply place that block apart, give it a name, and then call it whenever we need it. To call a method, simply write its name and pass it any arguments that it requires:

<name>(<arguments>)

You may have noticed that we’ve used two different words to describe a method’s input: parameters and arguments. While some people may use them interchangeably, they are technically different things. Parameters are what inputs the method can receive, while arguments are the actual inputs received. Let’s use a vending machine as an example: Some vending machines accept bills. If the vending machine is our method, then the slot for inserting bills would be its parameter, while the actual bills we insert are the arguments.

def vending\_machine(money): # “money” is the method’s only parameter  
 print(“You’ve entered a “, money, “ bill.”)

# we’ve called the method using “One Dollar” as its argument  
 vending\_machine(“One Dollar”)

Methods are not limited to only having one parameter: You can add as many as you want (even no parameters at all), as long as you separate them using commas.

Methods can also return a value. This is done through the “return” keyword. Whenever a method reaches a return statement followed by a value, that value is returned as an output.

def get\_fullname(first\_name, last\_name):  
 return first\_name + “ “ + last\_name  
  
 get\_fullname(“Alice”, “Smith”)

Running the code above will not produce any visible output. While we’ve called the method, supplied it with arguments, and the method has a return statement, we aren’t doing anything with the value being returned. If we wish to display the value that has been returned, we must send it to the console:

def get\_fullname(first\_name, last\_name):  
 return first\_name + “ “ + last\_name  
  
 full\_name = get\_fullname(“Alice”, “Smith”)  
 print(full\_name)

It is important to make the distinction that **printing something to the console does not constitute returning a value**. When a method returns a value, that value is being returned to a different part of the code, not to the user.

# prints the full name but returns None

def print\_fullname(first\_name, last\_name):  
 print(first\_name + “ “ + last\_name)

# returns a string but prints nothing

def get\_fullname(first\_name, last\_name):  
 return first\_name + “ “ + last\_name

Methods can also feature the return keyword without returning a value. This is because the return keyword’s primary duty is signaling that the method is done executing. As soon as the code being executed reaches a return, any code below that return will not be executed:

def only\_count\_to\_five(number):  
 for x in range(number):  
 if x >= 5:  
 return  
 print(x)

only\_count\_to\_five(20)

If the method above is supplied with a positive integer that is less than 5, it will print all integers from 0 up until that number. However, if it is supplied with a number greater than 5, it will always stop after having printed 4. All methods which terminate naturally (they execute all the code inside of it without hitting a return statement) or which hit a return keyword without an accompanying value always return **None**.

Methods in Python can also accept both optional and required arguments. Required parameters must receive arguments when the method is called, or the program will crash. Optional arguments have default values associated with them; if no arguments are passed, the default values will be used.

def greet\_user(age, name=”Person”):  
 print(“Hello “ + name + “. You are “ + str(age) + “ years old.”)

greet\_user(30)  
greet\_user(40, “Alice”)  
# this would crash, as the method requires at least one argument  
# greet\_user()

The code above illustrates this. The first time the method is being called, only the required argument “age” is being passed, so the method is using the default value for “name” (“Person”). The second time the method is called, two arguments are being passed: the first one being used as a value for “age”, and the second one for “name”.

Like before, your methods can have any number of required and optional parameters as necessary. Please note that, in the parameter list, required parameters must be listed before optional parameters.

Finally, you may have noticed that Python already comes with some methods for you to use out of the box (such as print(), input(), and len()). However, it is also possible for you to use methods from other files (either written by you or by other people) by importing them into your current file.

from file\_name import method\_name

Notice that, for the purpose of this lab, this will only work if the file you are trying to import is in the same folder as the file you are importing to.

# This is inside FileOne

def greet\_user(name):  
print(“Hello “ + name)

-----

# This is inside FileTwo

from FileOne import greet\_user

username = input(“Enter your name: “)

greet\_user(username)

**Lab6A:** Design and implement a program that implements the following 3 methods:

* Method isValid(...) returns True if the sum of the width and height is greater than 30. Otherwise, returns False

isValid(width, height)

* Method area(...) returns the area of the rectangle.

area(width, height)

* Method perimeter(...) returns the perimeter of the rectangle.

perimeter(width, height)

Prompt the user to enter the width and height of a rectangle (as floats) and use the methods above to print out a message followed by the area and perimeter if the rectangle is valid. Otherwise, it prints out only the message “This is an invalid rectangle.”

Note: that method isValid() is used to validate the input before attempting to compute the area and perimeter.

Allow the user to re-run the program with different inputs using a loop structure. The user input is indicated in bold.

Sample output:

Enter width: **4.0**

Enter height: **5.0**

This is an invalid rectangle.

Do you want to enter another width and height (Y/N)?: **Y**

Enter width: **20.0**

Enter height: **15.0**

This is a valid rectangle.

The area is: 300.0

The perimeter is: 70.0

Do you want to enter another width and height (Y/N)?: **N**

Program Ends

**Lab6B:** For this lab, you will need to write two different files; one called MyMath.py, and one called Lab6B.py.

Inside of MyMath.py, write the following three methods:

* my\_max(num\_one, num\_two): returns the largest between the two inputs
* my\_min(num\_one, num\_two): returns the smallest between the two inputs
* my\_avg(num\_one, num\_two): returns the mean average of the two inputs

Inside Lab6B.py, import MyMath.py. After that, prompt the user for two numbers, and then pass both numbers as arguments to each of MyMath’s methods, making sure to print out the result as in the sample output below (user input in bold).

Note: You can assume that the user will only ever enter numbers when prompted.

Sample output #1:

Enter number 1: **4**

Enter number 2: **9**

Min is 4

Max is 9

Average is 6.5

Sample output #2:

Enter number 1: **45**

Enter number 2: **11**

Min is 11

Max is 45

Average is 28.0

**Submit all 3 files (Lab6A.py, Lab6B.py, and MyMath.py) to Gradescope.**