# CMSC 201 Fall 2018
## Lab 07 – Functions

**Assignment:** Lab 07 – Functions
**Due Date:** **During discussion**, October 15th through October 18th
**Value:** 10 points (8 points during lab, 2 points for Pre Lab quiz)

This week's lab will put into practice the concepts you learned about functions: creating functions, passing parameters, and scope.

(Having concepts explained in a new and different way can often lead to a better understanding, so make sure to pay attention as your TA explains.)

## Part 1A: Review – Scope

Everything in Python has a ***scope*** – the places in the program in which it is accessible. For example, you can create a constant outside of **`main()`**.

```
MAX_VAL = 8

def otherFxn():
    # code in otherFxn() has access to MAX_VAL
    print("The maximum allowed value is", MAX_VAL)

def main():
    # code in main() has access to MAX_VAL
    guess = int(input("Guess the max value: "))
    while guess != MAX_VAL:
        print("Whoops, you guessed wrong.")
        guess = int(input("Guess again: "))
main()
```

That constant is now a ***global*** constant, which means it can be accessed by any line of code in the file. So **`main()`** can access it, as well as any other functions that you might write. Remember, for this course you are only allowed to have constants be global – regular variables (that aren't constants) should only be declared inside functions.

***Local*** variables are only accessible to code within their same scope. If a variable is declared in **`main()`**, another function called **`printInfo()`** will not be able to access it. In the same way, a variable in **`printInfo()`** will not be accessible to the code in **`main()`**.

```
def printInfo():
    # this variable can't be accessed by main()
    varForPrintInfo = 5

def main():
    # this variable can't be accessed by printInfo()
    varForMain = 17

main()
```

## Part 1B: Review – Functions

A function is a way of compartmentalizing our code: a well-written function does *one* thing, and does it very well. A function allows us to write a piece of code once, and to then use, or "call," the function whenever we want to use that code.

A function has a few key parts:

1. Function name
    - This is how we call the function. It tells Python that we want it to use that function and execute its…
2. Function body
    - This is the code that makes up the function. This is what the function does when called.
3. Formal parameters (optional)
    - A function uses parameters to take in information from the code that called it. This is one of the ways that data is passed from one piece of code to another. A function can have no parameters, one parameter, or it could have a hundred!

Let's take a look at some example functions and how they work:

```
def printName(name):
    print("Hello, my name is", name)
```

This function is called **`printName()`**, and it takes in one formal parameter (**`name`**). In order to use the code in this function, we must call the function and pass it an ***argument***. The argument could be a variable, or it could be a literal string (one with quotation marks around it).

Here's the **printName()** function again, but this time we also have a **main()** that calls the function multiple times.

```
def printName(name):
    print("Hello, my name is", name)

def main():
    userName = input("What's your name? ")
    prezUMBC = "Hrabowski"
    printName(userName)
    printName(prezUMBC)
    printName("John Jacob Jingleheimer Schmidt")

main()
```

Note that we have called the function multiple ways: both with variables and with a string literal. Note also that the variable names we passed as actual parameters (**prezUMBC** and **userName**) do not need to match the name of the formal parameter.

Here is the output for the code above:

```
Hello, my name is YOUR_NAME_HERE
Hello, my name is Hrabowski
Hello, my name is John Jacob Jingleheimer Schmidt
```

## Part 2: Exercise

In this lab, you'll be downloading a file and completing it by writing three function definitions, and then writing three function calls in **`main()`**.

The program you'll be coding will ask the user to create a list of names, and will then print out three things: the entirety of the list, the shortest name, and the longest name.

### Tasks

Starting:

- ☐ Copy the **`given_names.py`** file from Prof. Neary's **`pub`** directory
  - ☐ It should be renamed to be **`names.py`**
- ☐ Complete the file header comment at the top

Functions:

- ☐ Write the code for **`printList()`**
- ☐ Write the code for **`printMinStr()`**
- ☐ Write the code for **`printMaxStr()`**

**`main():`**

- ☐ Call the function **`printList()`**
- ☐ Call the function **`printMinStr()`**
- ☐ Call the function **`printMaxStr()`**

General:

- ☐ Run and test your code as needed
- ☐ Show your work to your TA

## Part 3A: Downloading the File

First, create the **`lab07`** folder using the **`mkdir`** command -- the folder needs to be inside your **`Labs`** folder as well.

Next, copy a file into your **`lab07`** folder using the **`cp`** command.

```
cp /afs/umbc.edu/users/m/n/mneary1/pub/cs201/given_names.py names.py
```

This will copy the files **`given_names.py`** from Prof. Neary's public folder into your current folder, and will change the file's name to **`names.py`** instead.

The first thing you should do in your file is complete the file header comment, filling in your name, section number, email, and the date.

## Part 3B: Creating Functions

At this point, if you try to run the file, you will get an error. That is because the file is only partially completed for you.

You will need to update the file to complete the three function definitions and three function calls. If you open the file, you should see comments boxed in by **#--------#** characters – these are where you need to write new code. Read the function header comments to see the details about the three functions.

You should have written code similar to all of these functions previously, either in homeworks, labs, or in-class exercises. The code for **`printList()`** should be familiar to you, as you have printed quite a few lists during in-class exercises. The big difference is that in this function, the index of each element should also be printed out.

The functions **`printMinStr()`** and **`printMaxStr()`** are very similar; once you complete one function, the other should be very easy to code up. Think carefully about what the function needs to accomplish: it needs to iterate over the entire list, keeping track of which string is the shortest or longest it's found so far. (*HINT: When choosing a starting value to compare every other string to, use the first string in the list.)*

(If two strings are the same length and are the shortest/longest length, picking either one for the final answer is fine. In the sample output below, we pick the first one we see, but your code may work differently.)

You will also need to write calls to each of these functions. The places where these calls need to happen in **`main()`** are indicated for you. You shouldn't need to write any other code.

(See the next page for sample output.)

Here is some sample output of the completed program, with user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
linux2[3]$ python3 names.py
Enter a name (or -1 to quit): Dr. Gibson
Enter a name (or -1 to quit): -1
At index 0 the name is "Dr. Gibson"
The shortest string in the list is Dr. Gibson
The longest string in the list is Dr. Gibson

linux2[4]$ python3 names.py
Enter a name (or -1 to quit): Matt
Enter a name (or -1 to quit): Zoee
Enter a name (or -1 to quit): Ben
Enter a name (or -1 to quit): Anna
Enter a name (or -1 to quit): Emily
Enter a name (or -1 to quit): Kristin
Enter a name (or -1 to quit): Sahil
Enter a name (or -1 to quit): Nick
Enter a name (or -1 to quit): Agatha
Enter a name (or -1 to quit): Hannah
Enter a name (or -1 to quit): -1
At index 0 the name is "Matt"
At index 1 the name is "Zoee"
At index 2 the name is "Ben"
At index 3 the name is "Anna"
At index 4 the name is "Emily"
At index 5 the name is "Kristin"
At index 6 the name is "Sahil"
At index 7 the name is "Nick"
At index 8 the name is "Agatha"
At index 9 the name is "Hannah"
The shortest string in the list is Ben
The longest string in the list is Kristin
```

## Part 4: Completing Your Lab

Since this is an in-person lab, you do not need to use the **`submit`** command to complete your lab. Instead, raise your hand to let your TA know that you are finished.

They will come over and check your work – they may ask you to run your program for them, and they may also want to see your code. Once they've checked your work, they'll give you a score for the lab, and you are free to leave.

### Tasks

Starting:

- ☐ Copy the **`given_names.py`** file from Prof. Neary's **`pub`** directory
  - ☐ It should be renamed to be **`names.py`**
- ☐ Complete the file header comment at the top

Functions:

- ☐ Write the code for **`printList()`**
- ☐ Write the code for **`printMinStr()`**
- ☐ Write the code for **`printMaxStr()`**

**`main():`**

- ☐ Call the function **`printList()`**
- ☐ Call the function **`printMinStr()`**
- ☐ Call the function **`printMaxStr()`**

General:

- ☐ Run and test your code as needed
- ☐ Show your work to your TA

**IMPORTANT:** If you leave the lab without the TA checking your work, you will receive a **zero** for this week's lab. Make sure you have been given a grade before you leave!