CSC 243: Python for Programmers

# Homework 2

## Reading

Read Chapters 4 and 5 in **Introduction to Computing using Python: An Application Development Focus, Second Edition** by Ljubomir Perković.

## Logistics

If you want to do the assignment on your own computer, you will need to download and install Python. There is a link for downloading Python under Content -> Course Information on D2L.

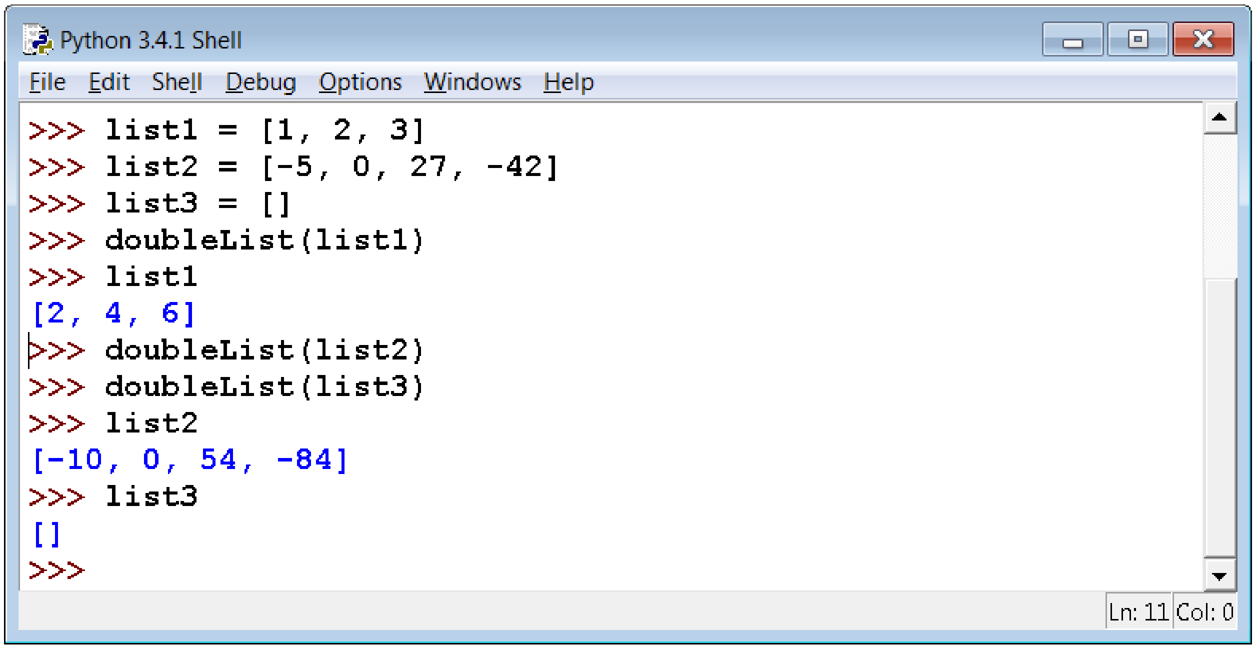
In this class programming assignments may be completed in consultation with up to two other classmates. You must identify the classmates with whom you collaborate in the comment box when you submit on D2L. You must also list their names at the top of the assignment. The total number of collaborators on any assignment **may not exceed two other people**. Please see the Collaboration Guidelines document found under Content -> Course Information on D2L for more details on what is or is not allowed when working on homework.

Remember that **everyone** submitting the assignment must be able to explain **all** of the code submitted, regardless of the type of collaboration that occurred. Anyone submitting code they cannot explain is violating the Academic Integrity policy and will earn a 0 on the assignment.

## Assignment

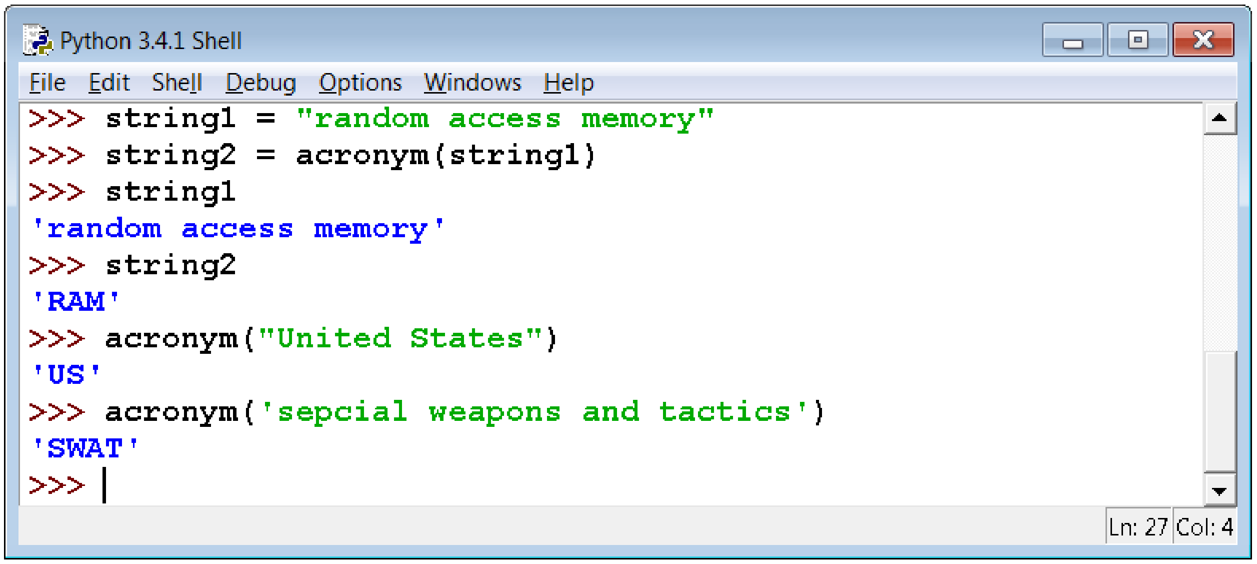
To begin the assignment, download the **HW2.py** template found on the D2L site. You will need to add appropriate doc strings to the functions there as well as complete the implementation as described below. **Submissions without doc strings will not earn full credit**.

1. Write a function **doubleList**() that takes a list of numbers as a parameter. It does not return anything, but instead **modifies** the input list so that each slot is twice its original value. Several examples appear below:



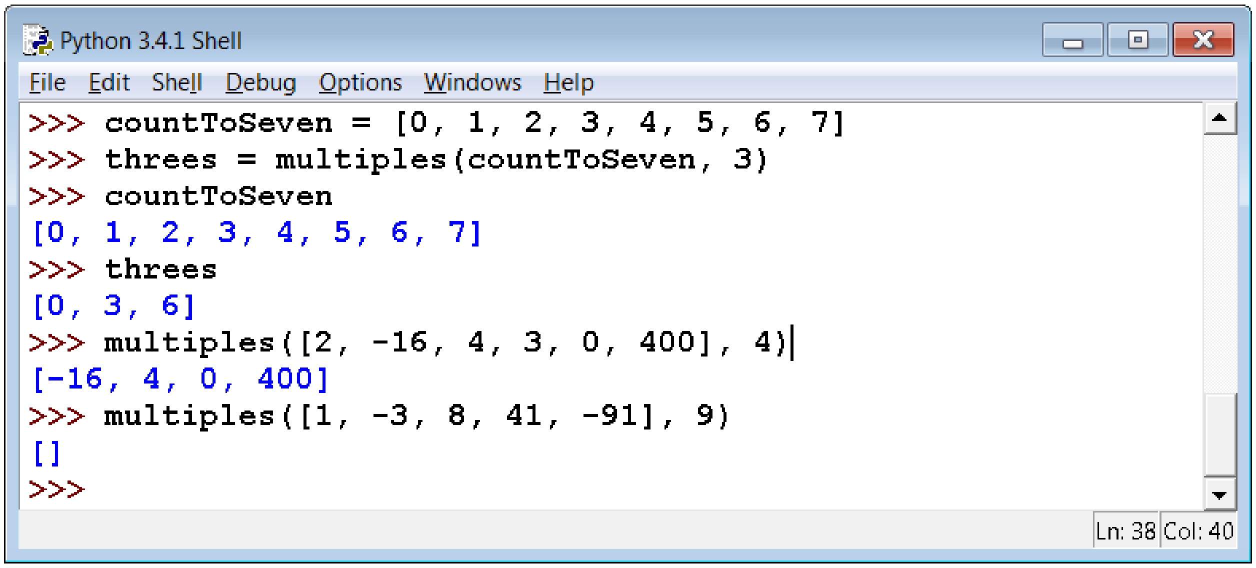
```
>>> list1 = [1, 2, 3]
>>> list2 = [-5, 0, 27, -42]
>>> list3 = []
>>> doubleList(list1)
>>> list1
[2, 4, 6]
>>> doubleList(list2)
>>> doubleList(list3)
>>> list2
[-10, 0, 54, -84]
>>> list3
[]
>>>
```

2. Write a function called **acronym()** that takes a single string argument that consists of sequence of words separated by spaces. It **returns** a string consisting of the first letter of each word capitalized and in the same order as the input string. The following shows several sample runs of the function:



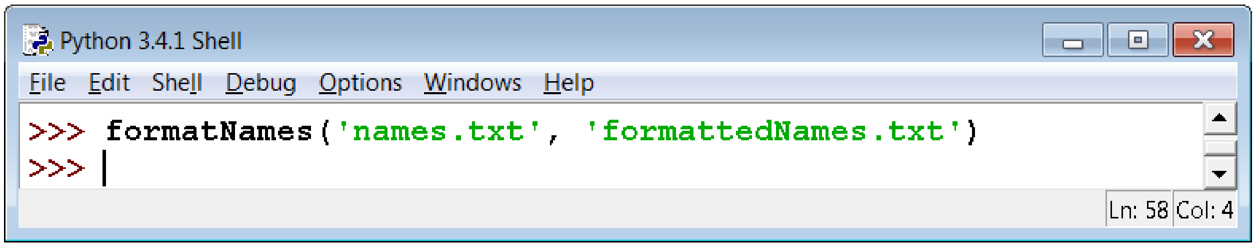
```
>>> string1 = "random access memory"
>>> string2 = acronym(string1)
>>> string1
'random access memory'
>>> string2
'RAM'
>>> acronym("United States")
'US'
>>> acronym('sepcial weapons and tactics')
'SWAT'
>>> 
```

3. Write a function **multiples()** that takes two arguments. The first argument is a list of integers. The second argument is an integer n. The function **returns** a new list of integers consisting of the integers from the original list that are a multiple of n (that are evenly divisible by n). The numbers in the result must appear in the same order as they appear in the original list. The following shows several sample runs of the function:



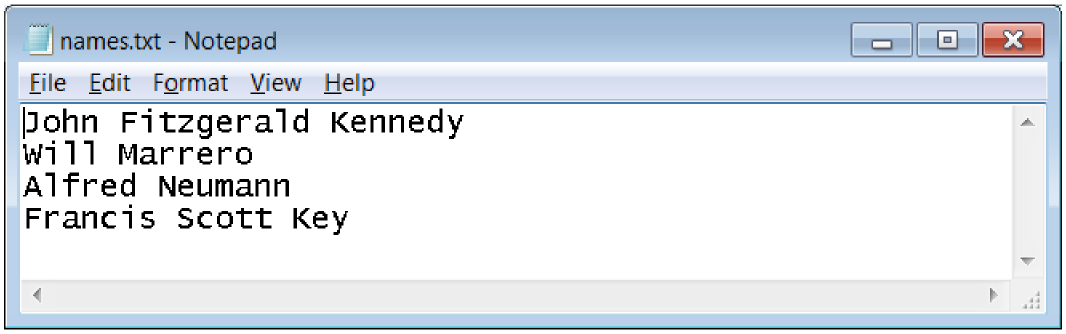
```
>>> countToSeven = [0, 1, 2, 3, 4, 5, 6, 7]
>>> threes = multiples(countToSeven, 3)
>>> countToSeven
[0, 1, 2, 3, 4, 5, 6, 7]
>>> threes
[0, 3, 6]
>>> multiples([2, -16, 4, 3, 0, 400], 4)
[-16, 4, 0, 400]
>>> multiples([1, -3, 8, 41, -91], 9)
[]
>>>
```

4. Write a function **formatNames()** that takes two string arguments. The first is the name of a file that contains a list of names, one per line. Each name in the file has one of two possible formats. Either it is the first name followed by a space followed by the last name or it is the first name followed by a space followed by the middle name followed by the last name. The function should **create a new file** whose name is second argument. The new file should contain all the names in from the original file one per line and in the same order, but they should instead be formatted as Last, First if the original name did not have a middle name, or as Last, First M. if the name did include a middle name. (Note that M. is the middle initial.) Below are some screenshots illustrating how this function works on a sample file.



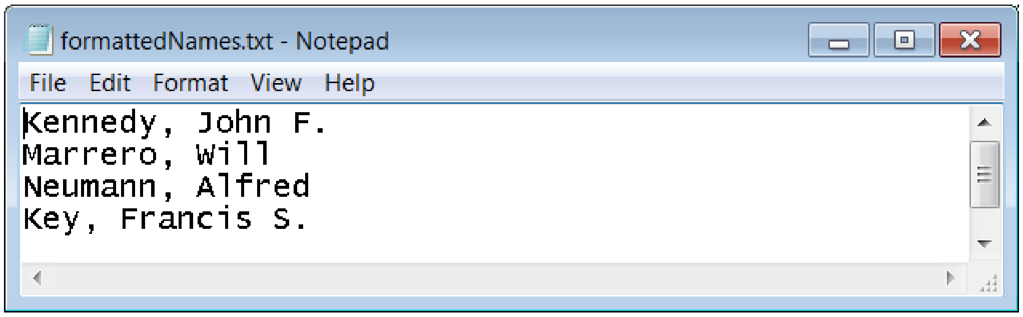
```
>>> formatNames('names.txt', 'formattedNames.txt')
>>> 
```

Assuming the following file called names.txt is in the same folder as HW2.py:



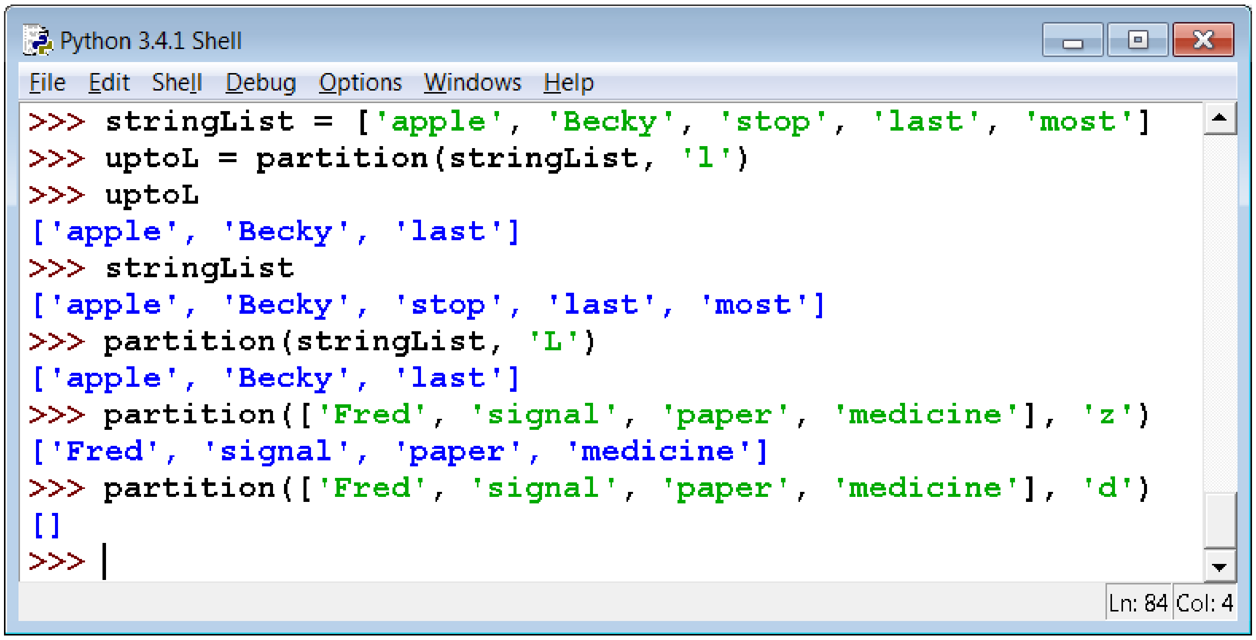
```
John Fitzgerald Kennedy
Will Marrero
Alfred Neumann
Francis Scott Key
```

a new file called formatedNames.txt will be created in the same folder as HW2.py:



```
Kennedy, John F.
Marrero, Will
Neumann, Alfred
Key, Francis S.
```

5. Write a function **partition**() that takes a list of strings and a character as parameters. It **returns** the list strings from the input list that begin with a letter between the letter ‘a’ and the letter provided as the character argument (inclusive). The strings in the result must appear in the same order as they appear in the input list. The capitalization of the character and the first letter of the string should not make a difference in whether the string is printed. Empty strings should be skipped. You may assume that the character given as the second parameter is a letter (either uppercase or lowercase). The following shows several sample runs of the function:



```
>>> stringList = ['apple', 'Becky', 'stop', 'last', 'most']
>>> uptoL = partition(stringList, 'l')
>>> uptoL
['apple', 'Becky', 'last']
>>> stringList
['apple', 'Becky', 'stop', 'last', 'most']
>>> partition(stringList, 'L')
['apple', 'Becky', 'last']
>>> partition(['Fred', 'signal', 'paper', 'medicine'], 'z')
['Fred', 'signal', 'paper', 'medicine']
>>> partition(['Fred', 'signal', 'paper', 'medicine'], 'd')
[]
>>> 
```

## Submitting the assignment

You must submit the assignment using the HW2 folder of the dropbox on D2L. Submit only two files. The first is a single Python file (HW2.py) with your solutions found in it. The second is a document (preferably Word or PDF) with the screenshots of your tests for the 5 functions.

Submissions after the deadline will be automatically rejected by the system and will receive a grade of 0.

## Grading

Each of the five problems is worth 20 points for a total of 100 points.