CSC 243: Python for Programmers

Homework 1

Reading

Read Chapters 2, and 3 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.

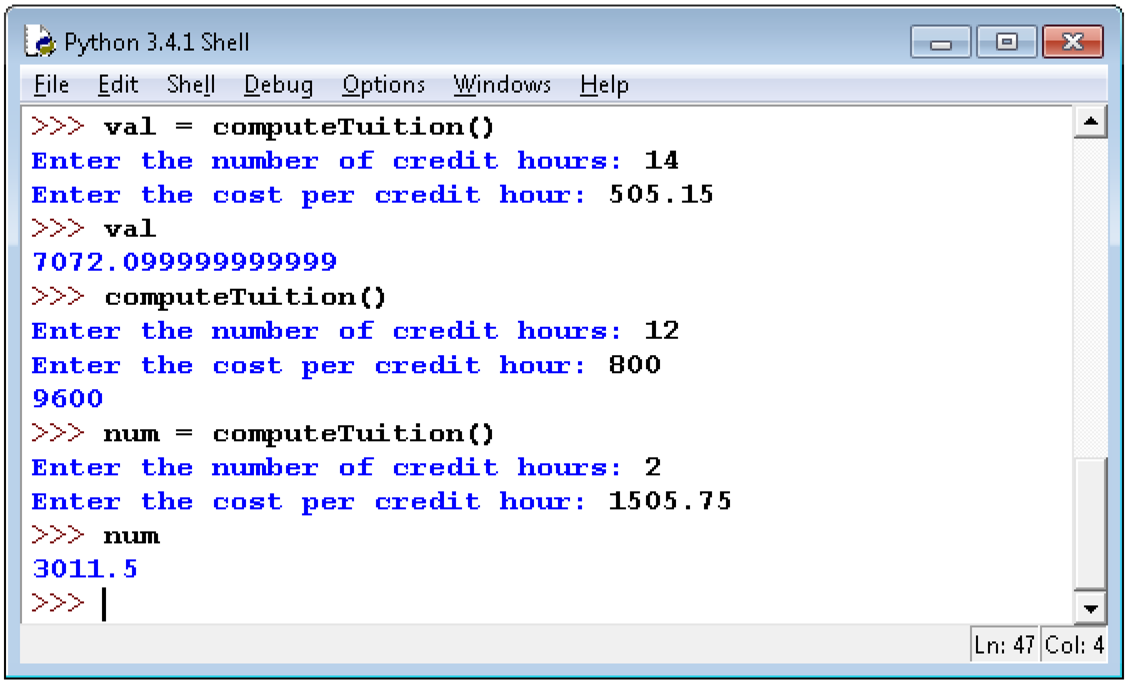
Remember that **everyone** submitting the assignment must be able to explain **all** of the code submitted, regardless of the type of collaboration that occurred. Anyone submitting code they cannot explain is violating the Academic Integrity policy and will earn a 0 on the assignment.

Also, in addition to submitting the file with your solution, you are required to submit screenshots of the results of testing your code. You must test your code so exactly the same as in my examples for each problem. To take a screenshot on a Windows machine, click on the python window and hit Alt-PrtScn (hold down the Alt key and press the PrtScn key). You can then paste into Word with Ctrl-v. Please submit one Word document with all the screenshots pasted into it. **Important:** While these tests are necessary, they are not sufficient to ensure full credit. Your code must work correctly on all valid inputs.

Assignment

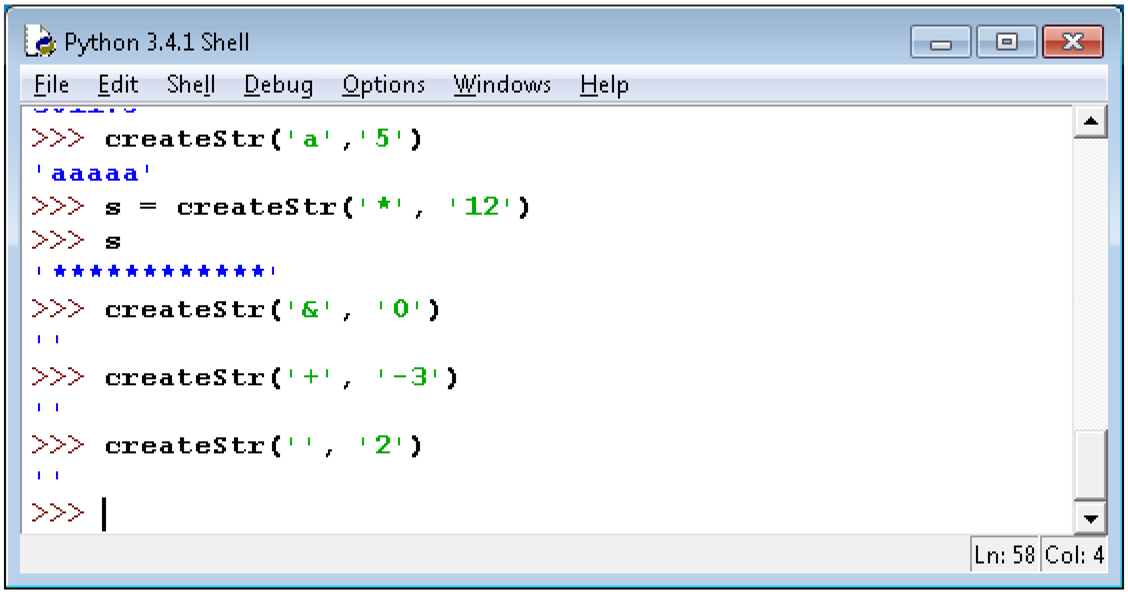
To begin the assignment, download the **HW1.py** template found on the D2L site. You will need to add appropriate doc strings to the functions there as well as complete the implementation as described below. **Submissions without doc strings will not earn full credit**.

1. Write a function **computeTuition**() that prompts the user to type in an integer representing the number of credit hours taken by a student and a floating point number representing the cost per credit hour of tuition and **returns** the total tuition cost for the student. The following shows how the function could be used and the screenshot you must submit of your testing.



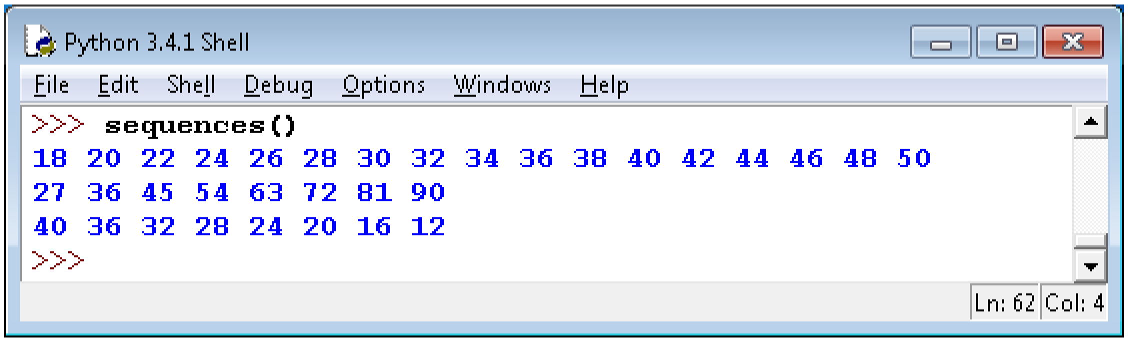
```
>>> val = computeTuition()
Enter the number of credit hours: 14
Enter the cost per credit hour: 505.15
>>> val
7072.099999999999
>>> computeTuition()
Enter the number of credit hours: 12
Enter the cost per credit hour: 800
9600
>>> num = computeTuition()
Enter the number of credit hours: 2
Enter the cost per credit hour: 1505.75
>>> num
3011.5
>>> 
```

2. Implement a function **createStr**() that takes a character (a string of length 1) and a **string** containing a number as **parameters** and **returns** a string consisting of the character duplicated as many times as indicated by the number. You may assume that the string containing a number represents a valid integer, and your function is not required to do anything meaningful if the second parameter does not represent an integer. If the second parameter represents an integer less than or equal to 0, the empty string should be returned. Note that solving this problem does not require using branching statements (e.g. if statements) or looping statements (e.g. for or while statements). Please ask if you don't see how to solve this without those statements. The following shows how the function could be used on several sample parameters and the screenshot you must submit of your testing.



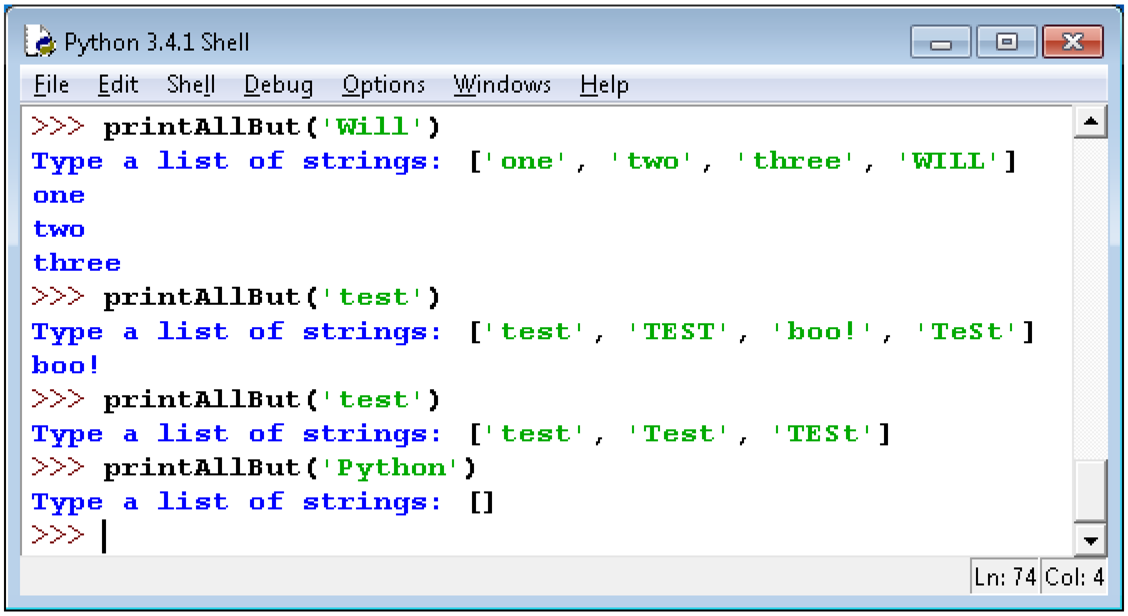
```
>>> createStr('a','5')
'aaaaa'
>>> s = createStr('*', '12')
>>> s
'************'
>>> createStr('&', '0')
''
>>> createStr('+', '-3')
''
>>> createStr('', '2')
''
>>> 
```

3. Write the function **sequences**() found in the template file so that the following three sequences are **printed** by the function using appropriate for loops that use the range() function. Hard-coded solutions will earn a 0. The information below shows how you would call the function **sequences**() and what it would display as a result. Make sure to submit this screenshot of your testing.



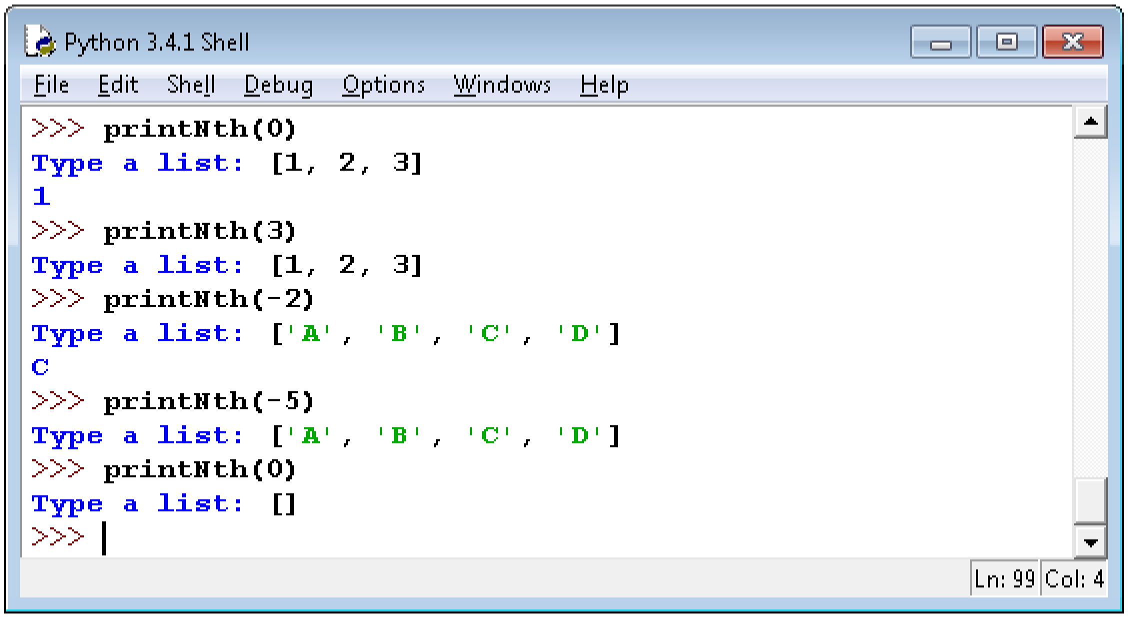
```
>>> sequences()
18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
27 36 45 54 63 72 81 90
40 36 32 28 24 20 16 12
>>> 
```

4. Write a function **printAllBut**() that takes a string as a parameter. The function then prompts the user to input a list of strings, and then **prints** all of the strings in the list that are not equal to the string provided as a parameter. The output strings are printed one per line. If the list of strings is empty, nothing will be printed. If the list is non-empty but only contains the string provided as a parameter, nothing will be printed. The case of the letters in the strings should **not** make a difference when matching. The following shows how the function could be used on several sample parameters as well as the screenshot you must submit of your testing.



```
>>> printAllBut('Will')
Type a list of strings: ['one', 'two', 'three', 'WILL']
one
two
three
>>> printAllBut('test')
Type a list of strings: ['test', 'TEST', 'boo!', 'TeSt']
boo!
>>> printAllBut('test')
Type a list of strings: ['test', 'Test', 'TESt']
>>> printAllBut('Python')
Type a list of strings: []
>>> 
```

5. Implement a function **printNth**() which takes an integer n as a parameter, prompts the user to enter a list, and **prints** the nth element in the list. If the list is empty or n is not a valid index into the list, the function will not print anything. The following shows several sample executions of the function and the screenshot you must submit of your testing.



```
>>> printNth(0)
Type a list: [1, 2, 3]
1
>>> printNth(3)
Type a list: [1, 2, 3]
>>> printNth(-2)
Type a list: ['A', 'B', 'C', 'D']
C
>>> printNth(-5)
Type a list: ['A', 'B', 'C', 'D']
>>> printNth(0)
Type a list: []
>>> 
```

Submitting the assignment

You must submit the assignment using the HW1 folder of the dropbox on D2L. Submit only two files. The first is a single Python file (HW1.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.