Lab Questions: Lab Session 6

Deadline: 27.09.2017 11:59pm SGT

Complete all assignments below. For those questions that are marked with an asterisk *, i.e. **Questions 7 and 11**, create the script files as requested. Once you are done with it, submit the file(s) via `iNTU`. Remember to put plenty of comments for all assignments, as this helps us to better understand your program (which might give you higher marks).

Important!!! Make sure your scripts work properly, as we give 0 marks otherwise. Please name the scripts according to the requirements, and upload each file separately and not in a Zip file or similar. The submission system closes at the deadline. Hence after that, you will get no marks for your solution.

1. Write a `for` loop that prints all characters of a string `my_str` vertically. Do two versions: one that will iterate through the elements of `my_str` themselves, and one that will iterate through the indexes of `my_str`.

2. Write a script `prodby2.py` that generates a random positive integer $n \in [1, 100]$ and will calculate and print the product of the odd integers from 1 to n (or from 1 to *n-1* if n is even).

3. Write a script `belong.py` that will pick a list L of 10 random integers $\in [1, 100]$. Then, using a `for` loop, iterate through the integers from 32 to 64 included and display the current integer if it belongs to the list L.

4. Create a Numpy array of 5 random integers, each in the range from -10 to 10 included. Perform each of the following tasks with two versions: using loop(s) (with `if` statements if necessary) and without loop.

   (a) subtract 3 from each element of the array

   (b) count how many elements of the array are positive

   (c) replace each elements of the array by its absolute value

   (d) find the maximum value of the array

5. Assume that a Numpy matrix `my_mat` is already defined. Write a script `matrixAverage.py` that will calculate and print the overall average of all numbers in the matrix. Use loops, not built-in functions, to calculate the average. The size of the matrix should be assumed to be unknown. What is the build in Numpy built-in function that performs exactly this task ?

6. Write a script `beautyofmath.py` that produces the following output. The script should iterate from 1 to 9 to produce the expressions on the left, perform the specified operation to get the results shown on the right, and print exactly in the format shown here.

```
>> beautyofmath
1 x 8 + 1 = 9
12 x 8 + 2 = 98
123 x 8 + 3 = 987
1234 x 8 + 4 = 9876
12345 x 8 + 5 = 98765
123456 x 8 + 6 = 987654
```

```
1234567 x 8 + 7 = 9876543
12345678 x 8 + 8 = 98765432
123456789 x 8 + 9 = 987654321
```

7. * Write a script <YourMatricNo>_Lab6_AboveandleftSumMatrix.py that asks the user for a number of rows r and a number of columns c. Then, it generates a $r \times c$ matrix with the following values:

   - the value of each element in the first row is the number of the column
   - the value of each element in the first column is the number of the row
   - the rest of the elements each has a value equal to the sum of the element above it and the element to the left

   Here is an example of such matrix if the user inputs $r = 4$ and $c = 5$:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 7 & 11 & 16 \\ 3 & 7 & 14 & 25 & 41 \\ 4 & 11 & 25 & 50 & 91 \end{pmatrix}$$

8. An inefficient sorting. Write a script `sorting.py` that first asks the user to enter a positive integer n (check if the integer entered is positive and if not continue asking the user). Then, it generates a vector of n random integers between 1 and 100 and finally sorts that vector (and prints it after being sorted). More precisely, you can use the following simple and inefficient algorithm to sort the vector (do not use a built-in Python function to sort the vector): check all pairs of consecutive elements of the vector and swap them if they are not sorted; repeat until all pairs of the vector are sorted.

9. Write a script `findmine.py` that will prompt the user for minimum and maximum integers, and then another integer which is the user's choice in the range from the minimum to the maximum. The script will then generate random integers in the range from the minimum to the maximum, until a match for the user's choice is generated. The script will print how many random integers had to be generated until a match for the user's choice was found. For example, running this script might result in this output:

```
>>> findmine
Please enter your minimum value:  -2

Please enter your maximum value:  3

Now enter your choice in this range:  0

It took 3 tries to generate your number
```

10. The inverse of the mathematical constant e can be approximated as follows:

$$\frac{1}{e} \approx \left(1 - \frac{1}{n}\right)^n$$

    Write a script `approxe.py` that will loop through values of n until the difference between the approximation and the actual value is less than 0.0001 (the actual value of e can be obtained in the `math` module). The script should then print out the built-in value of e^{-1} and the approximation to 4 decimal places, and also print the value of n required for such accuracy.

11. * Write a script <YourMatricNo>_Lab6_thresholdMatrix.py that asks the user to input a threshold value thres with thres $\in (0, 1)$. Then, using while and for loops, the script should generate a (5×5) matrix filled with random values in $(0, 1)$ and such that the average value for each row is greater or equal than thres. When a solution matrix has been found, print it as well as its average for each row, and the number of tries that were required to find it.

12. A blizzard is a massive snowstorm. Definitions vary, but for our purposes we will assume that a blizzard is characterized by both winds of 30 mph or higher and blowing snow that leads to visibility of 0.5 miles or less, sustained for at least four hours. Data from a storm one day has been stored in a file stormtrack.dat. There are 24 lines in the file, one for each hour of the day. Each line in the file has the wind speed and visibility at a location, e.g.
    29.2 1.1
    15.1 1.5
    ...
    29.4 0.78
    Create a sample data file stormtrack.txt or download a example version on iNTU, and place it in your working folder. Create a script stormtrack.py that reads this data from the file (you can do so by typing the command my_array = np.loadtxt('stormtrack.txt') that will put the data in the Numpy array my_array) and determines whether blizzard conditions were met during this day or not.