Lab Questions: Lab Session 7

Deadline: 11.10.2017 11:59pm SGT

Complete all assignments below. For those questions that are marked with an asterisk *, i.e. **Questions 6 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 function `calcrectarea` that will calculate and return the area of a rectangle. Test the function by passing the length and width to the function as input arguments and printing the corresponding output.

2. Write a function `vecout` that will receive one integer argument and will return a Numpy array that increments from the value of the input argument to its value plus 5, with a step of 1. For example,
```
>>> vecout(10)
array([10, 11, 12, 13, 14, 15])
```

3. If a certain amount of money (called the principal P) is invested in a bank account, earning an interest rate i compounded annually, the total amount of money T_n that will be in the account after n years is given by:
$$T_n = P * (1 + i)^n$$
Write a function that will receive input arguments for P, i, and n, and will return the total amount of money T_n.

   Also, give an example of calling the function, printing in a sentence the inputs and the output of the function. For that, try two methods to use the function: write the function directly in your script, or write the function in a separate file that you will import beforehand.

4. Write a script `time_diff.py` that will compute the number of seconds contained in duration entered by the user. The script will do the following:
   - prompt the user to input a number of years, days and hours (you can assume that only integers will be entered by the user)
   - call a function `convert` to compute the number of seconds contained in that duration. The function takes three inputs (the number of years, days and hours) and returns the number of seconds. You can assume for simplicity that all years have 365 days.
   - output the number of seconds in the format as shown below

   Submit your script file to `iNTU`.
   Example:
```
Enter the number of years:  24
Enter the number of days:  145
Enter the number of hours:  6
There are 769413600 seconds in 24 years, 145 days and 6 hours
```

5. Write a function `my_all` that takes a list of integers as input and performs the same action as the built-in function `all`. Same exercise with `my_any` and the built-in function`any`.

6. * Write a script `<YourMatricNo>_Lab7_Density.py` that will compute the population density of a given country. The script will do the following:

   - prompt the user to input the the name of the country, the total area and the population
   - call a function `countryDensity` to compute the density. The function takes two inputs (the total area and the population) and returns the density
   - output the name of the country and the density in the format as shown below

   Submit your script file and the function file to `iNTU`.
   Note, the density should be a whole number.

   Example:
   ```
   Computation of density
   Enter the name of the country:Singapore
   Enter the area (in km2):716.1
   Enter the population:5399200
   The population density of Singapore is 7539 people per km2
   ```

7. The cost of manufacturing n units (where n is an integer) of a particular product at a factory is given by the equation:
   $$C(n) = 5n^2 - 44n + 11$$
   Write a script `mfgcost.py` that will

   - prompt the user for the number of units n
   - call a function `costn` that will calculate and return the cost of manufacturing n units
   - print the result (the format must be exactly as shown below)

   Next, write the function `costn`, which simply receives the value of n as an input argument, and calculates and returns the cost of manufacturing n units.

   Here is an example of executing the script:
   ```
   >>> mfgcost
   Enter the number of units:  100
   The cost for 100 units will be $45611.00
   ```

8. Write a function that is called `pickone`, which will receive one input argument `x`, which is a Numpy array, and will return one random element from that array. For example,
   ```
   >>> pickone([0,1,2,3,4,5,6,7,8,9])
     7
   ```

9. An approximation for a factorial can be found using Stirling's formula:
   $$n! = \sqrt{2\pi n}(n/e)^n$$
   Write a function `approxNFact` to implement this, passing the value of n as an argument.

10. Let A, B and C be three functions defined as:
    $$A(x) = \begin{cases} x+2 & \text{when } x \leq 0 \\ x/\sqrt{x} & \text{when } x > 0 \end{cases}$$

$$B(x) = 2x^6 + 3x - 2$$

$$C(x) = \begin{cases} 6 & \text{when } x < -6 \\ -x & \text{when } x \geq -6 \text{ and } x < 3 \\ 0 & \text{when } x \geq 3 \end{cases}$$

Write a script **`func.py`** that contains six functions:

- three separate Python functions `funcA`, `funcB`, `funcC` that will implement functions A, B and C respectively. They all take a value `x` as input.
- a function `menu_func` that asks the user which one of the three functions he would like to compute (with an error check). The function returns a string 'A', 'B', or 'C' depending on the choice of the user.
- a function `menu_x` that asks the user on what value `x` he would like to compute the function. The function returns the value `x` chosen by the user.
- a function `my_menu` that combines all these previous functions to create a menu for the user to choose what function he would like to compute and on what value `x`. The function will then print a sentence to provide the result to the user.

11. * In a file <`YourMatricNo`>`_Lab7_occurrences.py`, build a function `occurrences` that takes as input a string `my_str` and a character `my_char`. It will then compute and output the number of occurrences of the character `my_char` in `my_str`. It will also output the list of the occurrences indexes (if no occurrence, it simply outputs an empty list).

12. What is the output of these codes, and why ?

```python
X = 'Hello'

def func():
    print(X)

func()
```

```python
X = 'Hello'

def func():
    X = 'Hi'

func()
print(X)
```

```python
X = 'Hello'

def func():
    X = 'Hi'
    print(X)

func()
print(X)
```

```python
X = 'Hello'

def func():
   global X
   X = 'Hi'

func()
print(X)
```