CS251: Python Outlab

Please refer to general instructions and submission guidelines at the end of this document before submitting.

P1. Nuclear Codes

The defence minister Mr. Gupta has acquired a confidential document having information about the nuclear codes. It is highly risky for the codes to be leaked. Help Mr. Gupta mask these codes.

- **Task 1**
  Mr. Gupta needs to send this document to Ministry of external affairs for a conference. But he doesn't want to give away the nuclear codes.

  An input file includes information along with a set of codes to activate a nuclear detonator. The codes in the file are in a particular format (A-ZA-Z0-9).
  Your job is to find these codes in the input file and mask each of them.
  For example,
  XX4 should become ***

  Write a bash script **dotcodes.sh** which uses the sed command to perform the above given task.
  The output of the bash script should be printed on stdout. Take the input file as a command line argument to your bash script.

  For example-
  data.txt

  ```
  ------------------------------------------------------------------------------------------
  This is a confidential document.
  It includes nuclear codes XF4. Following codes need to be authorized-
  VV4 HH6
  ML3
  LP9
  To self destruct, use XX0.
  ------------------------------------------------------------------------------------------
  ```

  The output format is as follows -

  ```
  This is a confidential document.
  It includes nuclear codes ***. Following codes need to be authorized-
  *** ***
  ***
  ***
  To self destruct, use ***.
  ```

- Usage : **bash dotcodes.sh <data-file>**
  - Ex: **bash dotcodes.sh data1.txt**
  - The data file must be passed as a command line argument and must not hard coded in your script

- **Task 2**

  Mr. Gupta wants to track the leakage of this document during the transfer from him to Ministry of external affairs. One way to do this is to set a bait, i.e. explicitly use fake codes and track how many times these fake codes are used.

  Write a bash script **fakecodes.sh** which has similar functionality as that of **dotcodes.sh** but instead of replacing codes with with *, it replaces it with a random three character code from the set (A-Z) U (0-9) (set of all capital letters and digits)

  You may use /dev/urandom in association with few other simple bash commands to generate random char in required set, but you are welcome to use any other way to produce random chars

  For example, for the above mentioned input file, the output format would be as follows (your output may be different due to randomness)-

  ```
  This is a confidential document.
  It includes nuclear codes PL6. Following codes need to be authorized-
  PL6 PL6
  PL6
  PL6
  To self destruct, use PL6.
  ```

  Notice that the you can replace each character of a code with a random character which would be better. That is a challenge for you (no need to submit that). You might want to use something other than sed in that case

- Usage : **bash fakecodes.sh <data-file>**
  - Ex: **bash fakecodes.sh data1.txt**
  - The data file must be passed as a command line argument and must not hard coded in your script

P2. String Repaired

Mr. Smith has been working in "ABC" company for about 15 years. He is incharge of storing information about all the employees in the company. He maintains a file **employees.txt** which includes the names of the employee and their respective joining date.

Due to a bug in the system, all the names in the file have been rotated by 3 characters. (1 time rotation means moving last character to the first position).
For example-
John 26-07-2017 becomes ohnJ 26-07-2017

Your task is to help Mr. Smith repair the data in the text file.

Create **strings.py** which reads the input from **employees.txt** and displays the repaired string on stdout.

The input file is as follows -
employees.txt

```
----------------------------
ohnJ 26-07-2017
aryM 13-09-2016
izaEl 13-09-2016
hewMatt 01-04-2012
----------------------------
```

The output should be displayed on stdout in the following format -

```
John
Mary
Eliza
Matthew
```

(Hint: 1. Read the file line by line 2. Access just the name in each line (split??) 3.Rotate the string (Slice in Python!!) 4. Print)

P3. Genie in the Lamp

Aladdin is trying to get to the genie lamp. But for doing so, he needs to follow a path filled with problems which he needs to solve to reach the end. Your job is to help him get the lamp. All the best!!

- **Task 1**
  Aladdin and Abu (Aladdin's pet monkey) enter the first cave. But oops, the caveman is guarding the door and has captured Abu in a cage. The key to the cage is hidden somewhere. The caveman gives Aladdin a set of clues in the form of location of the cave which is triangular in shape and the location of the key. Help Aladdin figure out whether the key is inside the cave or outside.

  Create **triangle.py** which takes the integer coordinates (x1,y1), (x2,y2) and (x3,y3) of a triangle as input. Your program needs to check whether a given point (x,y) lies inside the triangle or outside.
  Create a function **insideOut()** which checks whether the point is INSIDE the triangle or OUTSIDE the triangle.

(Hint: Sum of the areas of the triangles formed by inside point will be equal to area of whole triangle. Or, you can test if the query point lies on the same side of A-B as C etc.)
Use stdin for getting the 3 coordinates. The output should be displayed on stdout.

Example 1:

```
$ python triangle.py
Enter the first coordinate : 0 0
Enter the second coordinate : 20 0
Enter the third coordinates : 10 30
Enter coordinates of the key : 10 15
INSIDE
```

- **Task 2**

  Amazing Job!! You just need to solve one more problem to help Aladdin reach the end. The floor of the cave is crawling with snakes and the lamp is mounted on a stone table at the end of the room. Help Aladdin get across the room without getting bitten by venomous snakes. The information about the snakes is written in a scroll. Help Aladdin reach the end of the room by providing him with the information from the scroll.

  Create **venom.py** which contains multiple functions.

  1. snake(str name, int length, int venom): This function should take as an input name of the snake, it's length and it's venom power (which tells how venomous it's venom is). It should create an object for the class Snake and store it in a list.
     (Hint: You have studied about objects and classes right? Is it possible to use list to store any object??)
  2. findByVenom(int Venom): This function takes venom as an input and outputs the names of all the snakes with the given venom power.(Min power:1, Max power: 10).
  3. findByLength(int Length): This function takes length as an input and outputs the names of all the snakes of the given length.(Min Length: 1, Max Length: 10).

  Read input from a file **snakes.txt.** The format of the file is as below-

```
5
A 3 6
B 4 4
C 2 8
D 9 3
E 4 8
3
V 4
V 8
L 2
```

The first line of the file represents the number of snakes in the database (n). The next n lines (name, length, venom) store the information of the snake.
The next number represents the number of queries (q). The next q lines are used to represent the query.
The format of query includes -

- V <venom>
- L <length>

For each query, print the name of the snake.

The output format is as follows -
```
B
C
E
C
```

P4. Board Game

Everybody loves a game!!
Consider a 12x12 chess board where each tile can be represented using two numbers x and y, where x is the row number and y is the column number. Consider rows to be (0-11) and columns to be (0-11).
A number of students are standing in these tiles. The location of these students is given as input to the program.
You need to perform 3 tasks -

- **Task 1**
  Store the location of students using a dictionary. Create a dictionary dict which stores x and y. Store all the locations of students in a list.
  (Hint : Can list store dictionary object??)
  The input is to be read from a file **students.txt.**
  Example Input File -
  students.txt

  ```
  5
  0 4
  0 6
  1 4
  2 3
  5 11
  ```

  The first line is the number of students n. Next n lines contain location of each student (x,y).
  Create file **board.py** to store the input.

- **Task 2**

Create a function called find_loc(int x1, int y1, int x2, int y2). Given the location of two students s1(x1,y1) and s2(x2,y2), display the number of steps required by s1 to reach s2. A student can either move one tile up or one tile towards the right. Make sure that location of s1 and s2 is already stored in the list created in Task 1. Print -1 if s2 is unreachable from s1.
Create file **board_steps.py,** which is an extension of the file **board.py,** which performs Task 2.

Input :
x1 y1 x2 y2
Output :
Numbers of steps

- **Task 3**
  Create a function called find_total(int x1, int y1, int x2, int y2). Given the location of two students s1(x1,y1) and s2(x2,y2), find the total number of students (inclusive of s1 and s2) lying on the same line as the two.
  Create file **board_total.py,** which is an extension of the file **board.py,** which performs Task 3.

  Input :
  x1 y1 x2 y2
  Output :
  Numbers of students

Consider the following example for better understanding -
find_loc(0,0,4,5)
Output: 9.
find_total(0,0,2,2).
Output: 3 (Considering that apart from 0,0 and 2,2 there exists one more point on the x=y line).

P5. Complex Numbers

Create a file **complex.py** which includes a class implementation for performing operations on complex numbers. Create a class **Complex** with member variables **real** and **imag** to store the real and imaginary part of a complex number. Python has operator overloading similar to C++. This is a good reference for it. Override one operator for each operation to be performed. The operations include -

- Print the complex number
  - __str__(...)
  - Should display the complex number in the format **"a+bi"**
- Adding two complex numbers and return a new complex number
  - __add__(...)
- Subtracting one complex number from another and return a new complex number
  - __sub__(...)
- Multiplying two complex numbers and return a new complex number

    - __mul__(...)
- Dividing one complex number with another and return a new complex number
    - __truediv__(...)

You need to print the result of addition(i.e c1 + c2), subtraction(i.e c1 - c2) , multiplication(i.e c1 * c2) and division(i.e c1 / c2) of the given two complex numbers.

The input to the code are two complex numbers, one on each line, which is to be read from a file (same as above question).

Example Input File -
numbers.txt
----
3 4
4 -2
----
*The above file represents two complex numbers 3+4i and 4-2i.

Your code should call each function in the same sequence as above and display the output on stdout in the following form-
3+4i (first number)
4-2**i** (second number)
7+2**i** (addition)
-1+6**i** (subtraction)
20+10**i** (multiplication)
0.2+1.1**i** (division)

General Instructions

- Make sure you know what you write, you might be asked to explain your code at a later point in time
- Your code may be tested on hidden test cases
- Grading is done automatically, so please make sure you stick to naming conventions
- The deadline for this lab is **Sunday, 5th August, 23:55.**

Submission Instructions

After creating your directory, package it into a tarball **<rollno1>-<rollno2>-<rollno3>-outlab3.tar.gz** in ascending order. Submit once only per team from the moodle account of smallest roll number.

The directory structure is as follows-

- `<rollno1>-<rollno2>-<rollno3>-outlab3/`
    - `P1/`

        - dotcodes.sh
        - fakecodes.sh
    - P2/
        - strings.py
    - P3/
        - triangle.py
        - venom.py
    - P4/
        - board_steps.py
        - board_total.py
    - P5/
        - complex.py