COMP10200: Assignment 1, part a

Overview

This assignment will get you using Python **lists** and **sets** along with **related arrays** in **numpy**. Try to keep your solution short – after all the user input is done, you should not need more than about 15 to 20 lines of code (not including blank lines and comments) to do the processing and output. Pay attention to the detailed instructions below and make sure you cover everything.

Input

Get a list of player names from the user. Let the user decide how many players there should be. Don't let the user enter duplicate or empty names. Use **string methods** to strip leading and trailing whitespace and to make sure each name is capitalized. Make sure your code is efficient (use a **set**). At the end of the process, the names should be stored in a new random order in a **numpy array**.

Related Arrays

Now see how quickly the users can hammer on the Enter key. Use the **input** function to get a bunch of inputs from each user. Use the Python **time** function in the **time** module to compute the interval between each input. Intervals should be rounded to 3 decimal places. Let the user decide how many time intervals there should be.

Below is an example of a piece of code that computes the time (without rounding) for a single interval. You'll have to adapt this for your own code.

```
import time
input()
time1 = time.time()
input()
time2 = time.time()
print(time2-time1,"seconds")
```

You can use any data structures you like during the process of collecting times, but at the end you should convert the data so that it is stored in a two-dimensional **numpy array** with each row containing the corresponding user's times. This array is related (parallel) to the names array. Now sort the two parallel arrays so that they are both in **alphabetical order** by player name. Use numpy methods for this.

Output

Using numpy (no loops) compute and output the **mean time** for each player (**rounded** to 3 decimal places) then report the fastest and slowest mean times (**rounded** to 3 decimal places) as well as the fastest and slowest single interval times (rounded to 3 decimal places). Include the names of the players who achieved these times.

Finally, you should output both the names and the times array for debugging.

Example Run

```
How many players? 5
How many time intervals? 9

Enter 5 player names

Name can't be empty
Khadija
Lee
 lee
Name already entered
Lee
Name already entered
  LEE
Name already entered
alicia
lachlan
Margie
```

Notice that leading and trailing spaces as well as capitalization is ignored.

```
Lee's turn. Press enter 10 times quickly.
(10 blank lines)
Khadija's turn. Press enter 10 times quickly.
(10 blank lines)
Lachlan's turn. Press enter 10 times quickly.
(10 blank lines)
Margie's turn. Press enter 10 times quickly.
(10 blank lines)
Alicia's turn. Press enter 10 times quickly.
(10 blank lines)
```

The playing order is random. Names are capitalized.

The output order is alphabetical.

```
Names ['Alicia' 'Khadija' 'Lachlan' 'Lee' 'Margie']
Mean times: [0.176 0.361 0.194 0.116 0.174]
Fastest Average Time: 0.116 by Lee
Slowest Average Time: 0.361 by Khadija
Fastest Single Time: 0.107 by Lee
Slowest Single Time: 0.448 by Margie

['Alicia' 'Khadija' 'Lachlan' 'Lee' 'Margie']
[[0.127 0.257 0.112 0.239 0.113 0.239 0.129 0.239 0.128]
[0.43 0.447 0.432 0.433 0.383 0.4 0.224 0.256 0.241]
[0.128 0.32 0.112 0.288 0.111 0.273 0.128 0.255 0.129]
[0.112 0.112 0.113 0.111 0.127 0.114 0.132 0.107 0.112]
[0.127 0.241 0.112 0.049 0.112 0.239 0.128 0.448 0.111]]
```

Raw data from numpy arrays.

Commenting

Follow the documentation conventions for the course. You'll find these in a python file in the Student Resources section of Canvas.

Handing In

Zip your Python script and hand it in to the dropbox.

Evaluation

Your code will be evaluated for Performance, Structure, and Documentation using the Rubric attached to the drop box.