CS 201 Homework #9

Deadline: November 23rd, 9:00 pm.
Save your code as lastname_homework9.py and submit on Edmodo.
Please, run your code before submitting.
If you get an error, try to fix it before submitting your homework.

If you already completed the task below, skip to Part II.

I) ROCK PAPER SCISSORS TOURNAMENT

For this problem, you will need to use a random number generator.
Add "import random" to the beginning of your code.
Use "random.randit(x, y)" to generate a random integer between x and y (included!)

Let's model a Rock-Paper-Scissors tournament. There are 30 players. Let's assume that each player uses one simple strategy: either always plays Rock, always Paper, or always Scissors.

Generate a player database list, which is a nested list that consists of *three* lists below:
(a) player names in string format that look like this: "player1", "player2", and so on (hint: to generate the names use for loop, string concatenation, and remember that str(1) is "1");
(b) player strategy (let's define Rock as 0, Paper as 1, and Scissors as 2) – the strategy should be chosen randomly (use randint).
(c) player points to determine the winner. Initially, everyone starts with zero, obviously.

Assume that there are 1000 duels in the tournament: for each duel, two players are *randomly* chosen (yes, the number of games may be different for different players; let's not worry about this problem for this exercise). If you win you get 3 points; if it's a tie, you get 1 point; if you lose you get 0 points. Use the player strategy sublist to determine the outcome of each duel. Update the player points sublist given the result of each duel.

After 1000 games finish, identify the player with most points and print the name of the player, the strategy that was used, and the final number of points.

If you already completed the task below, skip to Part III.

Part II: COUNTRIES DATABASE (the solution to this problem is posted on School Nova website)

For this problem, you will need to use the following information:

name	*capital*	*population*	*area*
Canada	Ottawa	37.6	3.86
Mexico	Mexico City	129.2	0.76
USA	Washington DC	327.2	3.80

Note: population is in millions, area is in millions of square miles.

1) Using alphabetical order (as in the table), create the following four lists: country names, capitals, population, and area. For this step (and this step only) you will need to manually type the data from the table.

2) Using the four lists that you just created, create country profile lists (for example, pCanada, pMexico, and pUSA) that look like this: ['Canada', 'Ottawa', 37.6, 3.86], and so on.

Do NOT create the country profile lists by manually typing the data! Instead use the four lists of country names, capitals, population, and area that you created earlier. Use .append() method to add data to the country profile lists.

3) Can you do step #2 using a for loop? Hints: you can use the fact that the index for Canada is 0, Mexico is 1, and USA is 2. You can also use IF statement to choose a relevant list name (for example,

```
if i == 0:
   pCanada = []
```

(please, do not use dictionaries for this step; we have yet to learn about them in 101).

4) Create a nested list that consists of all countries data; let's call it *countries*. Its structure should look like this: [pCanada, pMexico, pUSA], where each country profile list contains the country level data: name, capital, population, and area.

5) Using while loop ask for a user input about a variable of interest for a given country. For example, the user could type "What is the population of Canada?". Given the user input, provide the requested information using the *countries nested list*.

Hint: the command *in* works with strings just like it works with lists! For example, *"Canada" in "Canada population"* will be **True**.

If the user input is "What is the population of Canada?" the output that your code generates should look like this:

“The population of Canada is 37.6 million people.”

If the user input has a country or variable that is not part of the *countries* list, the output should be:

“There is no data for this query”.

Finally, the use should have an option to finish the program. If the user types “exit” (probably, a Windows user) or “quit” (probably, a Mac user), the program should terminate, wishing the user a good day.

Part III: COUNTRIES DATABASE 2.0

For this task, we continue using the data below with some additional information:

Variable	“capital”	“population”	“area”
Unit		“million people”	“million square miles”
Canada	Ottawa	37.6	3.86
Mexico	Mexico City	129.2	0.76
USA	Washington DC	327.2	3.80

Question 1

Previously (see posted homework #8 solution) we had the following code:

```
if v == 1:
   print(f"The capital of {cdata[c][0]} is {cdata[c][1]}")
elif v == 2:
   print(f"The population of {cdata[c][0]} is {cdata[c][2]} million people.")
else:
   print(f"The area of {cdata[c][0]} is {cdata[c][3]} square miles.")
```

This is potentially inefficient. What if we have thousands of variables describing a country?! Can you can get rid of the IF statement and reduce the previous 6 lines of code to just one, which prints essentially:

The <variable> of <country name> is <country & variable data> <variable units>.

The table above should help with this task.

Question 2

On the basis of the information that you already have, generate a new variable, “population density”, which is equal to population divided by area (with the corresponding unit “residents

per square mile"). Please, do this AFTER cdata is already created. *Append* new data to the existing dataset.

Question 3

Allow the user to add new countries to the dataset if the user types "add country" in the user query. In this case, the user must type a new country name, capital, population, and area. The population density can be found on the basis of the latter two variables. Test your code (for example, you can use data for Brazil: "Brazil", "Brasilia", 209.3, 3.29). Notice that you can ask the user to provide the new country data one variable/entry at a time or all at once; feel free to use the option that you like more. (One entry at a time is probably easier).

Question 4

Allow the user to delete a country (with all of its data) by typing "delete <country name>". For example, if the user types "delete Canada", all information about Canada will be deleted from the dataset. Verify that your code works.

Question 5

For this exercise type:

```
cn = ["Canada", "Mexico", "USA"]
cn2 = cn
cn_copy = cn.copy()
```

Now change one of the items in cn and explore how it affected cn2 and cn_copy.

Question 6

For this question use cdata nested list from the homework.
Create a copy of cdata: cdata_copy = cdata.copy()
Change an item in cdata and explore how it affected cdata_copy.

Try the following changes (one at a time):

```
cdata[0][2] = 999
cdata[0] = 777
```

What can you conclude on the basis of Questions 5 and 6? (You can type your response using Python comments)