**Course: Introduction to Data Science (DS2006) - Laboratory 04**

**Student:**

Considering the context of Lecture 4 involving the legal battle regarding Battle of Dices, we need to make improvements in it. Right now our main limitation is that every time we run a new round the information is lost forever and we are not able to use that information to create a logfile for example. For this reason we are going to make some improvements on it.

Open the file named [**battle-of-dices-bad.py**](http://battle-of-dices-cooler.py)you created in Laboratory 03. Save it as [**battle-of-dices-bad-not-rigged.py**](http://battle-of-dices-not-rigged.py).Refactor the code in [**battle-of-dices-bad-not-rigged.py**](http://battle-of-dices-not-rigged.py) to:

* **Task 1**: Use several variables to store the values of each for roll by each player in each round. For example:

# Round 1

player1\_\_round1\_roll = random.randint(1, 6)

player2\_\_round1\_roll = random.randint(1, 6)

# Round 2

player1\_\_round2\_roll = random.randint(1, 6)

player2\_\_round2\_roll = random.randint(1, 6)

# … Keep using this logic for all rounds.

* **Task 2**: Implement a new piece of code when the game is over, that shows the information round by round, of which rolls each player had. One possible suggestion would be something like is shown in Figure 1. Please note that “...” means continue with the same logic):

-----------------------------------------

| Round | 1 | 2 | 3 | 4 | 5 | 6 | …

-----------------------------------------

| Player 1 | 6 | 4 | 5 | 6 | 4 | 5 | …

-----------------------------------------

| Player 2 | 1 | 2 | 3 | 1 | 2 | 3 | …

-----------------------------------------

Figure 1 - An example of what the game summary

could look like in the [battle-of-dices-bad-not-rigged.py](http://battle-of-dices-bad-not-rigged.py) .

As we have seen by implementing tasks 1 and 2, this is not the best way to handle situations when we need to store several values. A better option is to use lists. For the next tasks, open the file named [**battle-of-dices-better.py**](http://battle-of-dices-cooler.py)you created in Laboratory 03. Save it as [**battle-of-dices-not-rigged.py**](http://battle-of-dices-not-rigged.py). Refactor the code in [**battle-of-dices-not-rigged.py**](http://battle-of-dices-not-rigged.py) to:

* **Task 3**: In [**battle-of-dices-not-rigged.py**](http://battle-of-dices-not-rigged.py) we want to replace the current method you have that is similar (you may be using a different type of dice) to the code shown in Figure 2 to use Lists to store all the rolls from player 1 and player 2.

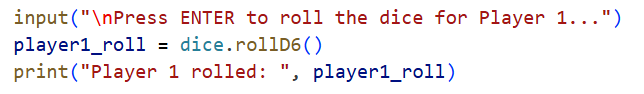


Figure 2. Code snippet to roll a dice for player1.

* **Task 4**: In [**battle-of-dices-not-rigged.py**](http://battle-of-dices-not-rigged.py) after the game is over we want to print a summary of the game that is going to be a table-like summary of the game, it should have information about the rounds, the player, the type of dice rolled. You can choose the layout you want to present this information, but if you want you may follow the same layout presentation for Task 2 in this list. The important thing here is that you use the values stored in the list to present the information.
* **Task 5**: In [**battle-of-dices-not-rigged.py**](http://battle-of-dices-not-rigged.py) we now want to save the summary of the table to a file. You should ask the user for the name of the file they want to use.

For the next set of activities we are going to implement some short python code snippets that were used in the Kahoot to better understand some of the things that were going on. For this last part of Laboratory 04 you should create a file named [**python-revision-01.py**](http://python-revision-01.py) .

* **Task 6**: What is the output of the code shown in Figure 3?

x = 5

y = 3

x = y

print(x)

Figure 3. Code snippet

**\*\*\*\* CONTINUES on the next page \*\*\*\*\*\*\*\*\*\*\***

* **Task 7**: What is the output of the code shown in Figure 4?

x = 5

y = 3

y = x

print(y)

Figure 4. Code snippet

* **Task 8**: What is the output of the code shown in Figure 5?

x = 10

y = x

x = 20

print(y)

Figure 5. Code snippet

* **Task 9**: What is the output of the code shown in Figure 6?

x = 10

y = x

x = 20

y = x

print(y)

Figure 6. Code snippet

* **Task 10**: What is the output of the code shown in Figure 7? Why do you get that result? Is it what you were expecting?

x = "5"

y = 3

print (x \* y)

Figure 7. Code snippet

**\*\*\*\* CONTINUES on the next page \*\*\*\*\*\*\*\*\*\*\***

* **Task 11**: What is the output of the code shown in Figure 8? Why do you get that result? Is it what you were expecting?

x = 7

y = 2

print("x / y")

Figure 8. Code snippet

* **Task 12**: What is the output of the code shown in Figure 9?

x = 7

y = 2

print(x / y)

Figure 9. Code snippet

* **Task 13**: What is the output of the code shown in Figure 10? What is the meaning of the symbol // ?

x = 12

y = 5

print(x // y)

Figure 10. Code snippet

* **Task 14**: What is the output of the code shown in Figure 11? What is the meaning of the symbol % ?

x = 7

y = 2

print(x % y)

Figure 11. Code snippet

* **Task 15**: What is the output of the code shown in Figure 12? What is the meaning of the symbol % ?

x = 2

y = "3"

print(str(x) + y)

Figure 12. Code snippet

* **Task 15**: What is the output of the code shown in Figure 13?

x = "2"

y = "3"

print(x + y)

Figure 13. Code snippet

* **Task 16**: What is the output of the code shown in Figure 14?

x = 10

if x > 5:

print("Big")

else:

print("Small")

Figure 14. Code snippet

* **Task 17**: What is the output of the code shown in Figure 15?

x = 100

if x > 5:

print("Big")

else:

print("Small")

Figure 15. Code snippet

* **Task 18**: What is the output of the code shown in Figure 16?

x = -10

if x > 5:

print("Big")

else:

print("Small")

Figure 16. Code snippet

**\*\*\*\* CONTINUES on the next page \*\*\*\*\*\*\*\*\*\*\***

* **Task 19**: What is the output of the code shown in Figure 17?

x = 3

if x == 5:

print("Five")

elif x <= 5:

print("Less")

else:

print("More")

Figure 17. Code snippet

* **Task 20**: What is the output of the code shown in Figure 18?

x = 3

if x == 5:

print("Five")

elif x <= 5:

print("Less")

else:

print("More")

Figure 18. Code snippet

* **Task 21**: What is the output of the code shown in Figure 19?

x = 5

if x == 5:

print("Five")

elif x <= 5:

print("Less")

else:

print("More")

Figure 19. Code snippet

**\*\*\*\* CONTINUES on the next page \*\*\*\*\*\*\*\*\*\*\***

* **Task 22**: What is the output of the code shown in Figure 20?

x = 10

y = 2

if x > 20:

print("A")

elif y > 5:

print("B")

elif x > 0:

print("C")

else:

print("D")

Figure 20. Code snippet

* **Task 23**: What is the output of the code shown in Figure 21?

x = 10

y = 20

if x > 20:

print("A")

elif y > 5:

print("B")

elif x > 0:

print("C")

else:

print("D")

Figure 21. Code snippet

* **Task 24**: What is the output of the code shown in Figure 22?

x = -5

y = -3

if x > 20:

print("A")

elif y > 5:

print("B")

elif x > 0:

print("C")

else:

print("D")

Figure 22. Code snippet

* **Task 25**: What is the output of the code shown in Figure 23?

names = ["Laura", "Yasmin"]

for each\_name in names:

print(each\_name)

Figure 23. Code snippet

* **Task 26**: Considering the code from Figure 24, change it to ask the user for a new name to add to the list:

names = ["Laura", "Yasmin"]

# Add your code here

for each\_name in names:

print(each\_name)

Figure 24. Code snippet