**Course: Introduction to Data Science (DS2006) - Laboratory 03**

**Student: Philip Haglund**

**Task 1**: In the Battle of the dices, replace the code shown in Figure 1 with the code shown in Figure 2. What happens when you run the code with that modification?

print("Player 1 rolled: ",roll1)

print("Player 2 rolled: ",roll2)

Figure 1 - Code example.

print("Player 1 rolled: " + str(player1\_roll))

print("Player 2 rolled: " + str(player2\_roll))

Figure 2 - Another code example.

A screenshot of a computer screen

AI-generated content may be incorrect.

The code sent along came like this and I’ll tell you that its hard to print a undeclared variable.

A screen shot of a computer program

AI-generated content may be incorrect.

I’ve taken the liberty of re-arranging the code into the figure above.

The result is as follows:

A screenshot of a computer screen

AI-generated content may be incorrect.

It seems that converting the roll into a string and simply adding it works perfectly fine.

**Task 2**: Replace the code shown in Figure 2 with the code shown in Figure 3. What happens when you run the code with that modification?

print("Player 1 rolled: " + player1\_roll)

print("Player 2 rolled: " + player2\_roll)

Figure 3 - Yet another code example.

A screen shot of a computer program

AI-generated content may be incorrect.

Removing the string conversion seems to not have pleased the IDE so much. As it doesn’t run, and tells us that it needs the integer formated correctly before joining them together.

**Task 3**: Revert back to the code shown in Figure 1 and now replace the section of the code shown in Figure 4 so that it first checks if the 1st player won the round (i.e. got the higher score) if not checks if the 2nd player won the round, else assume the players had a tie.

if player1\_roll > player2\_roll:

print("Player 1 wins this round!")

print("Because ", player1\_roll," is greater than ", player2\_roll)

elif player1\_roll == player2\_roll:

print("Amaaazzinng! This round has a tie!")

else:

print("Player 2 wins this round!")

print("Because ", player2\_roll," is greater than ", player1\_roll)

Figure 4 - Code example used to check who won the round.

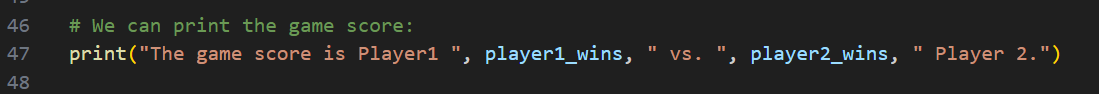
A screenshot of a computer screen

AI-generated content may be incorrect.

**Task 4**: Add variables to get information about who won the current round and then print the result of the round to the screen.

A black background with white text

AI-generated content may be incorrect.



A black background with white text

AI-generated content may be incorrect.

This was already implemented.

**Task 5**: What happens when you change the way you add to the variables created in Task 4 to use the format **variable += 1** instead of **variable = variable + 1**.

For example:

player1\_wins += 1

instead of

player1\_wins = player1\_wins + 1

Nothing changes. “player1\_wins += x” is just a simplified way to say “player1\_wins = player1\_wins + x”

A screen shot of a computer code

AI-generated content may be incorrect.

**Task 6**: Change the content of the messages shown when a player wins (in Figure 5) to something cooler!

# Now we need to check if either player won.

if player1\_wins == 3:

print("Player 1 is the newest Battle of Dices Champion! ")

elif player2\_wins == 3:

print("Player 2 is the newest Battle of Dices Champion! ")

else:

print("This heated Battle of Dices is still going on! Who will win? ")

Figure 5 - Code example used to check the winner of the game.

A screenshot of a computer screen

AI-generated content may be incorrect.

Hope this qualifies! :b

**Task 7**: Create a new file called **battle-of-dices-bad.py .** In this file make sure the game runs for as many rounds as needed without using any type of loop-related control structure.

You can have enough loops where its statistically unlikely that someone wont win but its not really feasible without a loop structure. Either through loops or a function that recalls itself or other such methods. I will do the bad practice of pasting the game loop 10 or so times.

I started by modifiying the win condition and adding a exit() to end the program once someone wins. So even if we have 20 sessions we can exit the program once a player win

A screenshot of a computer screen

AI-generated content may be incorrect.

I pasted the code snippet that ran the rounds 10 times making sure we have a pretty good chance of finishing the game before we run out of them.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

^^

Figure represents code I copied for each round.

**Task 8**: In **battle-of-dices-bad.py** replace all the calls you are making to roll a D6 to roll a different dice number. Choose between a D4, D8, D12, D20 or D100 and make both players use the same dice you choose in all rounds. I.e. replace all the d6 roll calls to the dice you choose roll calls.

A black background with white text

AI-generated content may be incorrect.

I replaced the rolls with D100’s.

A screenshot of a computer

AI-generated content may be incorrect.

**Task 9**: Reflect and write your impressions about refactoring (i.e. changing the code) in the battle-of-dices.bad.py without using loops.

Well this is clearly a horrible way to write code. I would change it out for a while loop, that just continues until player1 or player2 gets 3 wins. It leaves us with a single code block that we can easily change and modify.

**Task 10**: Create a new file called **battle-of-dices-better.py** . In this file, refactor the code to use a loop control structure with while.

A screenshot of a computer program

AI-generated content may be incorrect.

^^

Figure is after completing Task 11, 12, and 13.

**Task 11**: Make the necessary changes in the code of **battle-of-dices-better.py** to allow the program to print the number of rounds it took to a player win the game.

I did this task by declaring a rounds variable then adding one each time we played a round to it. Very similar to how we did results normally, then I just printed this number along with the win statement.

**Task 12**: Refactor the dice.py file we created for the last lab to provide **functions** for rolling common dice: d4, d6, d8, d12, d20, d100. Each dice roll should have their own function as illustrated in Figure 6.

A screenshot of a computer program

AI-generated content may be incorrect.

**Task 13**: Refactor the code in **battle-of-dices-better.py** to make use of the function calls from your dice class.

A screen shot of a computer program

AI-generated content may be incorrect.

**Task 14**: Create a new file called **battle-of-dices-cooler.py** and refactor the code from **battle-of-dices-better.py** to roll 2 dice (of different sizes) for each player. The winner of the round will be decided by the sum of the values obtained for each player.

A black screen with colorful text

AI-generated content may be incorrect.

I simply defined 2 new variable per player, one for each type or role and kept the old “playerX\_roll” as the total to line it up with the logic for the win condition. Then I rolled the dices into these new variables and combined them into the old “playerX\_roll”.

A screenshot of a computer

AI-generated content may be incorrect.