**Course: Introduction to Data Science (DS2006) - Laboratory 07**

**Student: Philip Haglund**

We want to expand on our Multiplayer Battle of Dices project to gather more information about our players.

* **Task 1: Think and reflect individually** about what are the necessary change(s) in your current code to implement the additional information from players in the multiplayer version of the Battle of the dices? What do you think will happen if we need to keep adding more things related to the players? (If you did this activity in class, just paste your individually submitted answer here)

Well so far we’ve worked a lot with the concept of implementing bad and good coding practises. When we needed just the player name storing them all in a list seemed reasonable but now that were storing two we probably want to look at classes or dictionaries. In the example provided in the slides dictionaries seems to be a very viable method.

* **Task 2: Think and reflect within your group** about what are the necessary change(s) in your current code to implement the additional information from players in the multiplayer version of the Battle of the dices? What do you think will happen if we need to keep adding more things related to the players? (If you did this activity in class, just paste your team submitted answer here)

I was not in class, and did not participate in a group reflection.

Before we implement the dictionaries in our multiplayer battle of dice, it is a good idea that we practice how to use dictionaries and their functionalities a little more.

* **Task 3:** Create a file named[**inventory.py**](http://multiplayer-battle-of-dices.py)and implement the code shown in Figure 1.

#Keep track of inventory:

inventory = {

"apple": 50,

"banana": 30,

"orange": 20

}

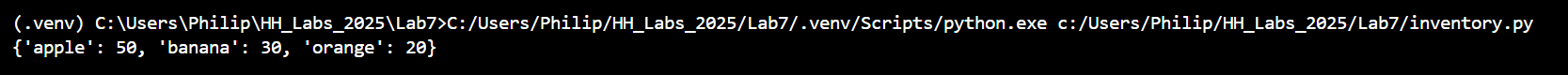
print(inventory)

**Figure 1. Code Snippet**

A screen shot of a computer program

AI-generated content may be incorrect.

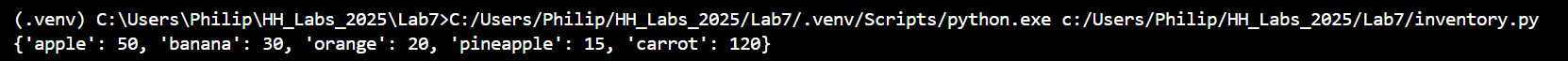
* **Task 4:** What is the output of the code shown in Figure 1**?**

****

* **Task 5:** Add two more items to the inventory dictionary in the [inventory.py](http://inventory.py) file.

I added 15 pineapples and 120 carrots.

* **Task 6:** What happens when you print after making the changes in task 5?



* **Task 7:** Improve the implementation of the inventory dictionary by allowing the user to:
  + Choose if they want to add, remove or check quantities for a certain item.
  + Choose by typing the item name in the dictionary, i.e. the key, for example “banana”, which item should be added or removed.
  + See the current stock for all items.

A computer screen shot of a program code

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

All the functions work as intended.

Now we are going to replicate the shallow and deep copy problems we have seen in Lecture 07 when using lists and dictionaries together.

* **Task 8:**  Implement the code shown in Figure 2 in a file called **single\_place.py**. Add a city you have visited before and show the output here.

visited\_places = {

"city": "",

"country": "",

"year": "",

}

visited\_places["city"] = input("Enter city name: ")

visited\_places["country"] = input("Enter country name: ")

visited\_places["year"] = input("Enter the year: ")

print(visited\_places)

**Figure 2. Code Snippet**

A computer screen shot of white text

AI-generated content may be incorrect.

* **Task 9:**  Implement the code shown in Figure 3 in a file called **two\_places.py**. Add two cities you have visited before and show the output here.

visited\_places = {

"city": "",

"country": "",

"year": "",

}

my\_visited\_places = []

for i in range(0, 2):

visited\_places["city"] = input("Enter city name: ")

visited\_places["country"] = input("Enter country name: ")

visited\_places["year"] = input("Enter the year: ")

my\_visited\_places.append(visited\_places)

print(my\_visited\_places)

**Figure 3. Code Snippet**

**A black background with white text

AI-generated content may be incorrect.**

* **Task 10:**  Implement the code shown in Figure 4 in a file called **better\_two\_places.py**. Add the exact information you added in task 9 and show the output here.

visited\_places = {

"city": "",

"country": "",

"date": "",

}

my\_visited\_places = []

for i in range(0, 2):

# Make a copy of the dictionary template:

places = visited\_places.copy()

places["city"] = input("Enter city name: ")

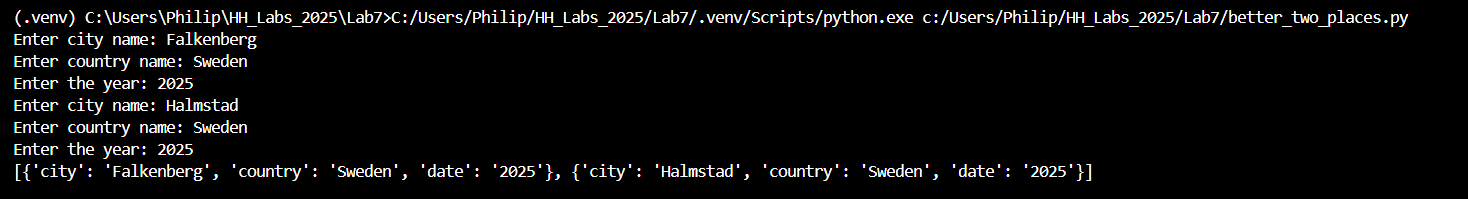
places["country"] = input("Enter country name: ")

places["date"] = input("Enter the year: ")

my\_visited\_places.append(places)

print(my\_visited\_places)

**Figure 4. Code Snippet**



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Continues on the next page \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

* **Task 11:**  Implement the code shown in Figure 5 in a file called **many\_two\_places.py**. Add information about two cities you visited in different years and show the output here.

visited\_places = {

"city": "",

"country": "",

"dates": [],

}

my\_visited\_places = []

for i in range(0, 2):

# Make a copy of the dictionary template:

places = visited\_places.copy()

places["city"] = input("Enter city name: ")

places["country"] = input("Enter country name: ")

times = input("How many times you visited?")

for j in range(0, int(times)):

year = input(f"Enter the year of the {j+1} time you went there: ")

places["dates"].append(year)

my\_visited\_places.append(places)

print(my\_visited\_places)

**Figure 5. Code Snippet**

A black rectangular frame with colorful dots

AI-generated content may be incorrect.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Continues on the next page \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

* **Task 12:**  Implement the code shown in Figure 6 in a file called **better\_many\_two\_places.py**. Add the exact information you added in Task 11 about two cities you visited in different years and show the output here.

import copy

visited\_places = {

"city": "",

"country": "",

"dates": [],

}

my\_visited\_places = []

for i in range(0, 2):

# Make a deep copy of the template for this player

places = copy.deepcopy(visited\_places)

places["city"] = input("Enter city name: ")

places["country"] = input("Enter country name: ")

times = input("How many times you visited?")

for j in range(0, int(times)):

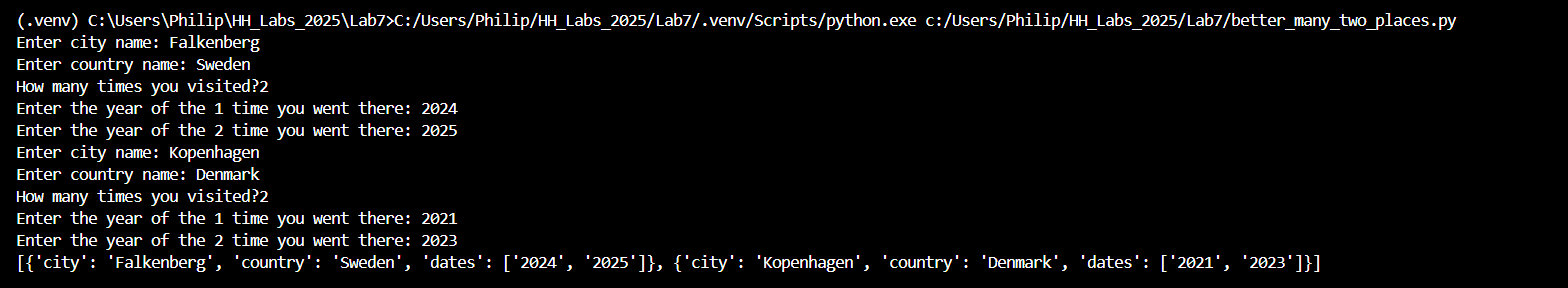
year = input(f"Enter the year of the {j+1} time you went there: ")

places["dates"].append(year)

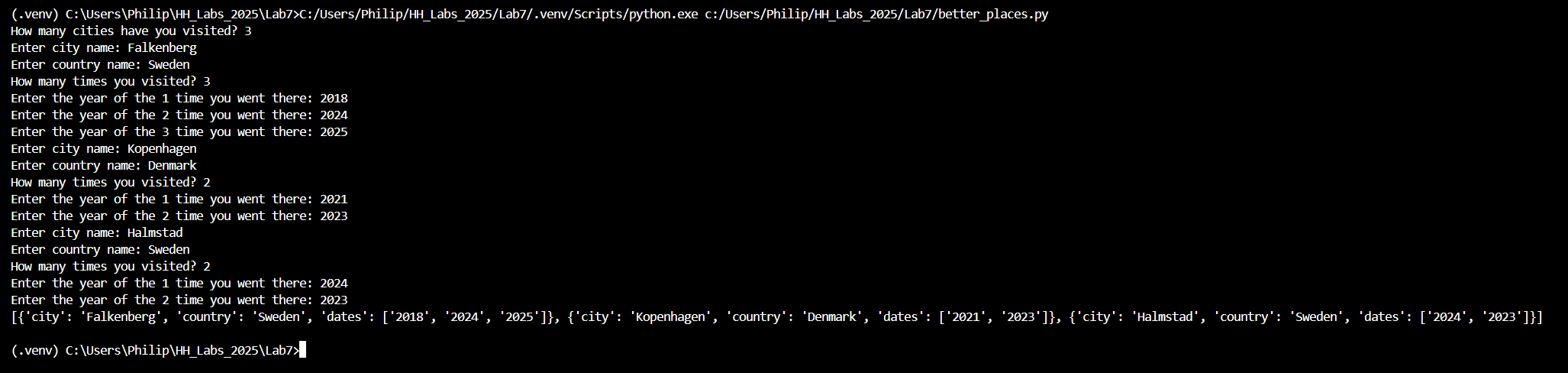
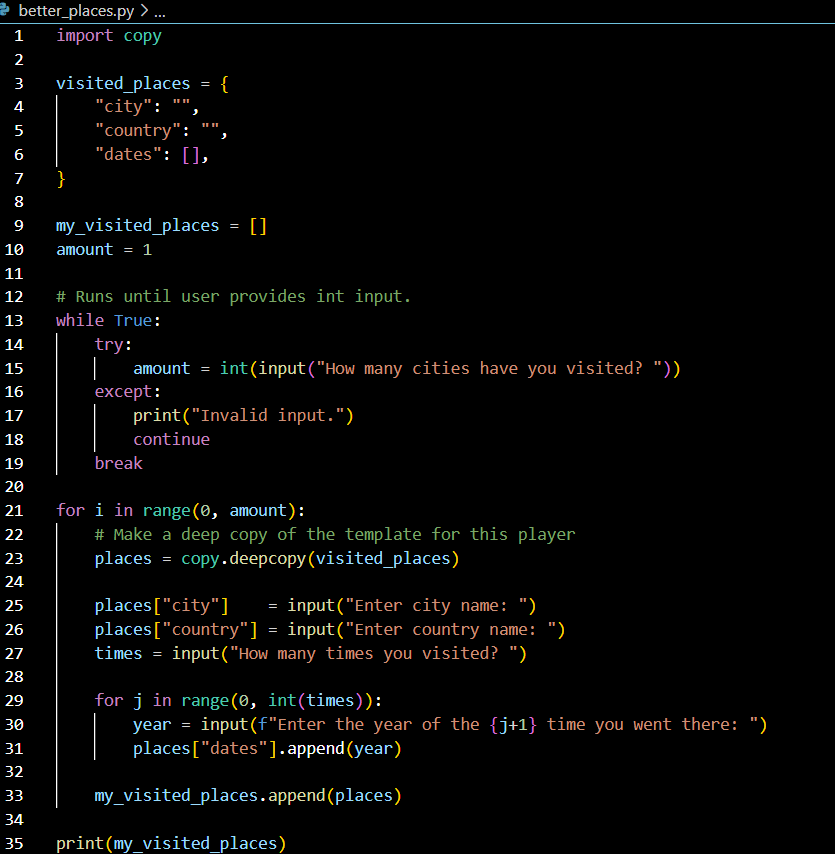
my\_visited\_places.append(places)

print(my\_visited\_places)

**Figure 6. Code Snippet**



* **Task 13:**  Implement the necessary code to allow the user to choose how many cities they want to add and then let them add them. This should be done in a file called **better\_places.py**.

****

Now it is time to return to our battle of dices project!

* **Task 14:** Create a file named[**multiplayer-battle-of-dices-dict.py**](http://multiplayer-battle-of-dices.py)and implement the solution presented in the slides of Lecture 07.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

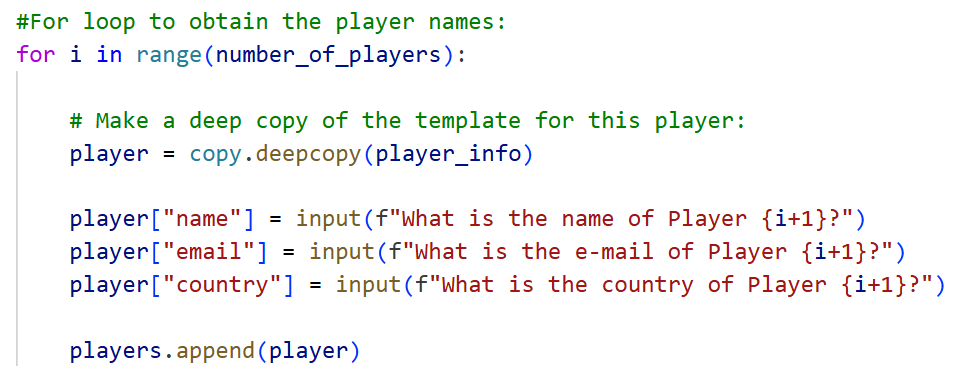
AI-generated content may be incorrect.

A screenshot of a computer

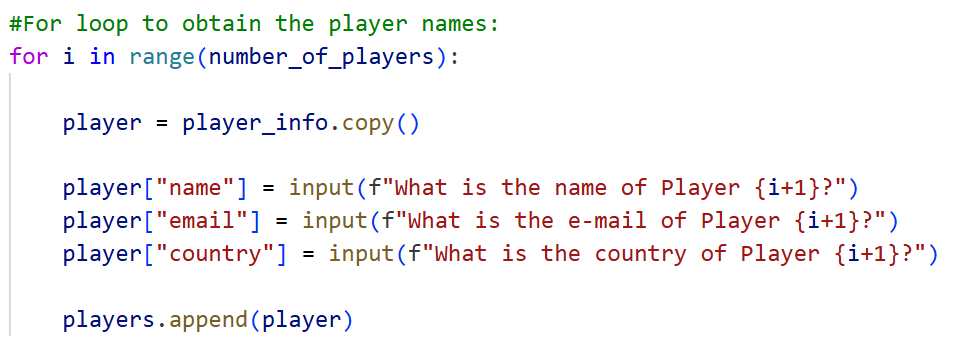
AI-generated content may be incorrect.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Continues on the next page \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

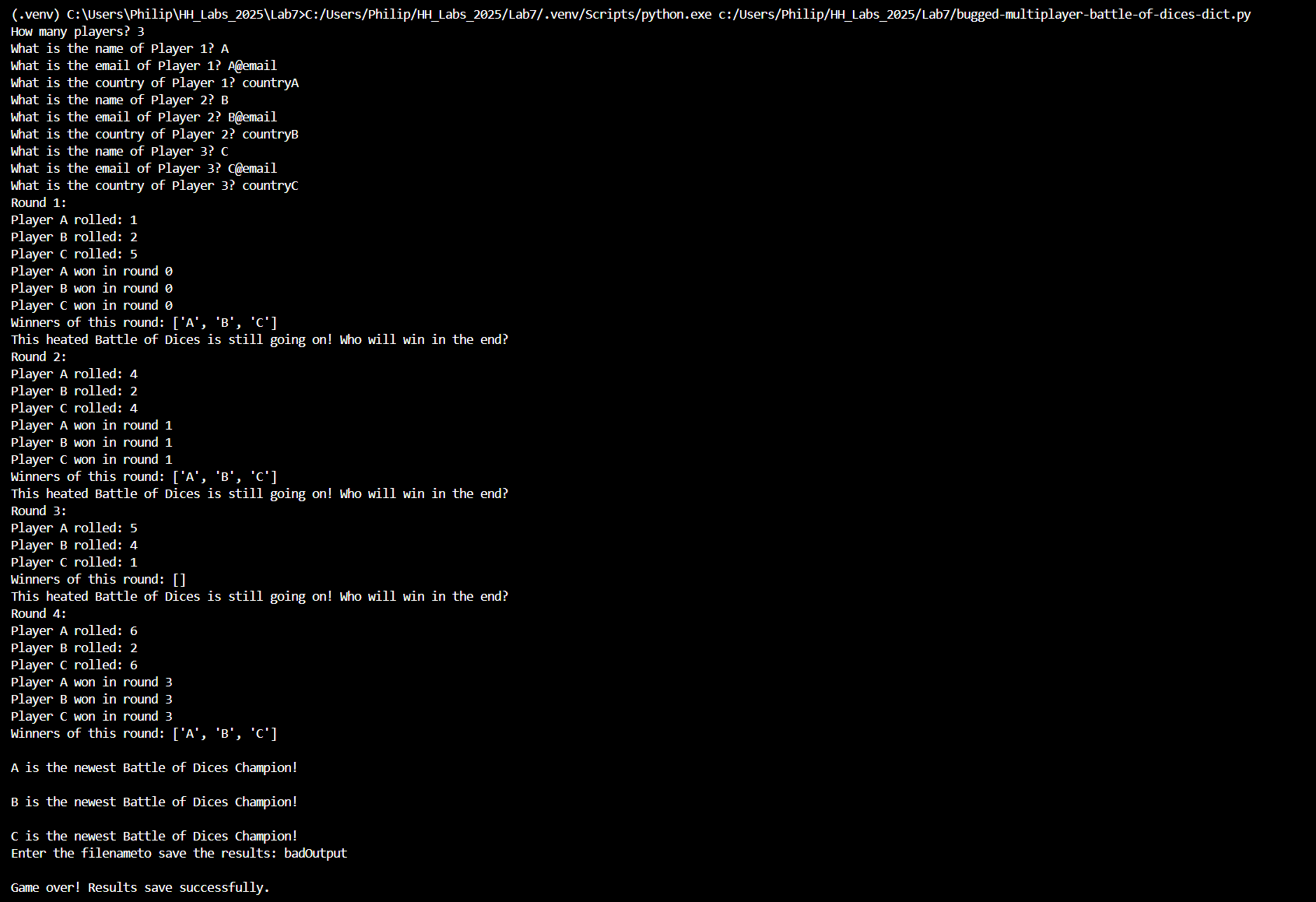
* **Task 15:** Create a file named[**bugged-multiplayer-battle-of-dices-dict.py**](http://multiplayer-battle-of-dices.py)**.** Replace the code from Figure 7 with the code from Figure 8. Run this code for 3 or more players and explain what happened.



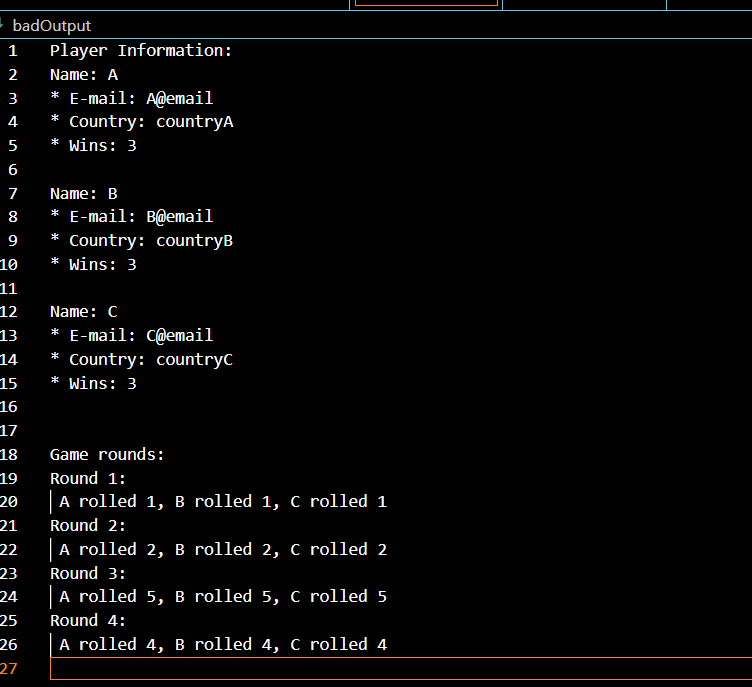
**Figure 7. Code Snippet**



**Figure 8. Code Snippet**



Because its not running a deep copy then all of them share the same list for their rolls. Because the player\_info is copied, copying the shared list for rolls along with it.



We can see that they all share the same roll. Since we append to the list, the same shared list then our list will be [1,2,5] after round one. All of them will think that they rolled 1 for the first round, 2 for the second, 5 for the third round. Having the all add to the same list is creating a quite easy to spot issue that a deepcopy wouldn’t do.

* **Task 16:** Create a file named[**cooler-multiplayer-battle-of-dices-dict.py**](http://multiplayer-battle-of-dices.py)and refactor the code from task 14 to use the same ideas from your previously developed [**cooler-multiplayer-battle-of-dices.py**](http://multiplayer-battle-of-dices.py).

This code runs much better and is a lot easier to work with. Nestled lists quickly become confusing to keep track of how our infrastructure runs, meanwhile this can easily hold a huge amount of data with little confusion.

A screenshot of a computer program

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Test run:

A screenshot of a computer

AI-generated content may be incorrect.

Log^^

A screenshot of a computer

AI-generated content may be incorrect.

Save file^^ (I have added wins too.)

A screenshot of a computer

AI-generated content may be incorrect.

Heres one with wins added too.