**Exercise: 7**

**Name: Pekka Lehtola**

**How many tasks did you do: 6**

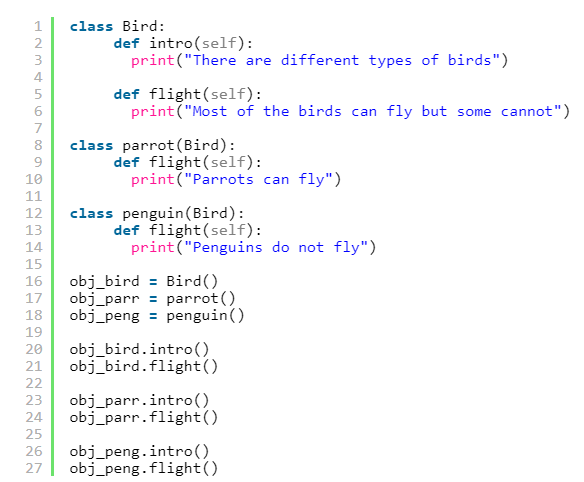
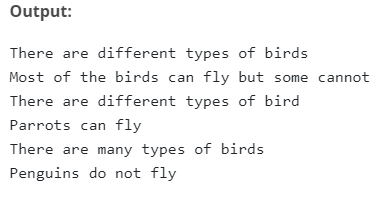
**Were the tasks easy, ok, difficult: ok**

**Do you need help/comments in any task (if yes, to which ones): -**

1. Answer the following questions.

a. What does polymorphism (in object-oriented programming) mean? Also give a short (coding) example, e.g. google for examples).

**Polymorphism lets us define methods in the child class that have the same name as the methods in the parent class**

**** ****

b. What is a class variable and how are they used?

**A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods.**

c. What is an instance variable and how is it different from the class variable?

**A variable that is defined inside a method and belongs only to the current instance of a class.**

d. What is a UML sequence diagram used for?

**UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration.**

e. What is a lifeline in UML sequence diagrams?

**A lifeline represents an individual participant in the Interaction.**

2. Multiple choice:

a. In an inheritance relationship, the \_\_\_\_\_\_ is the general class.

i. Child class

ii. Subclass

iii. Superclass

iv. Specialized class

b. In an inheritance relationship, the \_\_\_\_\_\_ is the specialized class:

i. Superclass

ii. Master class

iii. Parent class

iv. Subclass

c. Let’s say we have two classes in our program: BankAccount and SavingsAccount. Which one of them would most likely be the subclass?

i. BankAccount

ii. SavingsAccount

iii. Neither of them

iv. Both of them.

d. Which one of the option you will use if you want to check whether an object is an instance of a class.

i. The instance operator

ii. The is\_object\_of function

iii. The isinstance function

iv. There is not a way to check that at all.

e. Which one of the UML diagrams is a behavioral diagram?

i. Class diagram

ii. Sequence diagram

iii. Object diagram

iv. Deployment diagram

f. Which one of the UML diagrams is a structural diagram?

i. Use case diagram

ii. State machine diagram

iii. Activity diagram

iv. Composite structure diagram

g. In UML class diagrams, what does the notation \* mean.

i. Multiplication operation

ii. Power of operation

iii. Multiplicity 0..n

iv. Multiplicity 0..1

3. True or false?

a. It is not possible to call a superclass’s \_\_init\_\_ method from a subclass’s \_\_init\_\_ method.

**False**

b. A subclass never inherits any methods or attributes from the superclass.

**False**

c. A superclass can inherit methods from subclass, if they have been denoted with pass\_to\_super function.

**False**

d. In a subclass it is possible to have methods and attributes in addition to those that the subclass inherits from superclass.

**True**

e. In Python, multiple inheritance does not exist.

**False**

f. Aggregation and composition shall never be used in UML class diagrams.

**False**

g. Aggregation and composition mean exactly the same thing in UML class diagrams.

**False**

4. Take the code main.py from Itslearning, do not change it (except for the Author and code after line 42; obviously for testing purposes you can e.g. comment the methods out that you have not yet implemented, but in your final return the main.py shall be like given). Implement classes Card and Deck (in their own modules) so that the main.py can be run and the output is exactly the same than in file Ex7\_task4\_output.txt (of course, shuffled deck and drawn cards can be different). (You can look for help here: https://medium.com/@anthonytapias/build-a-deck-of-cards-with-oo-pythonc41913a744d3). Then implement one of the games described below (or even all of them if you like).

a. Draw 3 cards, highest value wins. Announce results (have clear output print). Hopefully have a re-draw if there are ties.

b. Implement card game Twenty-one (= Ventti in Finnish) or Blackjack for as many players as you like. Announce results clearly. Notice, you do not necessarily need a Player class in this game (but you are allowed to have it).

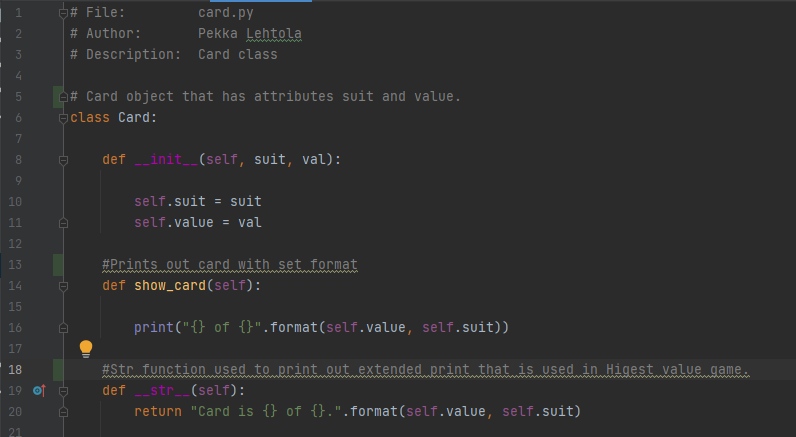
c. Create a Player class as well. Each player shall have a name and hand of cards. Then implement any card game you like (an existing one or create your own). Comment the code clearly and explain the rules as well. Have clear output prints.

Screen capture of Task 4

Deck class:



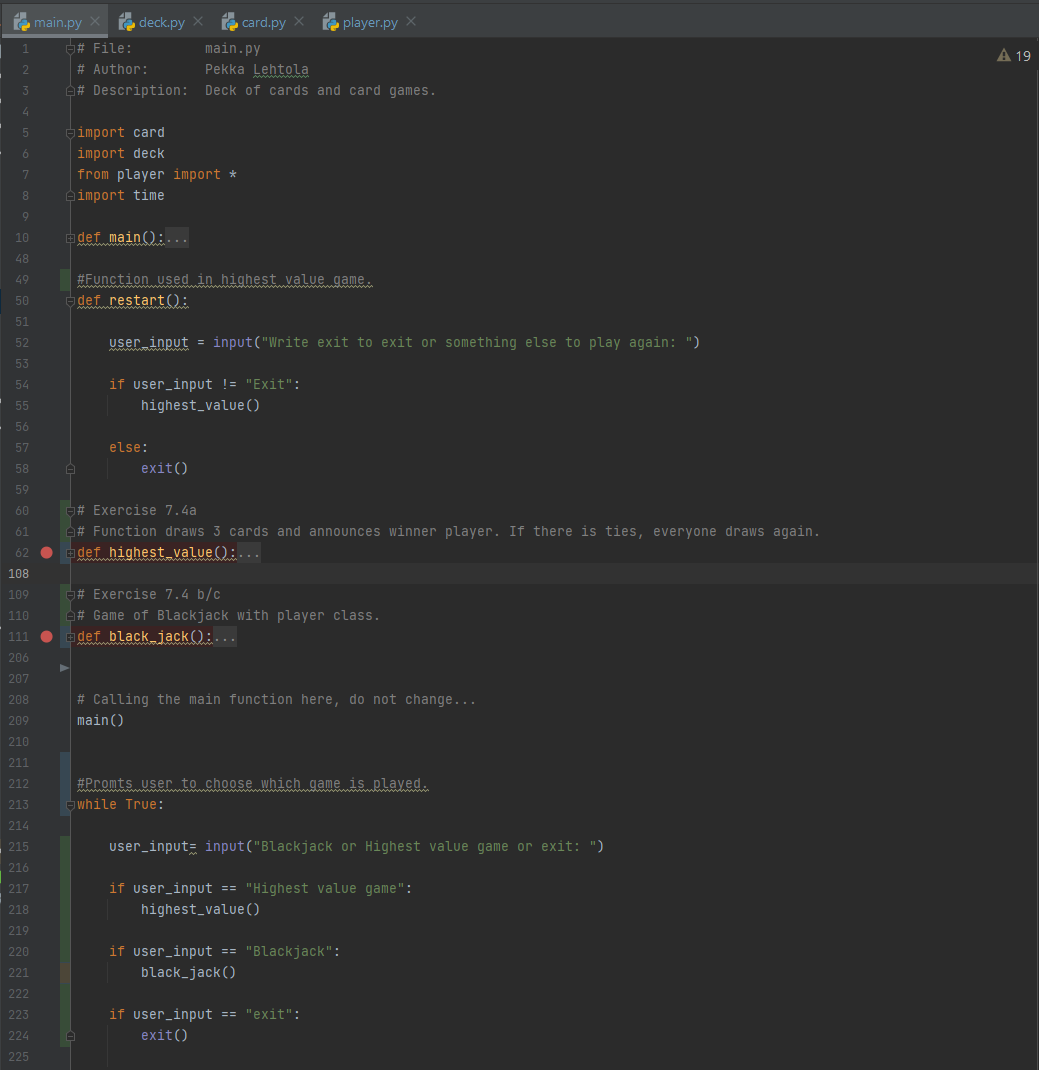
Card class:

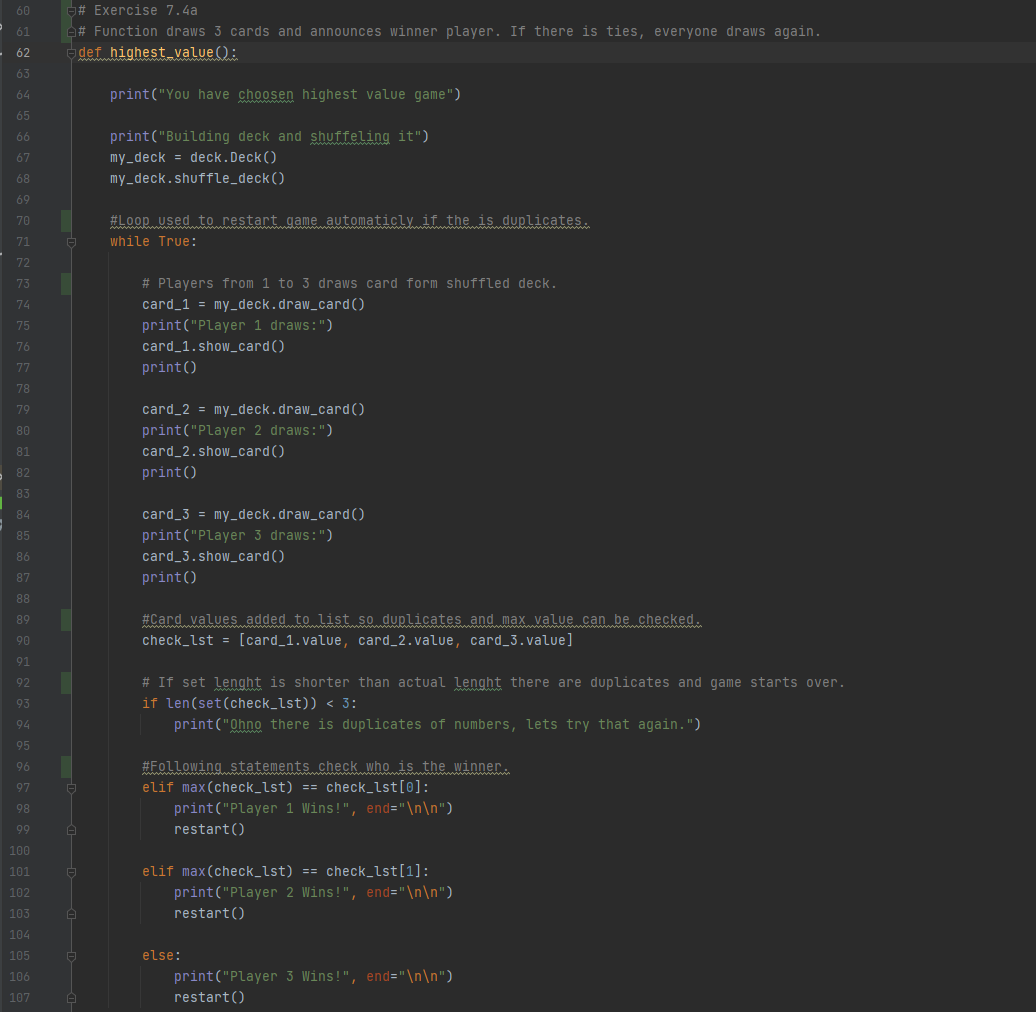


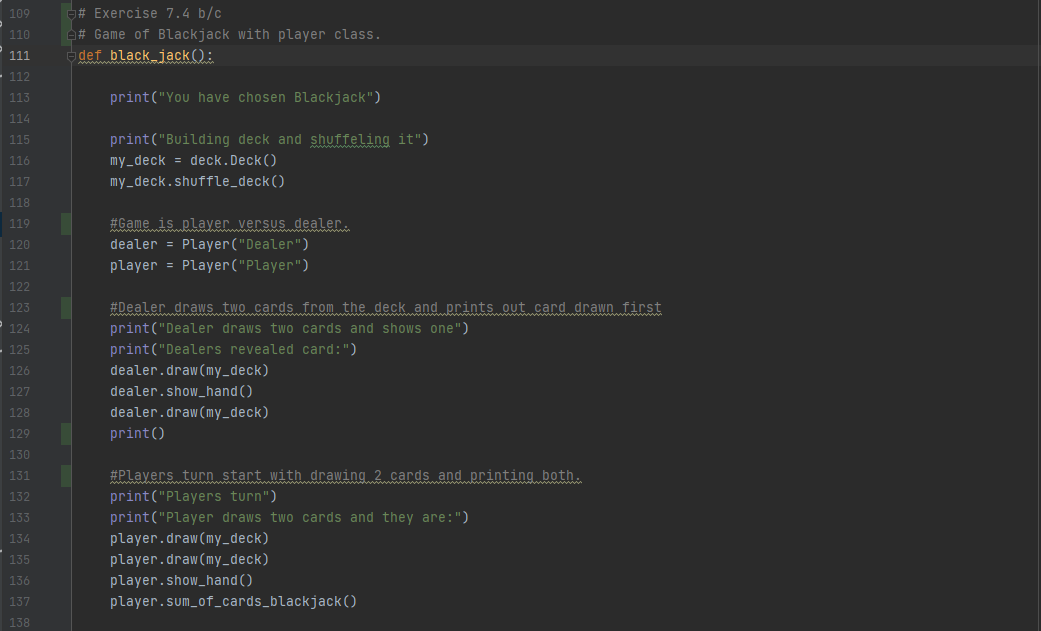
Player class:

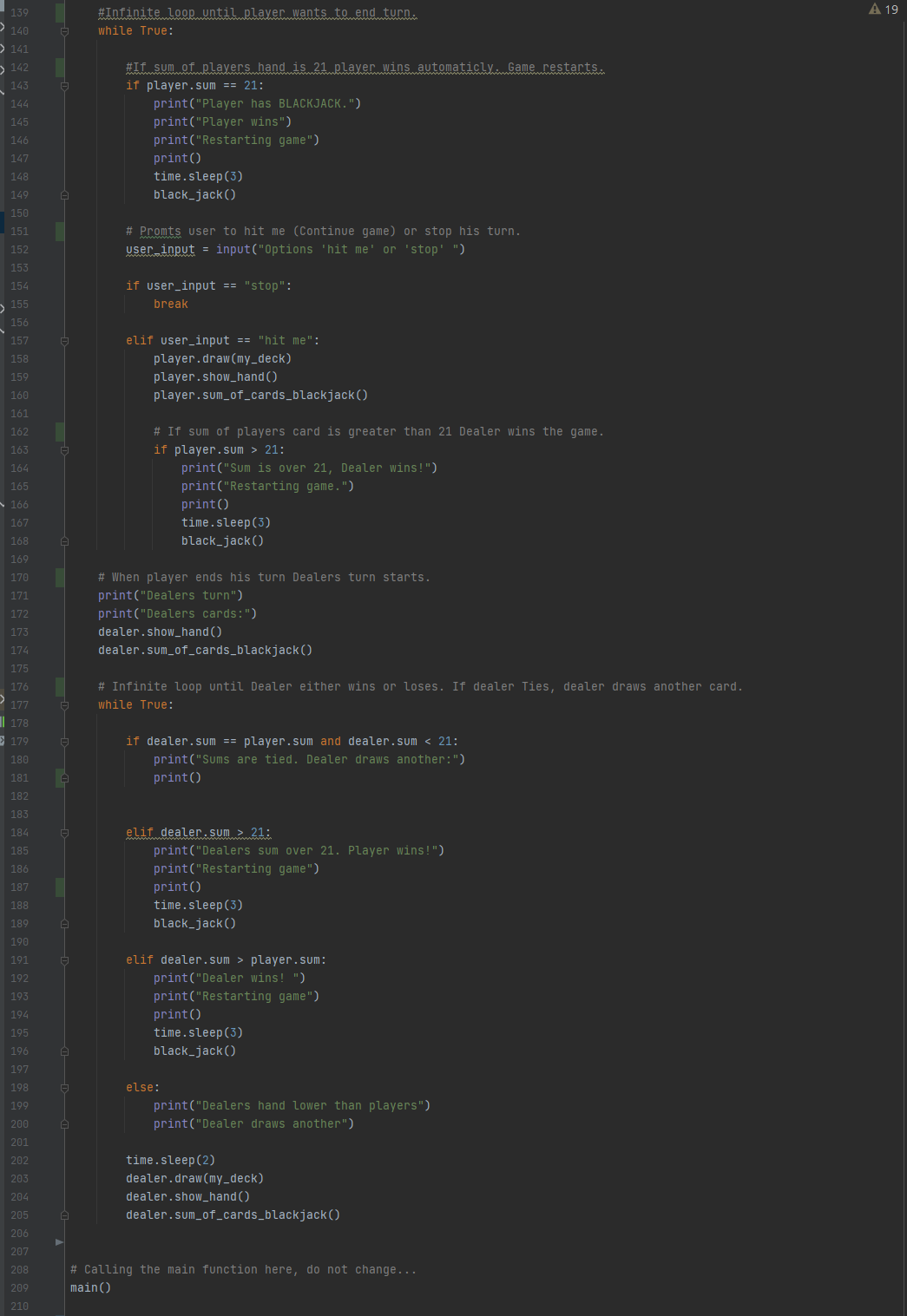


Main without game functions:

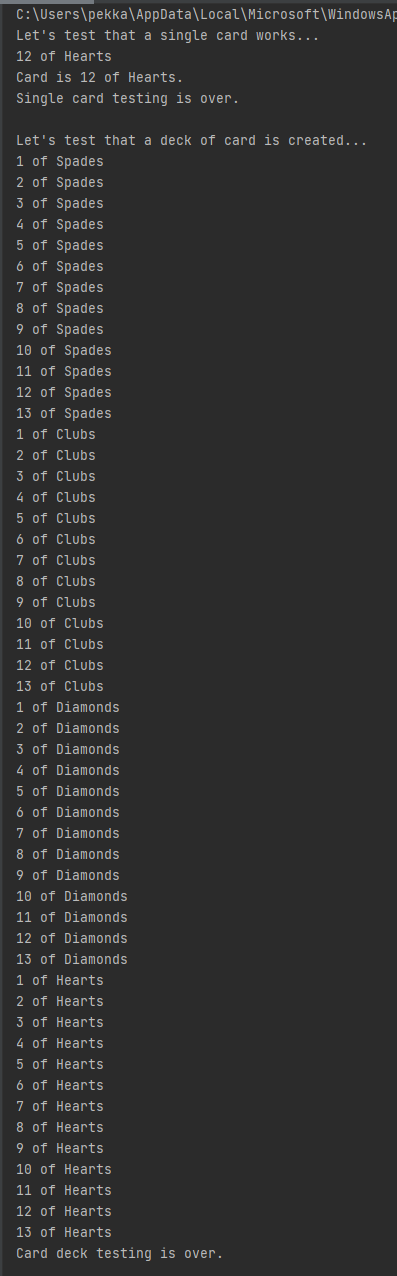
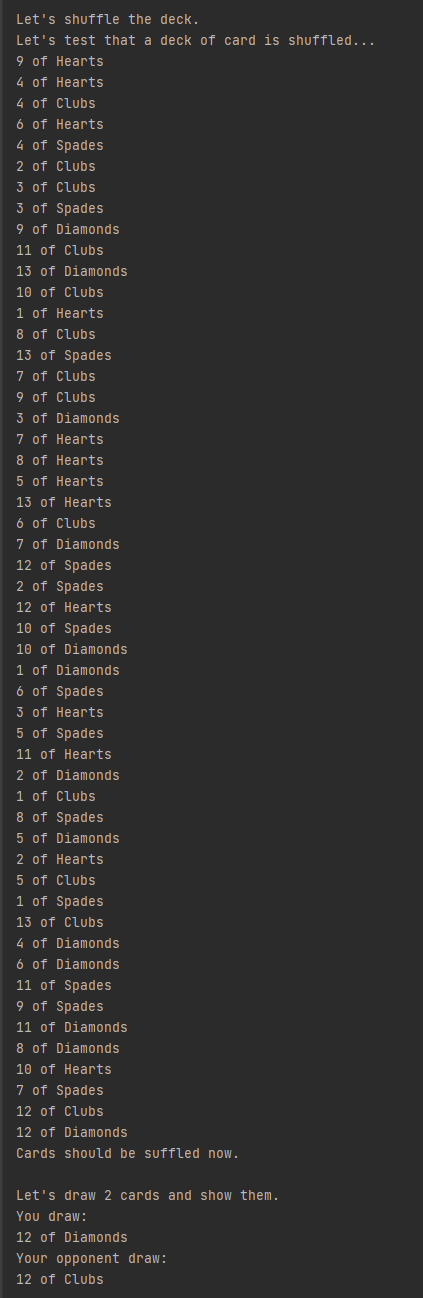


Highest value game:

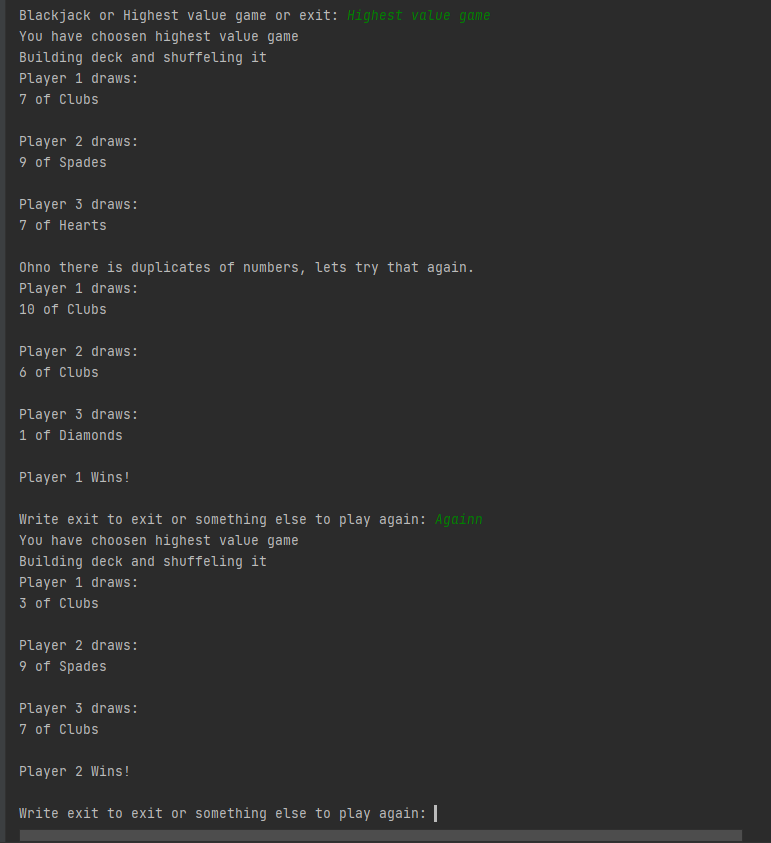
Blackjack :



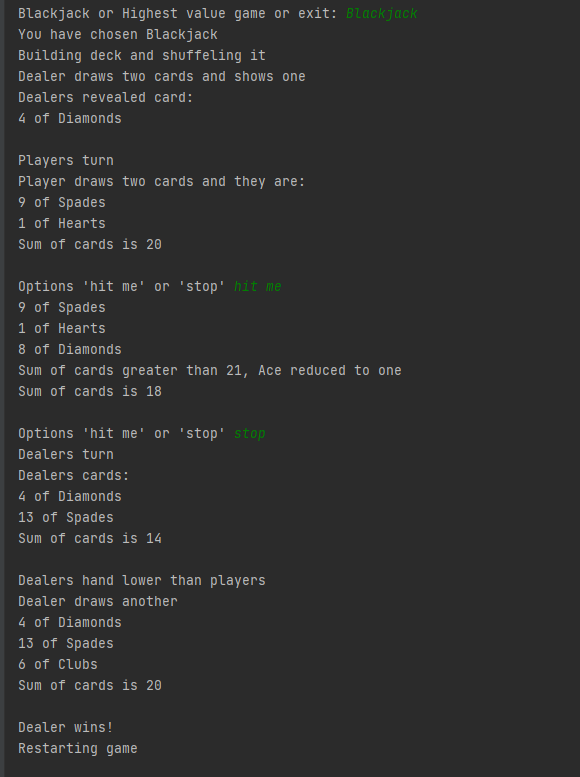
Screen capture of the output of Task 4

Original output 1: Original output 2:

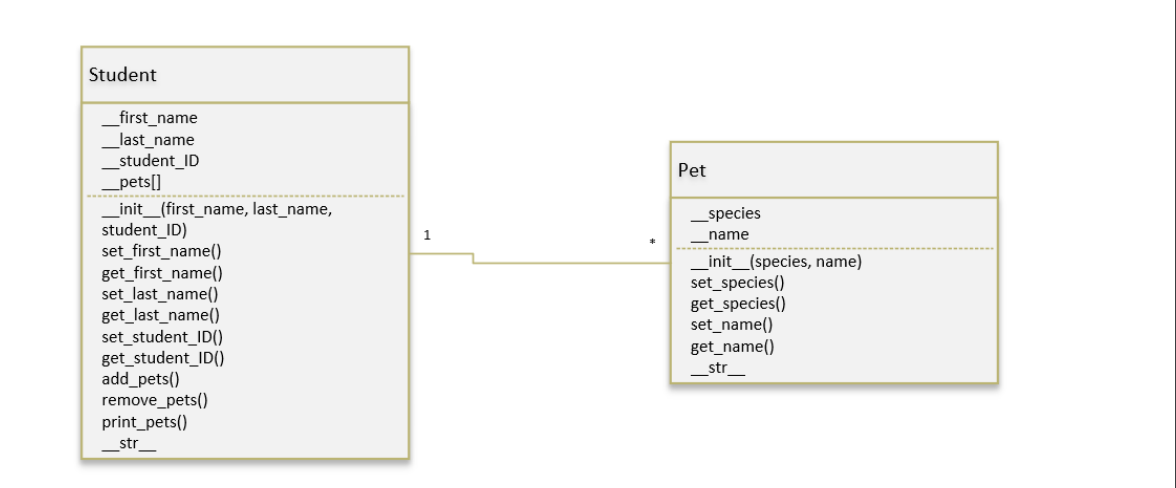
Highest value game output:



Blackjack output:



5. Implement the following UML diagram. Try to figure out the best way to have animals in appropriate data structure in the Student class (see the link in Task 4 above, there is an example of Deck class creating a deck of cards, pay attention to the relationship between the Deck class and Card class and think how that information can be applied in the relationship between the Student class and Animal class). Pet Class can also be called Animal (you most likely have implemented that in previous exercises). Think (carefully), do you need to have the owner of the pet information in the Pet/Animal class (in order to make the relationship between Student and Pet) If yes, add that.

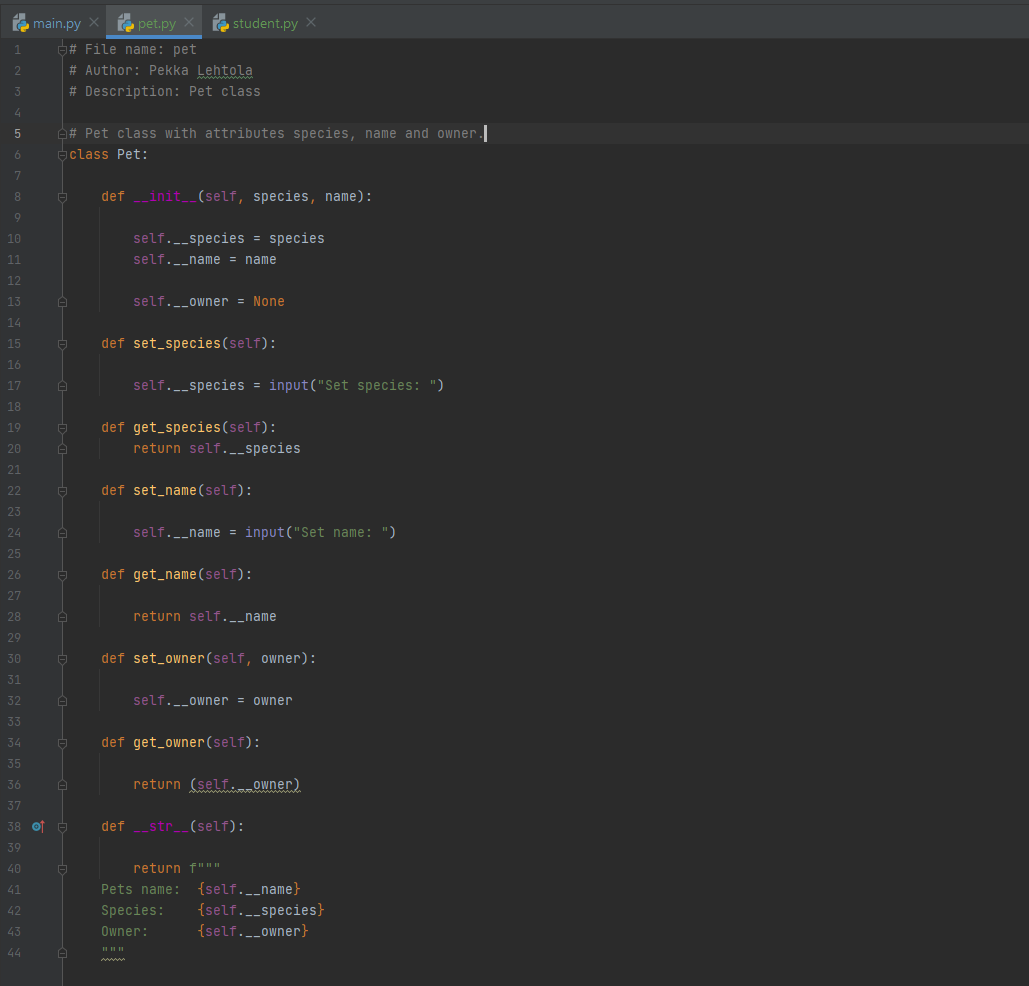


Screen capture of Task 5

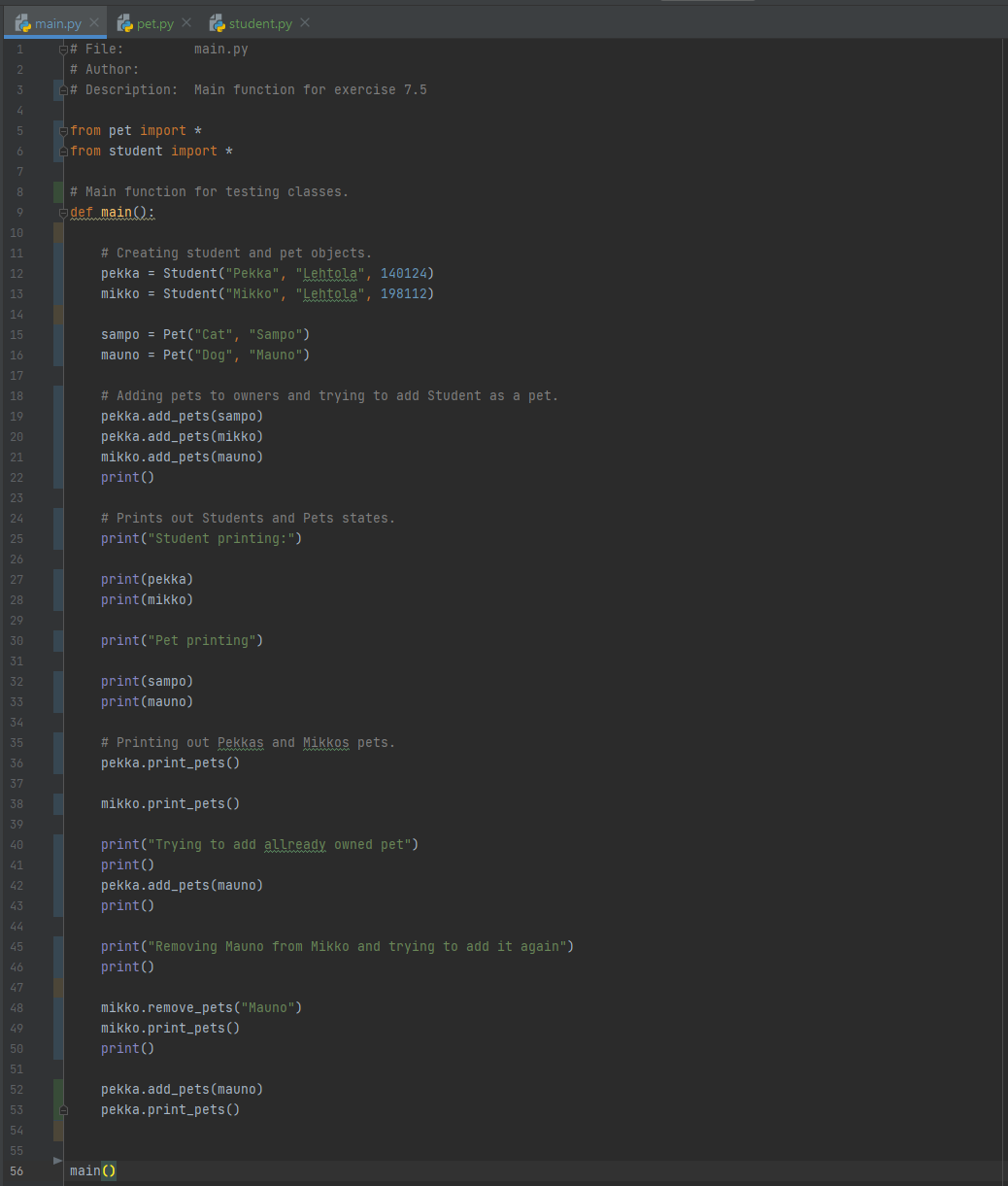
Student:



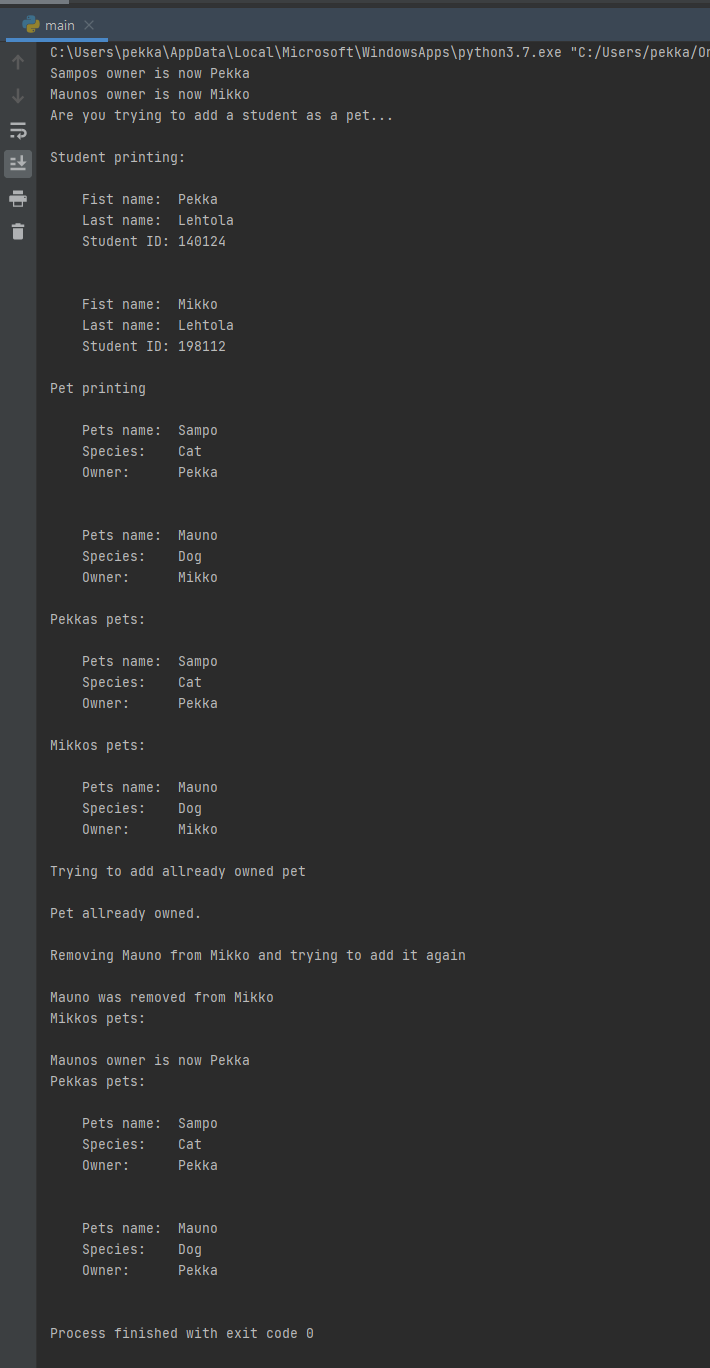
Pet:



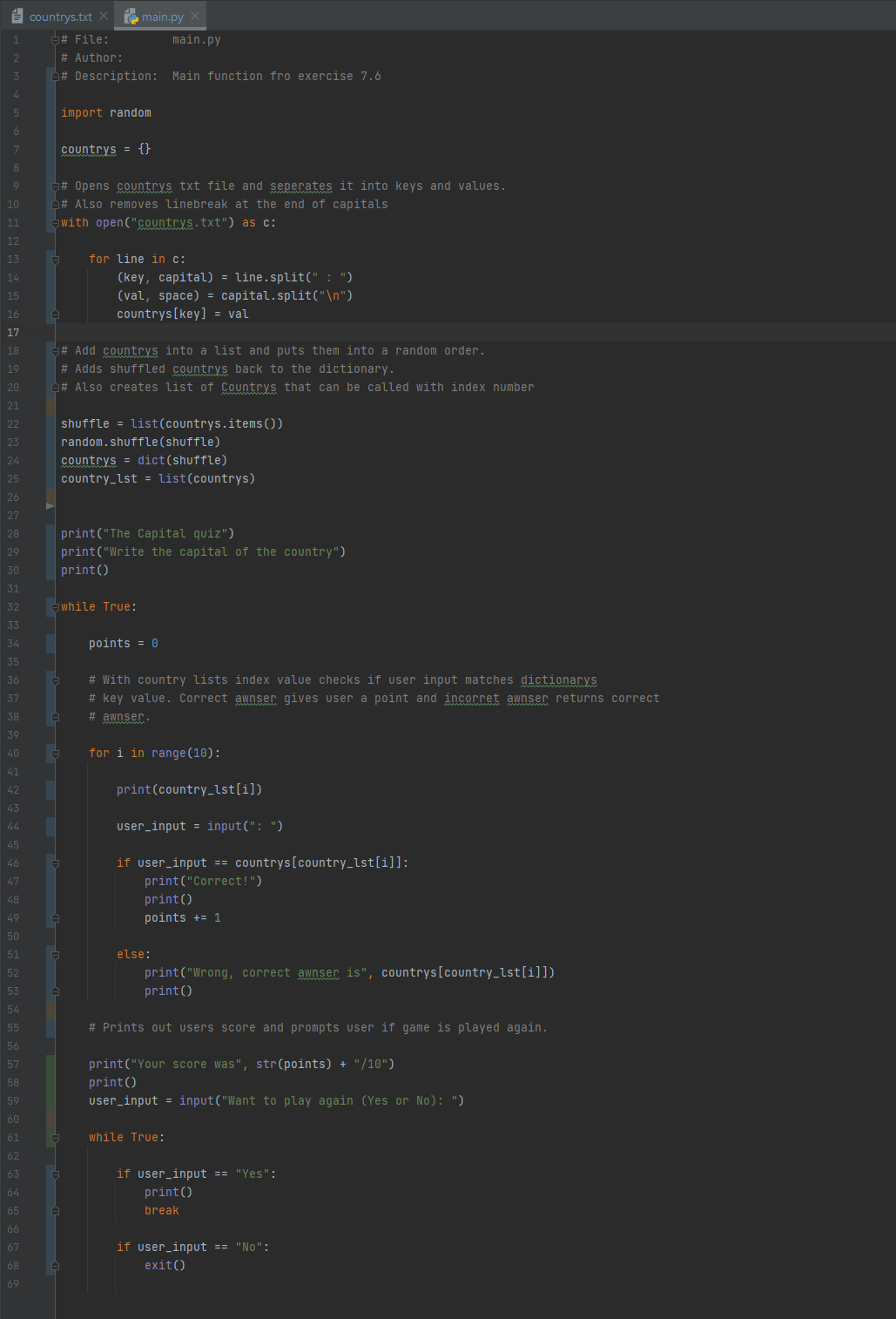
Main:

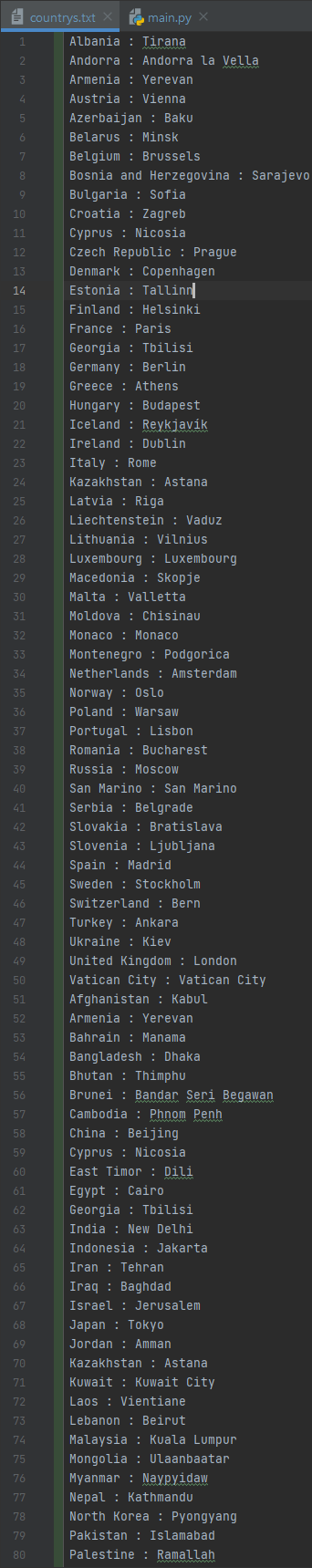


Screen capture of the output of Task 5

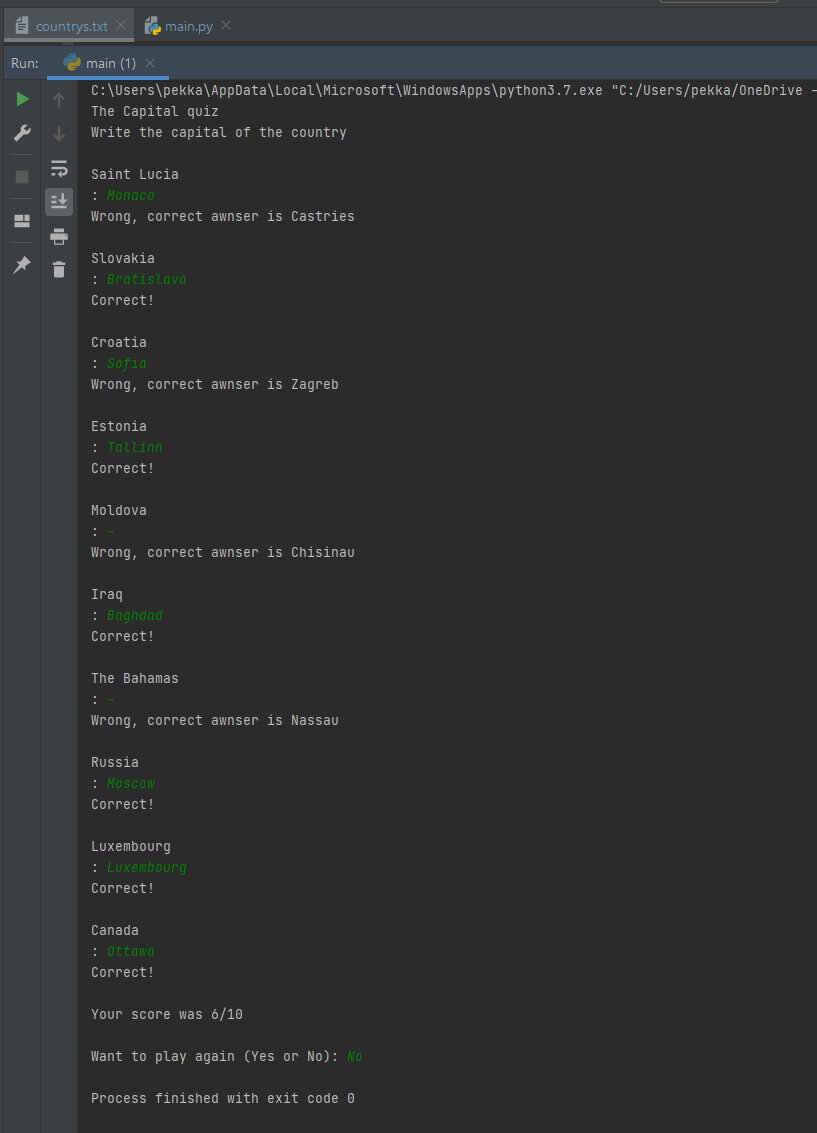


6. Still practicing the use of dictionary. Implement a simple quiz where the user is asked the capitals of countries. First, make a text file with at least 50 countries with their capitals. The information is read at the beginning of the program into the dictionary. In the quiz itself, the user is asked the capitals of ten countries. When correctly answered by the user, proceed to the next question. If the user answers incorrectly, they will be shown the correct answer and then proceed to the next question. After ten questions, the user is informed of the number of correct answers.

Screen capture of Task 6 Sample of countrys.txt file



Screen capture of the output of Task 6



Screen capture of git log (showing that you made a commit after every task).



Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Tehtävät olivat mukavia ja sopivan haastavia. Oli ilo huomata että on oppinut kurssin aikana jotain, koska tehtävä 5 sujui kuin rasvattu ja UML diagrammin lukeminen ei tuottanut minkäänlaisia ongelmia.

Doing this exercise, I learned…

Oppisin taas paremmin käyttämään dictionareja ja lisäksi en ole koskaan koodauksessa lukenut tiedostosta tekstiä.

I am still wondering…

-

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

Onnistuin kaiken tekemään tällä kertaa aika ongelmitta. Blackjackin kanssa oli vähän tahmeutta. Tehtävä 4c jäi tekemättä vain koska aika loppui kesken…