**Exercise: 5**

**Name: Pekka Lehtola**

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

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

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

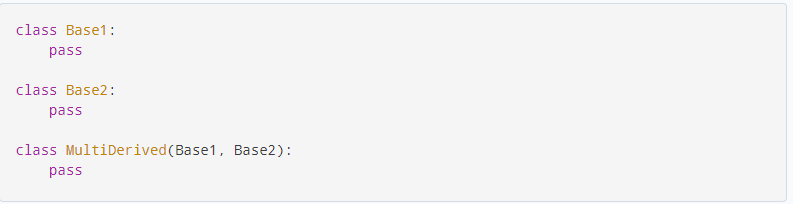
**Tehtävästä 6 voisi olla tunnilla esimerkki.**

Explain the following term and what is it used for:

1. Multiple inheritance

A class can be derived from more than one base class in Python, similar to C++. This is called multiple inheritance. In multiple inheritance, the features of all the base classes are inherited into the derived class. The syntax for multiple inheritance is similar to single inheritance.

Example:



2. True or false?

a. The practice of procedural programming is centered on the creation of objects.

False. The focus of procedural programming is to break down a programming task into a collection of variables, data structures, and subroutines, whereas in object-oriented programming it is to break down a programming task into objects that expose behavior (methods) and data (members or attributes) using interfaces.

1. Object reusability has been a factor in the increased use of object-oriented programming.

True. OOP was developed to increase the reusability and maintainability of source code. This is the reason its used in bigger projects.

1. It is a common practice in object-oriented programming to make all of a class’s data attributes accessible to statements outside the class.

False. Most of the time Setters and Getters are used to modify data attributes, for security reasons.

1. A class methods does not have to have a self parameter.

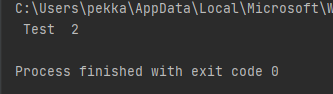
False/True Self can be any other word, but usually “self”, but that word is mandatory.

1. Starting an attribute name with two underscores will hide the attribute from code outside the class.

True. Double underscore prefix prevents access to attribute, except through accessors.

1. You cannot directly call the \_\_str\_\_method.

True



3. Multiple choice:

a. The \_\_\_\_\_ method is automatically called when an object is created.

i. \_\_init\_\_

ii. init

iii. \_\_str\_\_

iv. \_\_object\_\_

b. The \_\_\_\_\_ programming practice is centered on creating functions that are

separated from the data that they work on.

i. modular

ii. procedural

iii. functional

iv. object-oriented

c. The \_\_\_\_\_ programming practice is centered on creating objects.

i. object-centric

ii. objective

iii. procedural

iv. object-oriented

d. A(n) \_\_\_\_\_ is a component of a class that references data

i. method

ii. instance

iii. data attribute

iv. module

e. By doing this, you can hide a class’s attribute from code outside the class.

i. avoiding using the self-parameter to create the attribute

ii. begin the attribute’s name with private\_\_

iii. begin the name of the attribute with two underscores

iv. begin the name of the attribute with the symbol #

f. A(n) \_\_\_\_\_ method stores a value in the data attribute or changes its value in

some other way.

i. modifier

ii. constructor

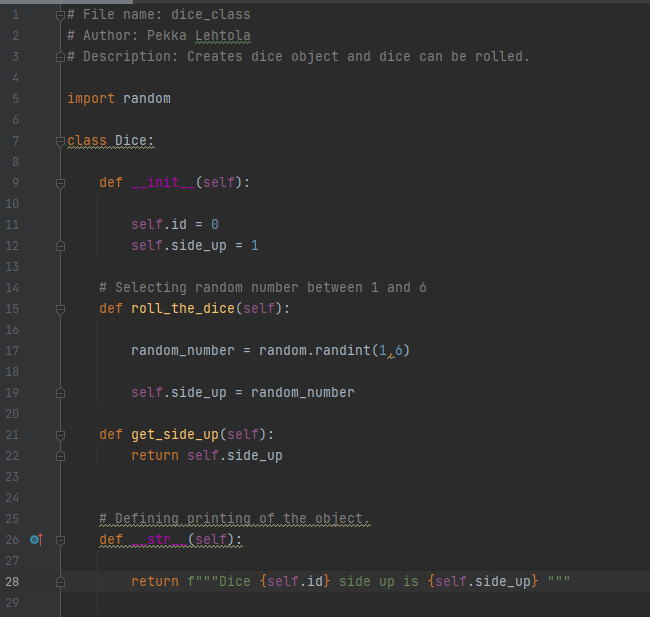
iii. mutator

iv. Accessor

4. Create multiple dices (at least three) and put them in a list. See that your dice can be rolled and the side can be shown. Create a small game where the best sum of three rolls wins. If the sum is a tie, tied dices are rolled as long as a winner is found (best side wins). Use functions and pass objects (or list of objects) as arguments. Use informative and clear output prints.

Screen capture of Task 4

Dice class:

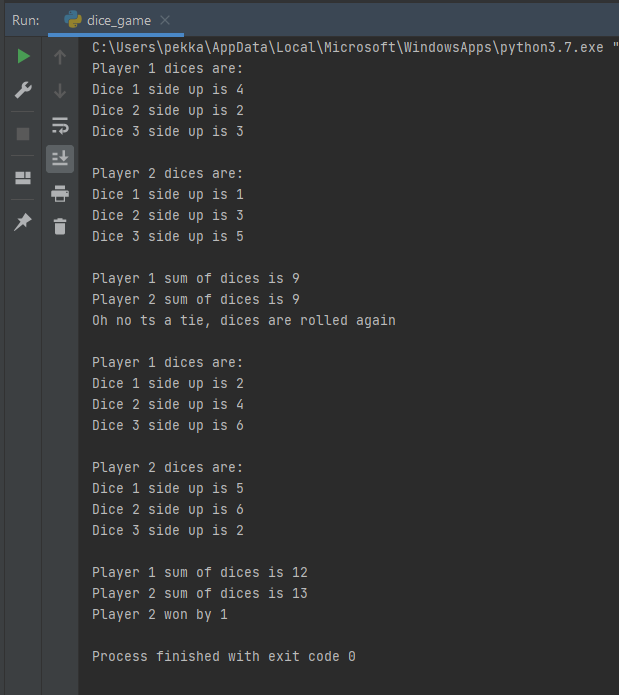


Main game:



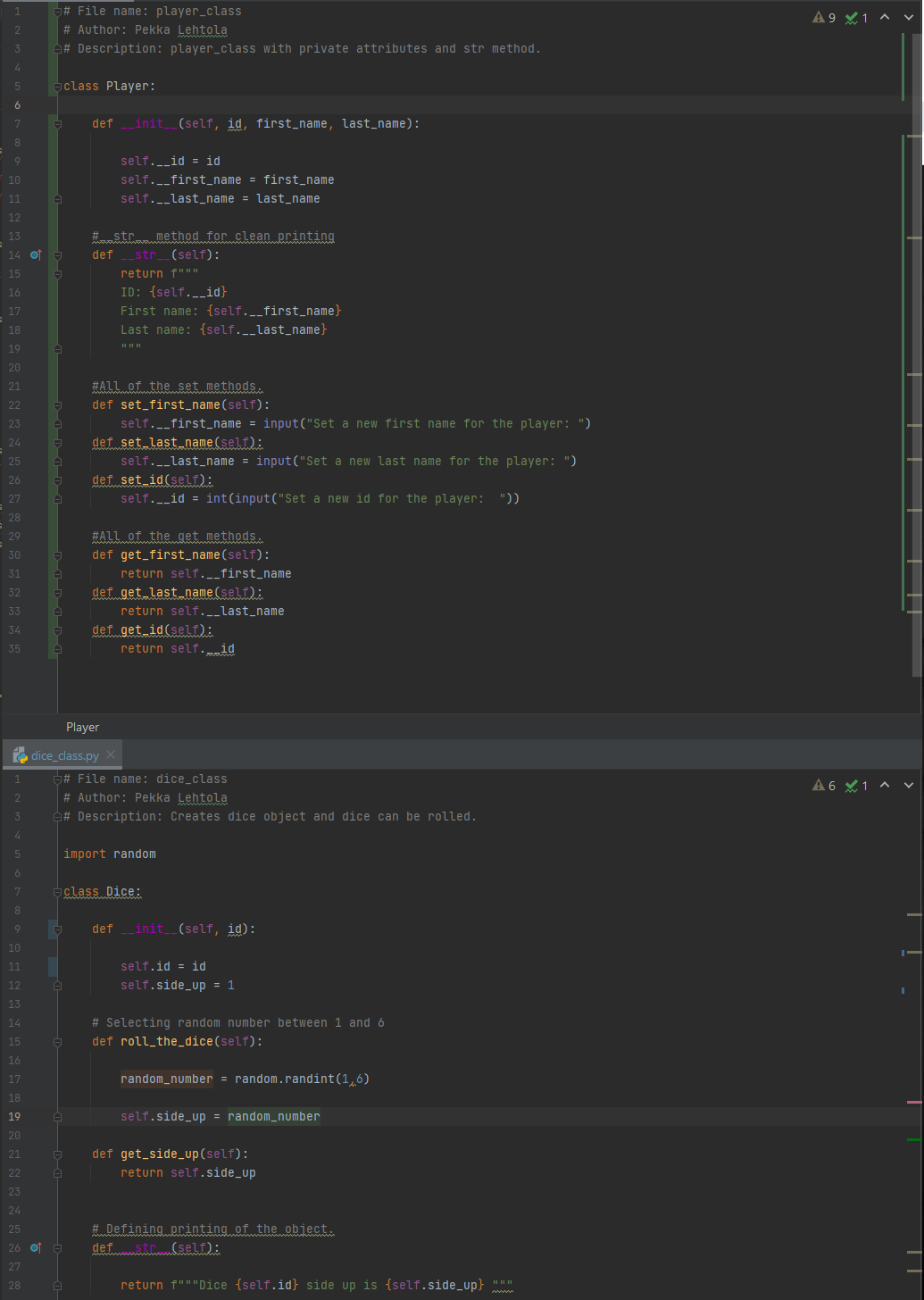


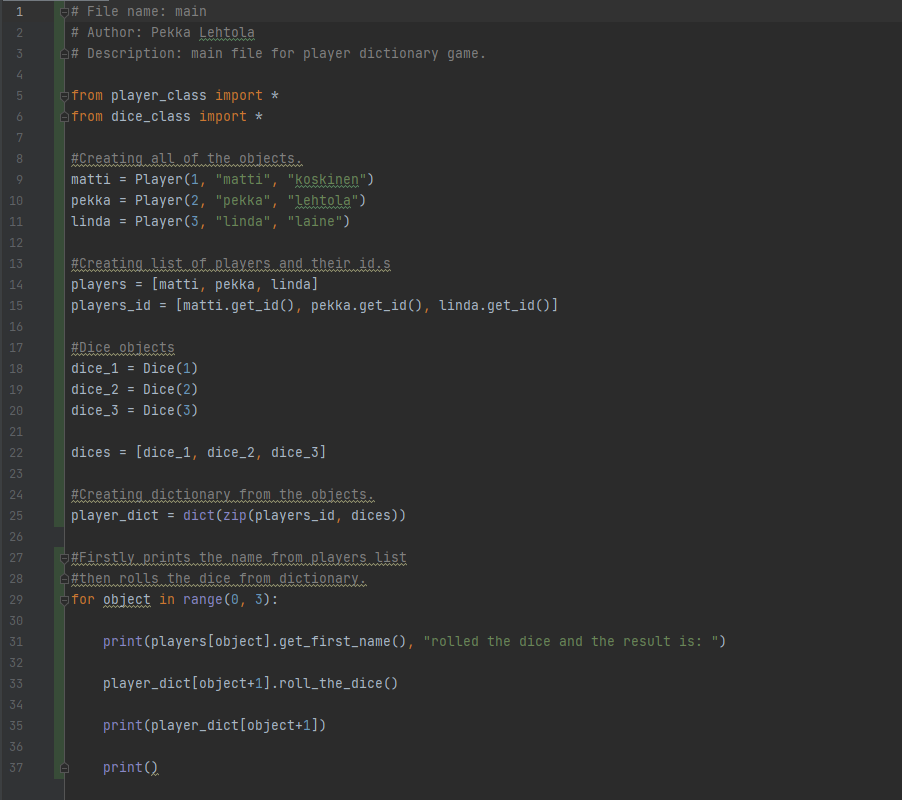
Screen capture of the output of Task 4



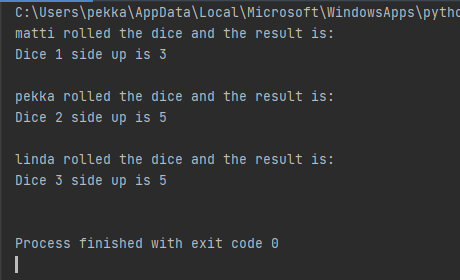
5. Create a class called Player. Player has at least the following data attributes: first name, last name and a player id. Remember to code accessor and mutator methods and strmethod. Create a dictionary so that the player id is a key and each player has one dice. Roll each player’s dice and print out each player’s dice’s side. Use informative and clear output prints.

Screen capture of Task 5



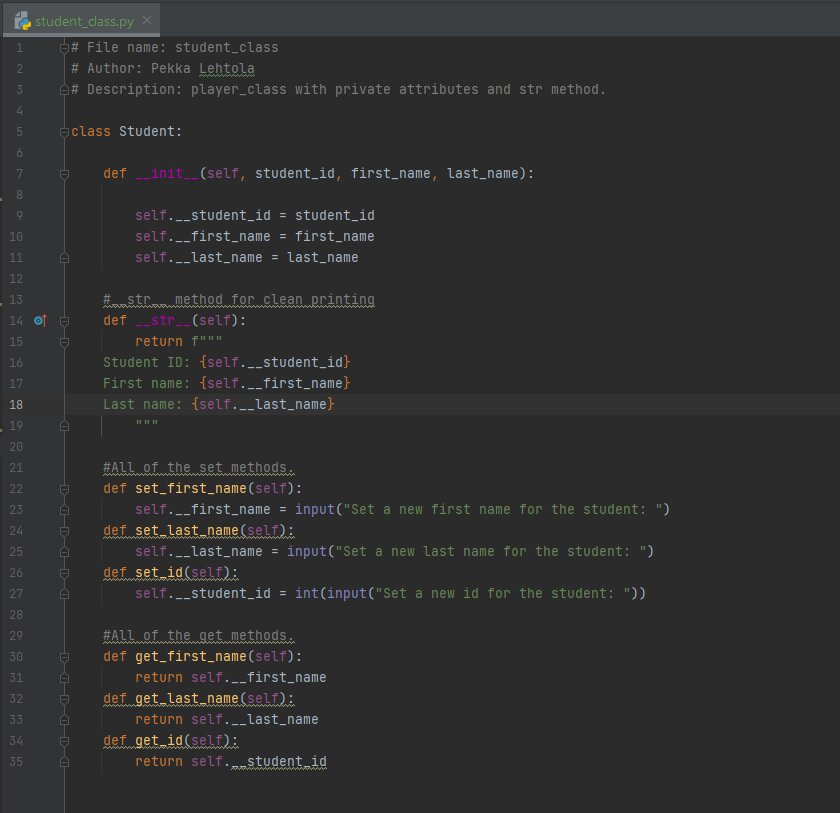
Main:

Screen capture of the output of Task 5

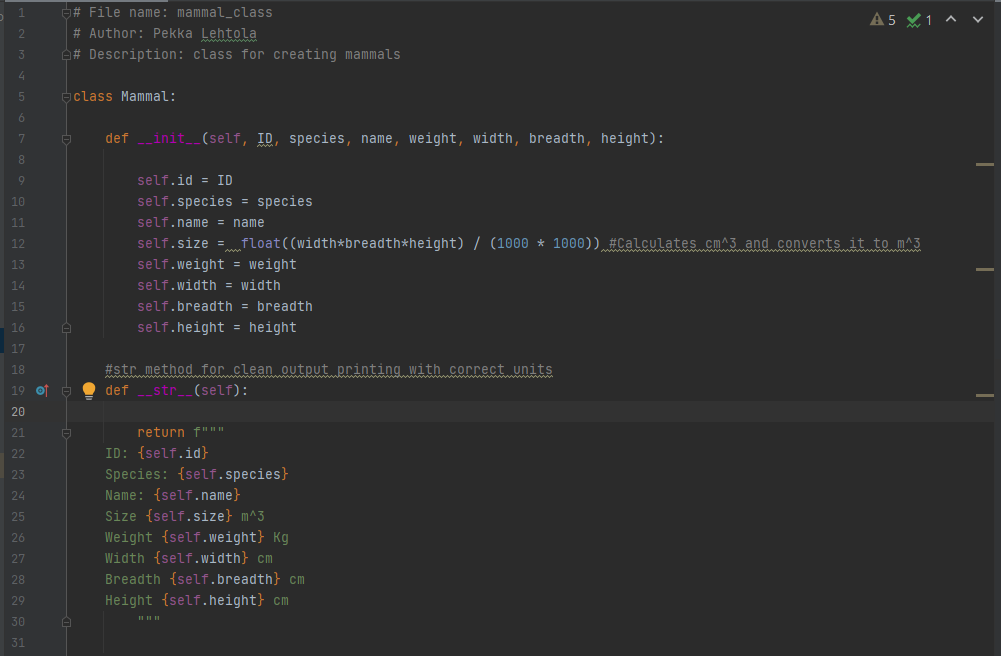


6. Create a class called Student and use the following data attributes: first name, last name and student id. Remember to code accessor and mutator methods and str-method. Store students and their pet mammal to dictionary (use the mammals from Exercise 4). Think, what should be used as the dictionary key. Code a function that prints out each student and their mammal’s information. Use informative and clear output print.

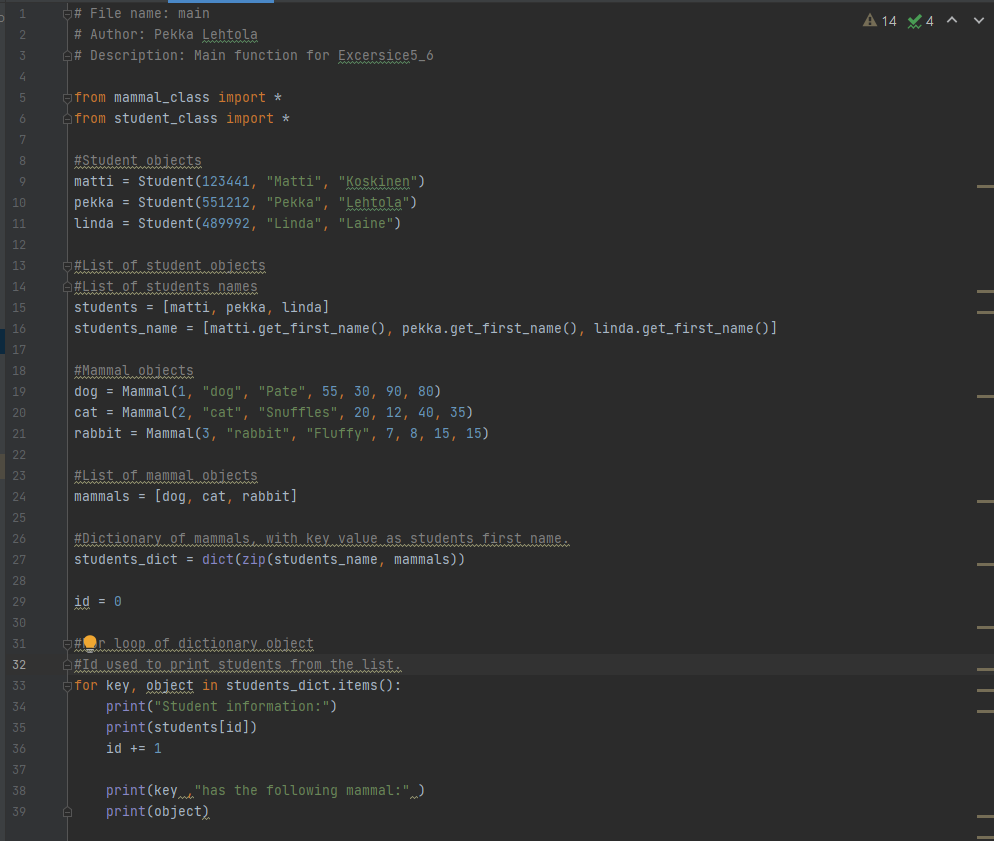
Screen capture of Task 6

Student class:

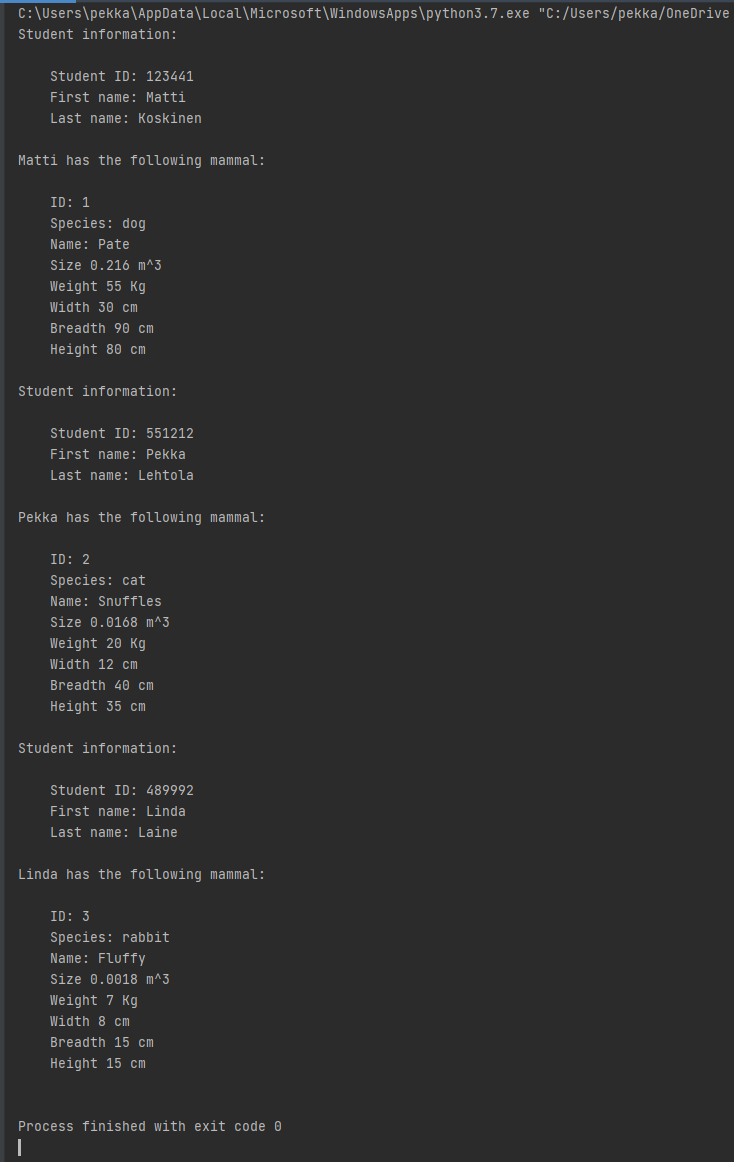
Mammal class:

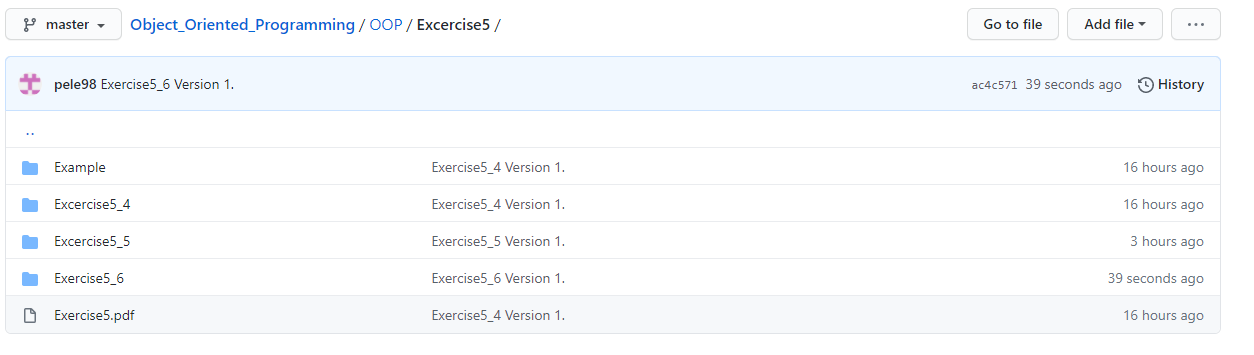


Main:



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 tuntuivat haastavilta. Sain dictionary objectit hädin tuskin toimimaan.

Doing this exercise, I learned…

Alkeellisen tavan käsitellä objecteja dictionary muodossa.

I am still wondering…

Voiko dictionaryn avain olla myös jotenkin objecti?

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

Omasta mielestä koodi ei ollut tällä kertaa onnistunutta. Yritin katsoa esimerkkiä, joka löytyi Itsistä, mutten saanut mitään toimimaan.