Wrocław University of Science and Technology



Faculty of Electronics, Photonics and Microsystems

Theme of class: Classes

Student: Hayrettin Aycetin (276807)

Date of class: 20.11.2023 15:15-16:55

Group No:3

Submition Date:1.12.2023

Lab assistant: Aleksander Kubeczek, Alicja Kwaśny GRADE:

Task 1

Write a Python program to create a person class. Include attributes like name, country and date

A screen shot of a computer program

Description automatically generatedof birth. Implement a method to calculate the person's age.

A black screen with white text

Description automatically generated

Comments:

Firstly, I defined a class named "Person" in Python. Using the class constructor \_\_init\_\_, I took parameters such as name, country, and date of birth. Subsequently, I assigned these parameters to corresponding attributes – name, country, and date of birth – using self for better readability.

Next, I implemented a method called "age" within the class. This method calculates the age of the person by utilizing the datetime library.

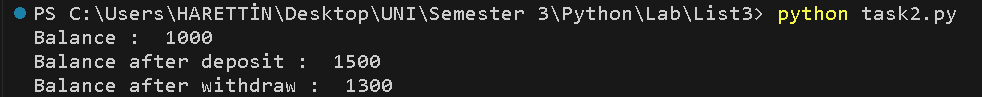
Finally, I instantiated four objects of the "Person" class and calculated their ages.

Task 2

Write a Python class Bank Account with attributes like account \_ number, balance,   
date\_ of\_ opening and customer \_ name, and methods like deposit, withdraw, and check \_ balance.

A screen shot of a computer program

Description automatically generated



Comments:

created a basic class named BankAccount in Python. In the class constructor (\_\_init\_\_), I defined parameters such as account\_number, balance, date\_of\_opening, and customer\_name. These parameters were then assigned to their corresponding attributes.

Within the class, I implemented three methods:

check\_balance: This method returns the current balance of the account.

deposit: To deposit money, I added a parameter named amount to specify the amount to be added to the account.

withdraw: For withdrawing money, I included the amount parameter to indicate the amount to be subtracted from the account.

To test the class, I instantiated an object called account\_1. I performed the following operations:

Checked the initial balance.

Deposited $500 into the account.

Checked the updated balance after the deposit.

Withdrew $200 from the account.

Checked the final balance after the withdrawal.

Task 3

Write a Python class Employee with attributes like emp\_id, emp\_name, emp\_salary, and

emp\_department and methods like calculate\_emp\_salary, emp\_assign\_department, and

print\_employee\_details.

Sample Employee Data:

"ADAMS", "E7876", 50000, "ACCOUNTING"

"JONES", "E7499", 45000, "RESEARCH"

"MARTIN", "E7900", 50000, "SALES"

"SMITH", "E7698", 55000, "OPERATIONS"

• Use 'assign\_department' method to change the department of an employee.

• Use 'print\_employee\_details' method to print the details of an employee.

• Use 'calculate\_emp\_salary' method takes two arguments: salary and hours\_worked,

which is the number of hours worked by the employee. If the number of hours worked

is more than 50, the method computes overtime and adds it to the salary. Overtime is

calculated as following formula:

overtime = hours\_worked - 50

overtime amount = (overtime \* (salary / 50))

A computer screen shot of a program

Description automatically generated

A screen shot of a computer code

Description automatically generated

A black screen with white text

Description automatically generated

Comments:

Firstly, I created a class called Employee in Python. In the class constructor (\_\_init\_\_), I defined parameters such as emp\_id, emp\_name, emp\_salary, and emp\_department. These parameters were assigned to their respective attributes within the class.

I implemented the following methods:

assign\_department: This method takes a parameter new\_department and updates the employee's department.

overtime\_salary\_calculator: With this method, I calculated overtime salary based on the hours\_worked and the employee's current salary. If the employee worked more than 50 hours, the method applies an overtime amount formula and adds it to the salary.

display: The display method prints the employee's data using the print function and basic self notation.

To test the class functionality, I instantiated four Employee objects. I demonstrated the following operations:

Changed the department of emp\_1 using the assign\_department method.

Displayed the updated information for emp\_1.

Checked the overtime salary for emp\_1 by assuming they worked 60 hours.

Task 4

Write a Python program to create a class that represents a shape. Include methods to calculate

its area and perimeter. Implement subclasses for different shapes like circle, triangle, and

square.

A screen shot of a computer code

Description automatically generatedA screen shot of a computer program

Description automatically generated

A black screen with white text

Description automatically generated

Comments:

I began by defining a base class called Shape. Using the \_\_init\_\_ constructor, I took perimeter and area as parameters and assigned them to their respective attributes within the class.

Next, I created three subclasses – Circle, Square, and Triangle – each inheriting from the Shape class. For each subclass, I implemented methods to calculate their specific area and perimeter.

For the Circle class, the area and perimeter calculation methods were tailored to the properties of circles. Similarly, for the Square and Triangle classes, methods were defined to compute the area and perimeter based on their respective shapes.

To demonstrate the functionality, I instantiated one object for each subclass. Subsequently, I calculated and obtained the area and perimeter for each shape.

Task 5

Write a program that generates a set of samples of a sine wave. The parameters are as follows:

Frequency = last 2 digits of your index number [Hz] (if it is ‘00’ then use ‘01’), Sampling

Frequency = 48 [kHz], Acquisition time = 2 [s], Amplitude = 2. Hint – use numpy and

matplotlib libraries. It should be realized as a class which stores the sine wave, has a method to

generate sine wave samples with chosen parameters, plotting the sine wave as well as returning

samples of down sampled wave. Plot both sets on one figure for comparison.  
A screen shot of a computer program

Description automatically generated

A red and blue lines

Description automatically generated

Comments:

To begin, I imported the necessary libraries and defined a class named SineWave. Using the \_\_init\_\_ constructor, I took parameters such as frequency, amplitude, acquisition\_time, and sampling\_frequency, assigning them to corresponding attributes of the class.

I implemented a method called generate, which serves as the sine wave generator. Utilizing the time formula obtained from online sources and the np.arange() function, I defined the time for the wave. The method then returns the complete formula for generating a sine wave.

Additionally, I created a method named downsample with a parameter downsample\_factor and a constant value of 2. This factor determines how much downsampling is applied manually. The downsampled frequency is calculated as the sampling frequency divided by the downsample factor. Similar to the generator method, the time formula is adjusted, replacing the sampling frequency with the downsampled frequency. The return statements, however, remain the same as in the generator method.

Subsequently, I instantiated an object called my\_sine\_wave, specifying its parameters. I then applied the downsample function to this object, creating a second object called my\_sine\_wave\_downsampled.

Finally, I designed a plot using the matplotlib.pyplot library and its attributes.

Task 6

Write a program with the same functionality as in List 1 task 6. It should be realized as a ‘Player’

class, where each of the player is a separate object. Inside it stores the score of the player and

has method for calculating the score of a given word. Make a simple text UI to enhance the

program's accessibility for players.

A screen shot of a computer program

Description automatically generated  
A computer screen shot of a program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Comments:

Firstly, I defined a class called Player and constructed it by taking a name parameter, which I assigned to the name attribute. I also set a constant score of 0 as an attribute. Later, I created a method called input\_words\_and\_multipliers, similar to my previous Scrabble game. In this version, I assigned the word, triple\_word, and double\_word as attributes to this class because I plan to use them later.

My second method, calculate\_score, computes the score of the word without considering multipliers and returns it as a score. The third method, calculate\_final\_score, is similar to the function in my previous Scrabble game, with the only difference being that I assign without\_score\_words to self.score. So, we can say methods are functions of classes, basically.

My fourth method is to display the final score.

In the main function, I created two players, named them, and stored them in a list. I initialized current\_player to zero to start from player one. Later, I implemented a while True statement, which finishes when we choose option 3. This is controlled by a menu using choice and if statements."Lastly I checked with if \_\_name\_\_ == "\_\_main\_\_" if the script is being run as the main program.

Conclusion

This list was very helpful to learn main techniques and usage of OOP in Python which is very important I think and related to real life examples.I think first four tasks were quite similar but they gave me a solid knowledge and synthax knowledge of classes in Python.