assignment

February 21, 2024

1 LAB 5 Assignment

1.0.1 1. WAP in python that uses class to store the name and marks of the students. Use list to store the marks in three subjects.

[ ]:
```python
class Student:
    def __init__(self, name):
        self.name = name
        self.marks = []

    def add_marks(self, subject, marks):
        self.marks.append((subject, marks))

    def display_marks(self):
        print(f"Student: {self.name}")
        for subject, marks in self.marks:
            print(f"{subject}: {marks}")

# Create a list to store student objects
students = []

# Get the number of students from user
num_students = int(input("Enter the number of students: "))

# Get the details of each student from user
for i in range(num_students):
    name = input(f"Enter the name of student {i+1}: ")
    student = Student(name)

    # Get the marks for three subjects from user
    for subject in ['Subject 1', 'Subject 2', 'Subject 3']:
        marks = float(input(f"Enter the marks for {subject}: "))
        student.add_marks(subject, marks)

    students.append(student)

# Display the marks of all students
for student in students:
    student.display_marks()
```

```
Student: Harshil
Subject 1: 100.0
Subject 2: 99.0
Subject 3: 93.0
```

1.0.2 2) WAP in python with class Employee that keeps a track of the number of employees in an organization and also stores their name, designation and salary details.

[ ]:
```
class Employee:
    num_employees = 0

    def __init__(self, name, designation, salary):
        self.name = name
        self.designation = designation
        self.salary = salary
        Employee.num_employees += 1

    def display_details(self):
        print(f"Employee Name: {self.name}")
        print(f"Designation: {self.designation}")
        print(f"Salary: {self.salary}")

# Create a list to store employee objects
employees = []

# Get the number of employees from user
num_employees = int(input("Enter the number of employees: "))

# Get the details of each employee from user
for i in range(num_employees):
    name = input(f"Enter the name of employee {i+1}: ")
    designation = input(f"Enter the designation of employee {i+1}: ")
    salary = float(input(f"Enter the salary of employee {i+1}: "))

    employee = Employee(name, designation, salary)
    employees.append(employee)

# Display the details of all employees
for employee in employees:
    employee.display_details()

# Display the total number of employees
print(f"Total number of employees: {Employee.num_employees}")
```

```
Employee Name: Harshil
```

```
Designation: Growth Lead
Salary: 100000.0
Employee Name: Eshita
Designation: CEO
Salary: 200000.0
Total number of employees: 2
```

1.0.3 3) WAP in python that has a class Person storing name and date of birth of a person. The program should subtract the date of birth from today's date to find out that the person is eligible to vote or not.

[ ]:
```python
import datetime

class Person:
    def __init__(self, name, dob):
        self.name = name
        self.dob = dob

    def calculate_age(self):
        today = datetime.date.today()
        age = today.year - self.dob.year - ((today.month, today.day) < (self.
 ↪dob.month, self.dob.day))
        return age

    def check_voting_eligibility(self):
        age = self.calculate_age()
        if age >= 18:
            return f"{self.name} is eligible to vote."
        else:
            return f"{self.name} is not eligible to vote."

# Get the name and date of birth from user
name = input("Enter the name: ")
dob_str = input("Enter the date of birth (YYYY-MM-DD): ")
dob = datetime.datetime.strptime(dob_str, "%Y-%m-%d").date()

# Create a person object
person = Person(name, dob)

# Check voting eligibility and print the result
result = person.check_voting_eligibility()
print(result)
```

```
Harshil is eligible to vote.
```

1.0.4 4)WAP in python that has a class Circle. Use a class variable that defines the value of constant Pie value=3.14. Use this class variable to calculate the area and circumference of the circle with specified radius

```
[ ]: class Circle:
         PI = 3.14

         def __init__(self, radius):
             self.radius = radius

         def calculate_area(self):
             return Circle.PI * self.radius ** 2

         def calculate_circumference(self):
             return 2 * Circle.PI * self.radius

     # Create a circle object with a specified radius
     radius = float(input("Enter the radius of the circle: "))
     circle = Circle(radius)

     # Calculate and display the area and circumference
     area = circle.calculate_area()
     circumference = circle.calculate_circumference()

     print(f"Area of the circle: {area}")
     print(f"Circumference of the circle: {circumference}")\
```

```
Area of the circle: 1519.76
Circumference of the circle: 138.16
```

1.0.5 5) WAP in python that has a class Student that stores the name ,roll no, marks(in three subjects). And display the information(name, roll no, total marks) stored about the students.

```
[ ]: class Student:
         def __init__(self, name, roll_no):
             self.name = name
             self.roll_no = roll_no
             self.marks = []

         def add_marks(self, subject, marks):
             self.marks.append((subject, marks))

         def calculate_total_marks(self):
             total_marks = sum(marks for _, marks in self.marks)
             return total_marks

         def display_information(self):
```

```
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Total Marks: {self.calculate_total_marks()}")

# Create a list to store student objects
students = []

# Get the number of students from user
num_students = int(input("Enter the number of students: "))

# Get the details of each student from user
for i in range(num_students):
    name = input(f"Enter the name of student {i+1}: ")
    roll_no = input(f"Enter the roll number of student {i+1}: ")
    student = Student(name, roll_no)

    # Get the marks for three subjects from user
    for subject in ['Subject 1', 'Subject 2', 'Subject 3']:
        marks = float(input(f"Enter the marks for {subject}: "))
        student.add_marks(subject, marks)

    students.append(student)

# Display the information of all students
for student in students:
    student.display_information()
```

```
Name: Harshil
Roll No: 2105889
Total Marks: 297.0
Name: Eshita
Roll No: 2105962999
Total Marks: 300.0
```

1.0.6 6) WAP in python that has a class Rectangle with attribute length and breadth and a method area that returns the area of the circle.

[ ]:
```
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth

    def area(self):
        return self.length * self.breadth

# Create a rectangle object with a specified length and breadth
length = float(input("Enter the length of the rectangle: "))
breadth = float(input("Enter the breadth of the rectangle: "))
```

```python
rectangle = Rectangle(length, breadth)

# Calculate and display the area of the rectangle
area = rectangle.area()
print(f"Area of the rectangle: {area}")
```

```
Area of the rectangle: 12.0
```

1.0.7 7) WAP in python that has a class Fraction with attributes numerator and denominator. Enter the values to the attributes and print the function in simplified forms.

[ ]:
```python
class Fraction:
    def __init__(self, numerator, denominator):
        self.numerator = numerator
        self.denominator = denominator

    def simplify(self):
        gcd = self.calculate_gcd(self.numerator, self.denominator)
        simplified_numerator = self.numerator // gcd
        simplified_denominator = self.denominator // gcd
        return f"{simplified_numerator}/{simplified_denominator}"

    def calculate_gcd(self, a, b):
        while b:
            a, b = b, a % b
        return a

# Get the numerator and denominator from user
numerator = int(input("Enter the numerator: "))
denominator = int(input("Enter the denominator: "))

# Create a fraction object
fraction = Fraction(numerator, denominator)

# Simplify the fraction and print the result
simplified_fraction = fraction.simplify()
print(f"Simplified fraction: {simplified_fraction}")
```

```
Simplified fraction: 3/4
```

1.0.8 8) WAP in python that has an abstract class Polygon. Derive two classes Rectangle and Triangle from polygon and write methods to get the details of their dimensions and hence calculate the area.

[ ]:
```python
from abc import ABC, abstractmethod

class Polygon(ABC):
    @abstractmethod
    def get_dimensions(self):
        pass

    @abstractmethod
    def calculate_area(self):
        pass

class Rectangle(Polygon):
    def get_dimensions(self):
        length = float(input("Enter the length of the rectangle: "))
        breadth = float(input("Enter the breadth of the rectangle: "))
        return length, breadth

    def calculate_area(self):
        length, breadth = self.get_dimensions()
        area = length * breadth
        return area

class Triangle(Polygon):
    def get_dimensions(self):
        base = float(input("Enter the base length of the triangle: "))
        height = float(input("Enter the height of the triangle: "))
        return base, height

    def calculate_area(self):
        base, height = self.get_dimensions()
        area = 0.5 * base * height
        return area

# Create a rectangle object and calculate its area
rectangle = Rectangle()
rectangle_area = rectangle.calculate_area()
print(f"Area of the rectangle: {rectangle_area}")

# Create a triangle object and calculate its area
triangle = Triangle()
triangle_area = triangle.calculate_area()
print(f"Area of the triangle: {triangle_area}")
```

```
Area of the rectangle: 12.0
Area of the triangle: 30.0
```

1.0.9 9) WAP in python with class Bill. The users have the option to pay the bill by card or by cash. Use the inheritance to model this situation.

[ ]:
```python
# Get the payment method from user
payment_method = input("Enter the payment method (card/cash): ")

# Create a bill object based on the payment method
if payment_method.lower() == "card":
    bill = CardBill(amount)
elif payment_method.lower() == "cash":
    bill = CashBill(amount)
else:
    print("Invalid payment method entered.")
    bill = None

# Get the bill amount from user
amount = float(input("Enter the bill amount: "))

# Display the payment method and bill amount
if bill:
    bill.display_payment_method()
    print(f"Bill amount: {bill.amount}")
    print("Bill Paid Successfully by the customer, Thank you for shopping with␣
 ↪us!")

class Bill:
    def __init__(self, amount):
        self.amount = amount

    def display_payment_method(self):
        pass

class CardBill(Bill):
    def display_payment_method(self):
        print("Payment method: Card")

class CashBill(Bill):
    def display_payment_method(self):
        print("Payment method: Cash")
```

```
Payment method: Card
Bill amount: 1000.0
Bill Paid Successfully by the customer, Thank you for shopping with us!
```

1.0.10 10) WAP in python has a class Person. Inherit a class Faculty from person which also have a class publications.

```
[ ]: class Person:
         def __init__(self, name):
             self.name = name

         def display_info(self):
             print(f"Name: {self.name}")

     class Faculty(Person):
         def __init__(self, name, department):
             super().__init__(name)
             self.department = department

         def display_info(self):
             super().display_info()
             print(f"Department: {self.department}")

     class Publications:
         def __init__(self, publications):
             self.publications = publications

         def display_publications(self):
             print("Publications:")
             for publication in self.publications:
                 print(publication)

     class FacultyWithPublications(Faculty, Publications):
         def __init__(self, name, department, publications):
             Faculty.__init__(self, name, department)
             Publications.__init__(self, publications)

         def display_info(self):
             super().display_info()
             super().display_publications()

     # Create a faculty object with publications
     faculty = FacultyWithPublications("John Doe", "Computer Science", ["Publication␣
      ↪1", "Publication 2"])

     # Display the faculty information and publications
     faculty.display_info()
```

```
Name: John Doe
Department: Computer Science
Publications:
Publication 1
```

Publication 2