lab-06

March 1, 2023

1 Questions

1. WAP to count the number of objects created
2. Write a program that has a class Student storing student information including DOB. The program should subtract DOB from present date to find out whether you are eligible to cast vote or not.
3. WAP to enter a number, find the factorial of a number, check, whether the number is prime or not, find the square of that number, through object and class concept
4. WAP a WAP that has a class circle. Use class variable to define the values of constant PI. Use class variable to calculate area and circumference with specified radius.
5. WAP that has a class Student that stores roll number, name, and marks (in three subjects). Display information roll number, total marks stored about a student.
6. Write a class rectangle that has attribute length, Breadth and a method area which returns area of the rectangle.
7. WAP to enter sides of different geometric figure, find the area of any 5 geometric figure
8. WAP to program to calculate simple interest and compound interest with appreciate input given to the program e.g p, r, t input.
9. WAP to enter the recipes ordered by the customer, in a restaurant, prepare the bill in a proper format.
10. Write a menu driven program that keeps records of books and journal available in a library.
11. Write a program that uses a class attributes to define some defaults titles in a college. Dis[lay the name along with title and department of the college.
12. Write a menu driven program to read, add, subtract, multiply, divide and transpose two matrices.

2 Answers

Dipankar Das 20051554

2.1 WAP to count the number of objects created

[3]:
```
class Q1:
  counter = 0
  def __init__(self):
    Q1.counter += 1

o1 = Q1()
```

```
o2 = Q1()
print(Q1.counter)
```

2

2.2 Write a program that has a class Student storing student information including DOB. The program should subtract DOB from present date to find out whether you are eligible to cast vote or not.

[21]:
```
from datetime import date

class Student:
  def __init__(self):
    self.name = input("Enter the student name")
    self.roll = int(input("enter the roll number"))
    self.yy = int(input("enter DOB year"))
    self.mm = int(input("enter DOB month"))
    self.dd = int(input("enter DOB day"))
    # print(datetime.today().year)
  def getInfo(self):
    print(f"Name: %s" % self.name)
    print(f"Roll: %d" % self.roll)
    currDate = date.today()
    dob = date(self.yy, self.mm, self.dd)
    ageSec = (currDate - dob).total_seconds()
    age = ageSec / (60 * 60 * 24 * 365)
    print(f"Age: %s" % age)

a = Student()
a.getInfo()
```

```
Name: dipankar
Roll: 20051554
DOB: 21.353424657534248
```

2.3 WAP to enter a number, find the factorial of a number, check, whether the number is prime or not, find the square of that number, through object and class concept

[24]:
```
class Q3:
  def __init__(self, n: int):
    self.number = n
  def __findFactorial(self)-> int:
    fact = 1
    n = self.number
    while n > 1:
      fact *= n
      n -= 1
```

```
    return fact
  def __findPrime(self)-> bool:
    for i in range(2, self.number):
      if self.number % i == 0:
        return False
    return True

  def __getSquare(self)-> float:
    return self.number * self.number

  def getResults(self):
    print(f"Factorial: {Q3.__findFactorial(self)}")
    print(f"Prime: {Q3.__findPrime(self)}")
    print(f"Square: {Q3.__getSquare(self)}")

o = Q3(5)
o.getResults()
```

```
Factorial: 120
Prime: True
Square: 25
```

2.4 WAP that has a class circle. Use class variable to define the values of constant PI. Use class variable to calculate area and circumference with specified radius.

[25]:
```
class Circle:
  PI = 3.1473
  def __init__(self, radius: float):
    self.radius = radius
  def area(self)-> float:
    return Circle.PI * pow(self.radius, 2)
  def circumference(self)-> float:
    return 2.0 * Circle.PI * pow(self.radius, 2)

cir = Circle(2)
print(f"Area: {cir.area()}")
print(f"Circumference: {cir.circumference()}")
```

```
Area: 12.5892
Circumference: 25.1784
```

2.5 WAP that has a class Student that stores roll number, name, and marks (in three subjects). Display information roll number, total marks stored about a student.

[28]:
```
class Student:
  def __init__(self):
    self.name = input("Enter the student name")
    self.roll = int(input("enter the roll number"))
    self.marks = []
    self.marks.append(int(input("Enter the marks for sub1")))
    self.marks.append(int(input("Enter the marks for sub2")))
    self.marks.append(int(input("Enter the marks for sub3")))
  def display(self):
    print(f"Name: {self.name}")
    print(f"Roll: {self.roll}")
    print(f"marks: {self.marks}")
    print(f"total marks: {self.marks[0]+self.marks[1]+self.marks[2]}")

o = Student()
o.display()
```

```
Name: dipankar
Roll: 20051554
marks: [40, 90, 100]
total marks: 230
```

2.6 Write a class rectangle that has attribute length, Breadth and a method area which returns area of the rectangle.

[29]:
```
class Rectangle:
  def __init__(self, l: float, b: float):
    self.length = l
    self.bredth = b
  def area(self)-> float:
    return self.length * self.bredth

o = Rectangle(2,5)
print(f"Area: {o.area()}")
```

```
Area: 10
```

2.7 WAP to enter sides of different geometric figure, find the area of any 5 geometric figure

[30]:
```
from math import sqrt

class GeometricFigure:
    def __init__(self, sides):
```

```
        self.sides = sides

    # def area(self):
    #     pass

class Triangle(GeometricFigure):
    def area(self):
        a, b, c = self.sides
        s = (a + b + c) / 2
        return sqrt(s * (s - a) * (s - b) * (s - c))

class Rectangle(GeometricFigure):
    def area(self):
        a, b = self.sides
        return a * b

class Square(GeometricFigure):
    def area(self):
        a = self.sides[0]
        return a ** 2

class Circle(GeometricFigure):
    def area(self):
        r = self.sides[0]
        return 3.14 * r ** 2

class Trapezium(GeometricFigure):
    def area(self):
        a, b, h = self.sides
        return ((a + b) * h) / 2

# Example usage
print("Enter the sides of a triangle:")
a = float(input("a = "))
b = float(input("b = "))
c = float(input("c = "))
t = Triangle([a, b, c])
print(f"The area of the triangle is {t.area()}")

print("Enter the sides of a rectangle:")
a = float(input("a = "))
b = float(input("b = "))
r = Rectangle([a, b])
print(f"The area of the rectangle is {r.area()}")

print("Enter the side of a square:")
a = float(input("a = "))
```

```
s = Square([a])
print(f"The area of the square is {s.area()}")

print("Enter the radius of a circle:")
r = float(input("r = "))
c = Circle([r])
print(f"The area of the circle is {c.area()}")

print("Enter the sides of a trapezium:")
a = float(input("a = "))
b = float(input("b = "))
h = float(input("h = "))
tr = Trapezium([a, b, h])
print(f"The area of the trapezium is {tr.area()}")
```

```
Enter the sides of a triangle:
The area of the triangle is 2.9047375096555625
Enter the sides of a rectangle:
The area of the rectangle is 30.0
Enter the side of a square:
The area of the square is 49.0
Enter the radius of a circle:
The area of the circle is 200.96
Enter the sides of a trapezium:
The area of the trapezium is 10.0
```

2.8 WAP to program to calculate simple interest and compound interest with appreciate input given to the program e.g p, r, t input.

[31]:
```
def simple_interest(p, r, t):
    return (p * r * t) / 100

def compound_interest(p, r, t):
    return p * ((1 + (r / 100)) ** t - 1)

p = float(input("Enter the principal amount: "))
r = float(input("Enter the annual interest rate: "))
t = float(input("Enter the time period (in years): "))

si = simple_interest(p, r, t)
ci = compound_interest(p, r, t)

print(f"Simple Interest = {si:.2f}")
print(f"Compound Interest = {ci:.2f}")
```

```
Simple Interest = 600.00
Compound Interest = 618.00
```

2.9 WAP to enter the recipes ordered by the customer, in a restaurant, prepare the bill in a proper format.

[48]:
```
class Hotel:
  def __init__(self):
    self.Items = []
  def add(self, item: dict):
    self.Items.append(item)
  def print(self):
    print("Item\tPrice")
    amt = 0
    for item in self.Items:
      amt += item['price']
      print(f"{item['name']}\t{item['price']}")
    print(f"Total bill: {amt}")
o = Hotel()
o.add({"name": "butter", "price": 34})
o.add({"name": "sugar", "price": 356})
o.add({"name": "chicken", "price": 234})
o.print()
```

```
Item    Price
butter  34
sugar   356
chicken 234
Total bill: 624
```

2.10 Write a menu driven program that keeps records of books and journal available in a library.

[59]:
```
class Items:
  def __init__(self, name: str="", description: str="", author: str="", Journal:
  ↪ str="N/A"):
    self.name = name
    self.description = description
    self.author = author
    self.journal_publisher = Journal

class Library:
  def __init__(self):
    self.records: list[Items] = []
  def addRecords(self, item: Items):
    self.records.append(item)
  def getAllRecords(self):
    for item in self.records:
      print(f"Name: {item.name}")
      print(f"Author: {item.author}")
      print(f"Description: {item.description}")
```

```
      print(f"Journal Publisher: {item.journal_publisher}")

if __name__ == "__main__":
  libraryObject = Library()
  while True:
    choice = input("Enter the book or journal[b/j] and [n] for exit()")
    if choice == "n":
      libraryObject.getAllRecords()
      break
    if choice != "b" and choice != "j":
      print("Invalid choice")
      break
    name = input("Enter the name of the book or journal")
    description = input("Enter the description of the book or journal")
    author = input("Enter the author of the book or journal")
    if choice == "b":
      libraryObject.addRecords(Items(name, description, author))
    else:
      journal_publisher = input("Enter the publisher of the book or journal")
      libraryObject.addRecords(Items(name, description, author,␣
 ↪journal_publisher))
```

```
Name: book1
Author: auth1
Description: sdcwcdwc
Journal Publisher: N/A
Name: jour1
Author: 223dewe
Description: ewce
Journal Publisher: ewdc
Name: edwdw
Author: wdcswac
Description: wadcc
Journal Publisher: N/A
```

2.11 Write a program that uses a class attributes to define some defaults titles in a college. Dis[lay the name along with title and department of the college.

[61]:
```
class College:
    # Default titles in the college
    default_title = "Professor"
    default_department = "CSE"

    def __init__(self, name):
        self.name = name
```

```
    def display_info(self):
        print(f"{self.name}: {College.default_title} in {College.
 ↪default_department}")


# Example usage
c1 = College("ABC1")
c1.display_info()  # Output: John: Professor in Computer Science

c2 = College("ABCD@")
College.default_title = "Associate Professor"
College.default_department = "Maths"
c2.display_info()  # Output: Sarah: Associate Professor in Mathematics
```

```
ABC1: Professor in CSE
ABCD@: Associate Professor in Maths
```

2.12 Write a menu driven program to read, add, subtract, multiply, divide and transpose two matrices.

[2]:
```
import numpy as np

def read_matrix():
    rows = int(input("Enter the number of rows: "))
    cols = int(input("Enter the number of columns: "))
    print("Enter the elements of the matrix:")
    matrix = []
    for i in range(rows):
        row = list(map(int, input().split()))
        if len(row) != cols:
            print("Error: Invalid number of elements in row. Please try again.")
            return None
        matrix.append(row)
    return np.array(matrix)

def add_matrices(m1, m2):
    if m1.shape != m2.shape:
        print("Error: Matrices must have the same shape.")
        return None
    return m1 + m2

def subtract_matrices(m1, m2):
    if m1.shape != m2.shape:
        print("Error: Matrices must have the same shape.")
        return None
    return m1 - m2
```

```
def multiply_matrices(m1, m2):
    if m1.shape[1] != m2.shape[0]:
        print("Error: Invalid matrix dimensions for multiplication.")
        return None
    return np.matmul(m1, m2)

def divide_matrices(m1, m2):
    if m1.shape != m2.shape:
        print("Error: Matrices must have the same shape.")
        return None
    return m1 / m2

def transpose_matrix(matrix):
    return matrix.transpose()

while True:
    print("\n---MATRIX OPERATIONS MENU---")
    print("1. Read two matrices")
    print("2. Add two matrices")
    print("3. Subtract two matrices")
    print("4. Multiply two matrices")
    print("5. Divide two matrices")
    print("6. Transpose a matrix")
    print("7. Exit")

    choice = int(input("Enter your choice (1-7): "))

    if choice == 1:
        print("\nEnter the first matrix:")
        matrix1 = read_matrix()
        if matrix1 is not None:
            print("\nEnter the second matrix:")
            matrix2 = read_matrix()
            if matrix2 is not None:
                print("\nMatrix 1:")
                print(matrix1)
                print("\nMatrix 2:")
                print(matrix2)

    elif choice == 2:
        result = add_matrices(matrix1, matrix2)
        if result is not None:
            print("\nResult:")
            print(result)

    elif choice == 3:
        result = subtract_matrices(matrix1, matrix2)
```

```
        if result is not None:
            print("\nResult:")
            print(result)

    elif choice == 4:
        result = multiply_matrices(matrix1, matrix2)
        if result is not None:
            print("\nResult:")
            print(result)

    elif choice == 5:
        result = divide_matrices(matrix1, matrix2)
        if result is not None:
            print("\nResult:")
            print(result)

    elif choice == 6:
        matrix = read_matrix()
        if matrix is not None:
            result = transpose_matrix(matrix)
            print("\nResult:")
            print(result)

    elif choice == 7:
        print("Exiting program...")
        break

    else:
        print("Error: Invalid choice. Please try again.")
```

```
---MATRIX OPERATIONS MENU---
1. Read two matrices
2. Add two matrices
3. Subtract two matrices
4. Multiply two matrices
5. Divide two matrices
6. Transpose a matrix
7. Exit
Enter your choice (1-7): 1

Enter the first matrix:
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements of the matrix:
1 2
3 4
```

```
Enter the second matrix:
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements of the matrix:
5 6
7 8

Matrix 1:
[[1 2]
 [3 4]]

Matrix 2:
[[5 6]
 [7 8]]

---MATRIX OPERATIONS MENU---
1. Read two matrices
2. Add two matrices
3. Subtract two matrices
4. Multiply two matrices
5. Divide two matrices
6. Transpose a matrix
7. Exit
Enter your choice (1-7): 2

Result:
[[ 6  8]
 [10 12]]

---MATRIX OPERATIONS MENU---
1. Read two matrices
2. Add two matrices
3. Subtract two matrices
4. Multiply two matrices
5. Divide two matrices
6. Transpose a matrix
7. Exit
Enter your choice (1-7): 3

Result:
[[-4 -4]
 [-4 -4]]

---MATRIX OPERATIONS MENU---
1. Read two matrices
2. Add two matrices
3. Subtract two matrices
4. Multiply two matrices
```

```
5. Divide two matrices
6. Transpose a matrix
7. Exit
Enter your choice (1-7): 4

Result:
[[19 22]
 [43 50]]

---MATRIX OPERATIONS MENU---
1. Read two matrices
2. Add two matrices
3. Subtract two matrices
4. Multiply two matrices
5. Divide two matrices
6. Transpose a matrix
7. Exit
Enter your choice (1-7): 5

Result:
[[0.2        0.33333333]
 [0.42857143 0.5       ]]

---MATRIX OPERATIONS MENU---
1. Read two matrices
2. Add two matrices
3. Subtract two matrices
4. Multiply two matrices
5. Divide two matrices
6. Transpose a matrix
7. Exit
Enter your choice (1-7): 6
Enter the number of rows: 7
Enter the number of columns: 3
Enter the elements of the matrix:
3 5 8
4 5 67
23 234 5
23 4 4
1 1 1
0 0 0
11 12 23

Result:
[[  3   4  23  23   1   0  11]
 [  5   5 234   4   1   0  12]
 [  8  67   5   4   1   0  23]]
```

```
---MATRIX OPERATIONS MENU---
1. Read two matrices
2. Add two matrices
3. Subtract two matrices
4. Multiply two matrices
5. Divide two matrices
6. Transpose a matrix
7. Exit
Enter your choice (1-7): 7
Exiting program…
```