Practical 10
Composition and Operator Overloading

March 24, 2019

Q.1 (The Triangle class) Design a class named Triangle that extends the GeometricObject class. The Triangle class contains:

- Three float data fields named side1, side2, and side3 to denote the three sides of the triangle.
- A constructor that creates a triangle with the specified side1, side2, and side3 with default values 1.0.
- The accessor methods for all three data fields.
- A method named getArea() that returns the area of this triangle.
- A method named getPerimeter() that returns the perimeter of this triangle.
- A method named **str**()that returns a string description for the triangle.
- The **str** ()method is implemented as follows: return "Triangle: side1 = " + str(side1) + " side2 = " + str(side2) + " side3 = " + str(side3)

```
In [2]: class Geometry:
            pass
        class Triangle(Geometry):
            def __init__(self, side1=1, side2=1, side3=1):
                self.side1 = side1
                self.side2 = side2
                self.side3 = side3

            def get_side_1(self):
                return self.side1

            def get_side_2(self):
                return self.side2

            def get_side_3(self):
                return self.side3
            def get_area(self):
                p = (self.side1 + self.side2 + self.side3)/2
                from math import sqrt
                return sqrt(p*(p-self.side1)*(p-self.side2)*(p-self.side3))

            def get_perimeter(self):
                return self.side1 + self.side2 + self.side3
```

```
            def __str__(self):
                return f'Triangle(side1={self.side1}, side2={self.side2}, side3={self.side3})'
        if __name__ == '__main__':
            t = Triangle(*map(int, input('Enter Triangle Sides : ').split()))
            print('Area of Triangle :', t.get_area())
            print('Perimeter of Triangle :', t.get_perimeter())
            print(t)
```

```
Enter Triangle Sides : 4 3 6
Area of Triangle : 5.332682251925386
Perimeter of Triangle : 13
Triangle(side1=4, side2=3, side3=6)
```

Q.2 User wants a bank account in the bank. (use object oriented fundamentals)

- Provide all the details to create bank account like Name, Account Number, Type of Account, Balance. (Class)
- User have to use bank facility like withdraw, deposit. (Methods)
- Check user withdraw amount or not if amount is less than 500.

```
In [4]: class Account:
            def __init__(self, name, acc_type,account_number, balance):
                self.name = name
                self.acc_type = acc_type
                self.balance = balance
                self.account_number = account_number

            def withdraw(self, amount):
                if self.balance < 500:
                    print('Minimum Balance Required Rs. 500')
                    return
                if self.balance - amount > 0:
                    self.balance -= amount

            def deposit(self, amount):
                self.balance += amount

            def set_balance(self, amount):
                self.balance = amount

            def get_balance(self):
                return self.balance

        if __name__ == '__main__':
            name = input('Enter Name : ')
            acc_type = input('Enter Account Type : ')
            account_number = input('Enter Account Number :')
```

```
            balance = float(input('Enter Balance : '))
            customer = Account(name, acc_type, account_number,  balance)
            while True:
                print('1 : Withdraw ')
                print('2 : Deposit ')
                print('3 : Quit ')
                choice = input('Enter Your Choice :')
                if choice == '1':
                    amount = float(input('Enter Amount : '))
                    customer.withdraw(amount)
                    print('New Balance : ', customer.get_balance())
                elif choice == '2':
                    amount = float(input('Enter Amount : '))
                    customer.deposit(amount)
                    print('New Balance : ', customer.get_balance())
                elif choice == '3':
                    break
                else:
                    print('Invalid Choice')
```

```
Enter Name : Nikhil
Enter Account Type : savings
Enter Account Number :99098
Enter Balance : 10000
1 : Withdraw
2 : Deposit
3 : Quit
Enter Your Choice :1
Enter Amount : 1000
New Balance :  9000.0
1 : Withdraw
2 : Deposit
3 : Quit
Enter Your Choice :2
Enter Amount : 400
New Balance :  9400.0
1 : Withdraw
2 : Deposit
3 : Quit
Enter Your Choice :3
```

Q.3 Implement the stack class along with its operations using OOP concept .

```
In [5]: class Stack:
            __stack = []

            def __init__(self, size=10):
```

```
            self.__size = size

        def get_size(self):
            return self.__size

        def push(self, value):
            if len(self.__stack) is 10:
                print('Stack OverFlow..')
                return False
            else:
                self.__stack.append(value)
            return True

        def peek(self):
            if len(self.__stack) is 0:
                print('Stack UnderFlow..')
                return
            return self.__stack[-1]

        def pop(self):
            if len(self.__stack) is 0:
                print('Stack UnderFlow..')
            else:
                return self.__stack.pop()

        def __str__(self):
                return f'Stack({self.__stack})'

    if __name__ == '__main__':
        stack = Stack()
        stack.push(5)
        stack.push(6)
        stack.push(18)
        print(stack)
        stack.push(15)
        stack.push(19)
        print(stack)
        stack.pop()
        print(stack)
        stack.pop()
        print(stack)
```

```
Stack([5, 6, 18])
Stack([5, 6, 18, 15, 19])
Stack([5, 6, 18, 15])
Stack([5, 6, 18])
```