WEEKLY TASK 3

NAME : SANJEEV N
DATE : 26-07-25

PROBLEM 1 : Create a class BankAccount in Python with private attributes __accountno,__name, __balance.
Add
parameterized constructor
methods:
deposit(amount)
withdraw(amount)
set_accountno
get_accountno
set_name
get_name
get_balance()
set_balance()

```
Account No: 1
Name: Sanjeev
Balance: 5000
Deposited 10500. New balance is 15500
Withdrew 12000. New balance is 3500
Insufficient balance or invalid amount
```

```python
class BankAccount:
    def __init__(self, accountno, name, balance):
        self.__accountno = accountno
        self.__name = name
        self.__balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
            print(f"Deposited {amount}. New balance is {self.__balance}")
        else:
            print("Invalid deposit amount")

    def withdraw(self, amount):
        if 0 < amount <= self.__balance:
            self.__balance -= amount
            print(f"Withdrew {amount}. New balance is {self.__balance}")
```

```
        else:
            print("Insufficient balance or invalid amount")

    def set_accountno(self, accountno):
        self.__accountno = accountno

    def get_accountno(self):
        return self.__accountno

    def set_name(self, name):
        self.__name = name

    def get_name(self):
        return self.__name

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

    def get_balance(self):
        return self.__balance

acc = BankAccount(1, "Sanjeev", 5000)
print("Account No:", acc.get_accountno())
print("Name:", acc.get_name())
print("Balance:", acc.get_balance())

acc.deposit(10500)
acc.withdraw(12000)
acc.withdraw(26000)
```

PROBLEM 2 : How will you define a static method in Python?Explore and give an example.

```
eskto
15
```

```
class Calculator:
    @staticmethod
    def add(a, b):
```

```
        return a + b

print(Calculator.add(10, 5))
```

PROBLEM 3 : Give examples for dunder methods in Python other than __str__ and __init__ .

```
[1, 2, 3, 4, 5]
3
```

```
class Demo:
    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        return self.value + other.value

    def __len__(self):
        return len(self.value)

d1 = Demo([1,2,3])
d2 = Demo([4,5])
print(d1 + d2)
print(len(d1))
```

PROBLEM 4 : Explore some supervised and unsupervised models in ML.

Supervised Models (with labelled data):

Linear Regression (predict continuous output)

Logistic Regression (binary classification)

Decision Trees

Random Forest

Support Vector Machine (SVM)

K-Nearest Neighbors (KNN)

Naive Bayes

Unsupervised Models (unlabelled data):

K-Means Clustering

Hierarchical Clustering

DBSCAN

Principal Component Analysis (PCA) – for dimensionality reduction

Apriori Algorithm – market basket analysis (association rules**)**

PROBLEM 5 : Implement Stack with class in Python.

```
Pushed: 10
Pushed: 20
Top element: 20
Popped: 20
Is stack empty? False
```

```python
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)
        print(f"Pushed: {item}")

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            return "Stack is empty"

    def peek(self):
        if not self.is_empty():
```

```
            return self.items[-1]
        else:
            return "Stack is empty"

    def is_empty(self):
        return len(self.items) == 0

    def size(self):
        return len(self.items)


s = Stack()
s.push(10)
s.push(20)
print("Top element:", s.peek())
print("Popped:", s.pop())
print("Is stack empty?", s.is_empty())
```