Week End Task
Module-3

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()

Coding :

```
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: Rs.{amount}")
        else:
            print("Invalid deposit amount!")

    def withdraw(self, amount):
        if 0 < amount <= self.__balance:
            self.__balance -= amount
            print(f"Withdrawn: Rs.{amount}")
        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):
        if balance >= 0:
            self.__balance = balance
        else:
            print("Balance cannot be negative!")

    def get_balance(self):
        return self.__balance



if __name__ == "__main__":

    acc = BankAccount(1234567, "Alice", 5000)

    print("Account Number:", acc.get_accountno())
    print("Account Holder Name:", acc.get_name())
    print("Current Balance: Rs.", acc.get_balance())

    acc.deposit(1500)

    acc.withdraw(1000)


    print("Updated Balance: Rs.", acc.get_balance())


    acc.set_name("Alice Ashok")
    print("Updated Account Holder Name:", acc.get_name())
```

OUTPUT:

```
Account Number: 1234567
Account Holder Name: Alice
Current Balance: Rs. 5000
Deposited: Rs.1500
Withdrawn: Rs.1000
Updated Balance: Rs. 5500
Updated Account Holder Name: Alice Ashok
```

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

CODING:

```
class MathOperations:

    @staticmethod
    def add(a, b):
        return a + b

    @staticmethod
    def multiply(a, b):
        return a * b


print("Addition:", MathOperations.add(10, 5))
print("Multiplication:", MathOperations.multiply(4, 3))
```

OUTPUT:

```
Addition: 15
Multiplication: 12
```

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

CODING:

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

    def __repr__(self):
        return f"Person(name='{self.name}')"
```

```
p = Person("Alice")
print(repr(p))
class MyData:
    def __init__(self, data):
        self.data = data

    def __getitem__(self, index):
        return self.data[index]

obj = MyData(["apple", "banana", "cherry"])
print(obj[1])

class MyList:
    def __init__(self, items):
        self.items = items

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

ml = MyList([1, 2, 3])
print(len(ml))
```

OUTPUT:

```
Person(name='Alice')
banana
```

4. Explore some supervised and unsupervised models in ML.

supervised

Linear Regression

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Unsupervised

K-Means Clustering

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3)
model.fit(data)
```

labels = model.predict(data)

5.Implement Stack with class in Python.

CODING:

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

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


    def pop(self):
        if not self.is_empty():
            removed = self.stack.pop()
            print(f"Popped: {removed}")
            return removed
        else:
            print("Stack is empty. Cannot pop.")

    def peek(self):
        if not self.is_empty():
            return self.stack[-1]
        else:
            print("Stack is empty.")
            return None


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

    def display(self):
        if self.is_empty():
            print("Stack is empty.")
        else:
            print("Stack contents (top to bottom):")
            for item in reversed(self.stack):
                print(item)
if __name__ == "__main__":
    s = Stack()
```

```
s.push(10)
s.push(20)
s.push(30)

s.display()

print("Top element is:", s.peek())

s.pop()
s.display()

print("Is stack empty?", s.is_empty())
```

OUTPUT: