Week3
Dharunya S
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()

```
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} successfully.")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if 0 < amount <= self.__balance:
            self.__balance -= amount
            print(f"Withdrew ₹{amount} successfully.")
        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
if __name__ == "__main__":
    account = BankAccount(123456, "Alice", 1000)
    account.deposit(500)
    account.withdraw(200)
    print("Account Number:", account.get_accountno())
    print("Account Holder:", account.get_name())
    print("Account Balance:", account.get_balance())
```
output
```
python account.py
Deposited ₹500 successfully.
Withdrew ₹200 successfully.
Account Number: 123456
Account Holder: Alice
Account Balance: 1300
```

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

A static method is defined using the `@staticmethod` decorator. It belongs to the class but does not access or modify class or instance variables.
```
class MathUtils:
    @staticmethod
    def add(a, b):
        return a + b
print(MathUtils.add(5, 7))
```
output
```
python static.py
12
```

3.Give examples for dunder methods in Python other than __str__ and __init__ .
)
```
class Dunder:
    def __init__(self, title):
        self.title = title

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

dunder = Dunder("Python ")
print(len(dunder))
```
Output:
```
python dunder.py
7
```

4. Explore some supervised and unsupervised models in ML.

Supervised Learning Models

- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forest
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Gradient Boosting (XGBoost, LightGBM)

Unsupervised Learning Models

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN
- Principal Component Analysis (PCA)
- Autoencoders
- t-SNE / UMAP

5.Implement Stack with class in Python.
```
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(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)
s.push(30)
s.push(40)
print(s.is_empty())
print(s.size())
print(s.pop())
print(s.peek())
```

Output:

```
python stack.py
False
4
40
```