KARTHIKEYAN
25/07/2025

1.

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

    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

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

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

```
        else:
            print("Withdraw amount must be positive.")

account = BankAccount(101, "Karthi", 5000)

account.deposit(1500)

account.withdraw(2000)

print(account.get_accountno())
print(account.get_name())
print(account.get_balance())

account.set_name("Karthikeyan")
print(account.get_name())
```

OUTPUT

```
python.exe c:/Users/subramaniya.k/Desktop/WeeklyTask3-25-07-2025/Task1.py
Deposited 1500. New balance: 6500
Withdrew 2000. New balance: 4500
101
Karthi
4500
Karthikeyan
```

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

In Python, a static method is defined using the `@staticmethod` decorator. Static methods do not receive the implicit `self` (for instances) or `cls`(for class itself) parameter. They behave like plain functions, but reside in the class's namespace. Static methods are useful when some functionality is related to the class but does not require access to any instance- or class-specific data.

- Use the `@staticmethod` decorator just above the method definition in the class.
- The method does not take `self` or `cls` as its first argument.

EXAMPLE

```python
class BankAccount:

    @staticmethod
    def is_positive_amount(amount):
        return amount > 0

# Usage:
print(BankAccount.is_positive_amount(100)) # True
print(BankAccount.is_positive_amount(-5))  # False
```

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

- add(self, other): Enables use of the + operator.
- repr(self): Returns the official string representation of an object, often used for debugging
- len(self): Used by the built-in len() function.
- getitem(self, key): Allows bracket notation indexing (e.g., obj[key]).
- eq(self, other): Supports == comparisons.
- call(self, ...): Makes the object callable like a function.
- enter(self) and exit(self, exc_type, exc_val, exc_tb): Enable an object to be used as a context manager (with statement).
- iter(self) and next(self): Make object iterable.
- del(self): Called when the object is about to be destroyed.

KARTHIKEYAN
25/07/2025

4. Explore some supervised and unsupervised models in ML.

Supervised Learning

Supervised learning algorithms use labeled datasets—meaning each input data point comes with a known output. The model learns to predict the output from the input by generalizing from examples. Common tasks are classification (discrete outputs, like spam detection) and regression (continuous outputs, like predicting housing prices).

- Linear Regression: Predicts continuous values (e.g., predicting sales numbers).
- Logistic Regression: Used for binary classification (e.g., email spam/not-spam).
- k-Nearest Neighbors (kNN): Classifies new cases based on similarity to neighboring cases.
- Decision Trees: Models decisions by splitting data based on feature values.
- Support Vector Machines (SVM): Finds the optimal boundary separating different classes.
- Random Forests: Ensemble of decision trees for more accurate predictions.

How it works:

> The algorithm is trained on labeled data, learns the mapping from inputs to outputs, and is then evaluated on new, unseen data to measure how well it generalizes.

Unsupervised Learning

Unsupervised learning deals with unlabeled data—there are no target outputs, and the model tries to find patterns, relationships, or structure within the dataset by itself. The goal might be to group similar items, reduce dimensionality, or detect outliers.

- K-Means Clustering: Groups data points into clusters based on similarity (e.g., customer segmentation).
- Hierarchical Clustering: Builds a hierarchy of clusters.

- Principal Component Analysis (PCA): Reduces the number of dimensions (features) while retaining key information.
- t-SNE (t-Distributed Stochastic Neighbor Embedding): Visualizes high-dimensional data by reducing dimensions.
- Autoencoders: Neural networks used for data compression and anomaly detection.
- Generative Adversarial Networks (GANs): Learn to generate new data samples similar to the input dataset.
- One-class SVM: Used for anomaly detection.

How it works:

These algorithms analyze input data to find hidden patterns or structures (such as groupings or correlations) without any predefined labels.

5.Implement Stack with class in Python.

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

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

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            print("Stack Underflow - cannot pop from empty stack.")
```

```
    def peek(self):
        if not self.is_empty():
            return self.items[-1]
        else:
            print("Stack is empty.")

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

    def display(self):
        print("Stack (top -> bottom):", self.items[::-1])

# Creating a stack
s = Stack()

s.push(10)
s.push(20)
s.push(30)
s.push(40)
s.push(50)
s.display()

print(s.peek())

print(s.pop())

print(s.size())

s.display()
```

KARTHIKEYAN
25/07/2025

OUTPUT