In [ ]:

```
# Object-Oriented Programming Assignment

## Q1. Explain Class and Object with respect to Object-Oriented Programming
Give a suitable example.

"""
A class is a blueprint or template to create objects. It defines the struct
and behavior (attributes and methods) of an object.

An object is an instance of a class. It represents a real-world entity
and contains actual data.

Example:
"""

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

    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")

# Creating object of the class
p1 = Person("Amit", 25)
p1.display()
```

In [ ]:

```
## Q2. Name the four pillars of OOPs.

"""
The four pillars of Object-Oriented Programming (OOP) are:
1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism
"""
```

In [ ]:

```
# 3. Multilevel Inheritance
class Grandfather:
    def history(self):
        print("Grandfather: Freedom Fighter")

class Parent(Grandfather):
    def culture(self):
        print("Parent: Values and Discipline")

class Child2(Parent):
    def modern(self):
        print("Child2: Tech-savvy")

c2 = Child2()
c2.history()
c2.culture()
c2.modern()
```

In [ ]:

```
# 4. Hierarchical Inheritance
class Vehicle:
    def start(self):
        print("Vehicle started")

class Car(Vehicle):
    def drive(self):
        print("Car is driving")

class Bike(Vehicle):
    def ride(self):
        print("Bike is riding")

c = Car()
c.start()
c.drive()

b = Bike()
b.start()
b.ride()
```

In [ ]:

```
## Q5. What is inheritance? Give an example for each type of inheritance.

"""
Inheritance allows a class (child/derived class) to inherit properties
and methods from another class (parent/base class).

Types of Inheritance:
1. Single Inheritance
2. Multiple Inheritance
3. Multilevel Inheritance
4. Hierarchical Inheritance
"""

# 1. Single Inheritance
class Animal:
    def sound(self):
        print("Animal makes sound")

class Dog(Animal):
    def bark(self):
        print("Dog barks")

d = Dog()
d.sound()
d.bark()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: