In [ ]:
```
Explain Class and Object with respect to Object-Oriented Programming. Give a suitable example.
```

In [1]:
```
# Class definition
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        print(f"{self.name} is barking!")

# Object creation
my_dog = Dog("Buddy", 3)

# Accessing attributes and calling methods
print(f"{my_dog.name} is {my_dog.age} years old.")
my_dog.bark()
```

```
Buddy is 3 years old.
Buddy is barking!
```

In [ ]:
```
Name the four pillars of OOPs
```

In [ ]:
```
The four pillars of Object-Oriented Programming are:

Encapsulation: The bundling of data and methods that operate on that data into a single unit (class).

Inheritance: The mechanism by which a class can inherit properties and behaviors from another class.

Polymorphism: The ability of objects to take on multiple forms, i.e., the same method name having different implementations :

Abstraction: The process of simplifying complex systems by modeling classes based on essential properties and behaviors.
```

In [ ]:
```
Explain why the __init__() function is used. Give a suitable example.
```

In [7]:
```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

# Creating an object and initializing attributes
my_car = Car("Toyota", "Camry", 2022)
```

In [ ]:
```
Why self is used in OOPs?
```

In [ ]:
```
f is passed as the first argument to the method. self is used to access and modify the attributes of the object within the cl
```

In [ ]:
```
What is inheritance? Give an example for each type of inheritance.
```

In [ ]:
```
SINGLE
```

In [4]:
```
class Animal:
    def speak(self):
        print("Animal speaks")

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

In [ ]:
```
MULTIPLE
```

In [8]:
```
class A:
    def methodA(self):
        print("Method A")

class B:
    def methodB(self):
        print("Method B")

class C(A, B):
    def methodC(self):
        print("Method C")
```

In [ ]:
```
MULTILEVEL
```

In [6]:
```
class A:
    def methodA(self):
        print("Method A")

class B(A):
    def methodB(self):
        print("Method B")

class C(B):
    def methodC(self):
        print("Method C")
```

In [ ]:
```

```