DPP -06

(Coding)

Easy Level:

Fill in the Blanks

Define a Class

class First:

pass

1.Creating an Object

class Dog:

def \_\_init\_\_(self, name):

self.name = name

my\_dog = Dog("Buddy")

2.Using the \_\_init\_\_ Method

class Car:

def \_\_init\_\_(self, make, model):

self.make = make

self.model = model

my\_car = Car("Toyota", "Corolla")

3. Accessing Attributes

class Person:

def \_\_init\_\_(self, name):

self.name = name

p = Person("Alice")

print(p.name) # Should print Alice

4. Defining a Method

class Calculator:

def add(self, a, b):

return a + b

calc = Calculator()

print(calc.add(2, 3)) # Should print 5 if the second argument is 3

5. Class Attributes

class Circle:

pi = 3.14 # pi is a class attribute

print(Circle.pi) # Should print 3.14

6. Instance Attributes

class Book:

def \_\_init\_\_(self, title):

self.title = title

my\_book = Book("1984")

print(my\_book.title) # Should print 1984

Mid-Level Coding Problems

1. Create a Class with Methods

Define a class Rectangle that has methods to calculate the area and perimeter. Instantiate the class and print the results.  
class Rectangle:

def \_\_init\_\_(self, width, height):

self.width = width

self.height = height

def area(self):

return (self.height\*self.width)

def perimeter(self):

return (2\*(self.height\*self.width))

rect = Rectangle(5, 10)

print("Area:", rect.area())

print("Perimeter:", rect.perimeter())

2. Inheritance

Create a base class Animal and a subclass Dog. The subclass should have a method that overrides a method from the base class.

class Animal:

def speak(self):

return "Animal sound"

class Dog(Animal):

def speak(self):

return "Bark"

my\_dog = Dog()

print(my\_dog.speak()) # Should print "Bark"

3. Polymorphism

Create two classes Cat and Dog, both having a method speak. Demonstrate polymorphism by calling the method on both objects.

class Cat:

def speak(self):

return "Meow"

class Dog:

def speak(self):

return "Woof"

def animal\_sound(animal):

print(animal.speak())

my\_cat = Cat()

my\_dog = Dog()

animal\_sound(my\_cat) # Should print "Meow"

animal\_sound(my\_dog) # Should print "Woof"

5 Static Methods

Create a class MathUtils with a static method add that takes two numbers and returns their sum. class MathUtils:

class MathUtils:

@staticmethod

def add(a, b):

return a + b

print(MathUtils.add(5, 10)) # Should print 15