**Level 8: The Control Room / Advanced Vault**

**Classes and Objects**

**MCQs:**

1. What is a class in Python?  
   a) A function that returns an object  
   b) A blueprint for creating objects ✅  
   c) A built-in Python module  
   d) A method inside a function
2. What keyword is used to create a class?  
   a) object  
   b) define  
   c) class ✅  
   d) type

**Fill in the Blanks:**

1. An **object** is an instance of a \_\_\_\_\_\_\_\_\_\_. (class)
2. The \_\_init\_\_ method in a class is called the \_\_\_\_\_\_\_\_\_\_. (constructor)

**True/False:**

1. A class must always have a constructor method. (False ❌ – It's optional but recommended)

**Methods**

**MCQs:**

1. What is a **method** in Python?  
   a) A function inside a class ✅  
   b) A variable inside a class  
   c) A built-in function  
   d) A standalone function
2. How do you define a method inside a class?  
   a) def method(self): ✅  
   b) method(self):  
   c) define method(self):  
   d) class method(self):

**Fill in the Blanks:**

1. The **first parameter** of an instance method is always \_\_\_\_\_\_\_\_\_\_. (self)
2. The self parameter represents the \_\_\_\_\_\_\_\_\_\_ of the class. (instance)

**True/False:**

1. A class method can be called without creating an object. (False ❌ – Unless it's a @staticmethod)

**Attributes**

**MCQs:**

1. How are attributes defined inside a class?  
   a) Using the class keyword  
   b) Inside the \_\_init\_\_ method ✅  
   c) Using attribute = value outside methods  
   d) Both **b** and **c** ✅
2. What will be the output of the following?

class Student:

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

self.name = name

s1 = Student("Alex")

print(s1.name)

a) Alex ✅  
b) name  
c) Error  
d) None

**Fill in the Blanks:**

1. Instance attributes belong to a specific \_\_\_\_\_\_\_\_\_\_. (object)
2. Class attributes are shared among all \_\_\_\_\_\_\_\_\_\_ of a class. (instances)

**True/False:**

1. Instance attributes are defined outside the \_\_init\_\_ method. (False ❌ – They are defined inside \_\_init\_\_)

**Inheritance**

**MCQs:**

1. What is inheritance in Python?  
   a) A class acquiring properties of another class ✅  
   b) Copying variables from one function to another  
   c) Making private attributes public  
   d) Using super() in a function
2. Which syntax is used to inherit from a class A?  
   a) class B inherit A:  
   b) class B(A): ✅  
   c) class B -> A:  
   d) class B:A

**Fill in the Blanks:**

1. The super() function is used to call methods of the \_\_\_\_\_\_\_\_\_\_ class. (parent)
2. A child class can override methods of the \_\_\_\_\_\_\_\_\_\_ class. (parent/base)

**True/False:**

1. Python supports multiple inheritance. (True ✅)

**Polymorphism**

**MCQs:**

1. What is polymorphism in Python?  
   a) One function working differently for different inputs ✅  
   b) Creating multiple objects of the same class  
   c) Using the self keyword  
   d) None of the above
2. Which of the following is an example of method overloading in Python?  
   a) Defining multiple methods with the same name but different arguments ✅  
   b) Calling a method with different objects  
   c) Using super()  
   d) None

**Fill in the Blanks:**

1. **Overriding** occurs when a child class provides a new definition for a \_\_\_\_\_\_\_\_\_\_ method. (parent)
2. The + operator behaves differently with integers and strings because of \_\_\_\_\_\_\_\_\_\_. (operator overloading)

**True/False:**

1. Python allows method overloading in the same way as Java. (False ❌ – Python handles overloading differently using default parameters)