# python – Assessment Bank

Generated on: 2025-06-07 20:33:58

1. Consider a Python class designed for representing bank accounts. Which of the following design choices best demonstrates encapsulation and promotes data integrity, preventing direct access to the account balance while allowing controlled modification?  
  
 A. A public `balance` attribute directly accessible from outside the class.  
 B. A private `\_balance` attribute accessed only through public methods like `deposit()` and `withdraw()`.  
 C. Using a global variable to store the balance, accessible by any part of the program.  
 D. Storing the balance in a separate file, read and written to by the class methods.  
 \*\*(Answer: B)\*\*  
  
  
2. You have a `Dog` class with attributes `name` and `breed`. You create a subclass `GoldenRetriever` inheriting from `Dog`. Which statement accurately describes the relationship and method overriding in this scenario?  
  
 A. `GoldenRetriever` automatically inherits all methods from `Dog` and cannot override them.  
 B. `GoldenRetriever` inherits all attributes and methods, but cannot add new attributes or methods.  
 C. `GoldenRetriever` inherits attributes and methods, and can override existing methods or add new ones.  
 D. Inheritance is not possible in Python, so `GoldenRetriever` must be defined independently.  
 \*\*(Answer: C)\*\*  
  
  
3. Analyzing the concept of polymorphism, which scenario best exemplifies its practical application within a Python OOP context?  
  
 A. Creating multiple classes with the same name.  
 B. Using a single function to process objects of different classes with a common interface.  
 C. Restricting access to class attributes using underscores.  
 D. Implementing inheritance without overriding any methods.  
 \*\*(Answer: B)\*\*  
  
  
4. Given a scenario where you need to restrict the instantiation of a class directly, but allow its subclasses to be instantiated, what design pattern is most appropriate?  
  
 A. Using a private constructor.  
 B. Using a static method.  
 C. Making the class abstract.  
 D. Using a global variable to control instantiation.  
 \*\*(Answer: C)\*\*  
  
  
5. Which statement about the `\_\_init\_\_` method in a Python class is most accurate concerning its role in object creation and attribute initialization?  
  
 A. It's optional; if omitted, objects are created without attributes.  
 B. It's automatically called when an object is destroyed (garbage collection).  
 C. It's called when an object is created and is primarily used for initializing object attributes.  
 D. It's used for defining class-level attributes, not object attributes.  
 \*\*(Answer: C)\*\*  
  
  
6. Analyzing the differences between composition and inheritance, which statement best describes when composition is preferred?  
  
 A. When you need to create a "is-a" relationship between classes.  
 B. When you want to tightly couple classes, increasing dependency.  
 C. When you want to achieve flexibility and avoid tight coupling between classes, promoting reusability.  
 D. When you need a strong, inflexible hierarchical relationship between classes.  
 \*\*(Answer: C)\*\*  
  
  
7. How does the concept of abstraction contribute to improved code design within a Python OOP program, specifically concerning the management of complex systems?  
  
 A. It increases code complexity by introducing unnecessary layers.  
 B. It hides implementation details and exposes only essential information to the user, simplifying interaction.  
 C. It eliminates the need for classes and objects, making the code simpler.  
 D. It forces the programmer to expose all implementation details.  
 \*\*(Answer: B)\*\*  
  
  
8. Considering the principle of data hiding in OOP, what is the primary purpose of using double underscores (\_\_) before an attribute name (e.g., `\_\_private\_attribute`) in Python?  
  
 A. It makes the attribute faster to access.  
 B. It has no effect; it's purely stylistic.  
 C. It prevents accidental modification from outside the class, implementing name mangling.  
 D. It makes the attribute public, accessible from anywhere.  
 \*\*(Answer: C)\*\*  
  
  
9. A software system modeling different types of vehicles (cars, trucks, motorcycles) would best utilize which OOP concept to represent their common characteristics and unique attributes efficiently?  
  
 A. Polymorphism only.  
 B. Inheritance and Polymorphism.  
 C. Encapsulation only.  
 D. Abstraction only.  
 \*\*(Answer: B)\*\*  
  
  
10. In the context of exception handling and OOP design, how can you best ensure robustness and graceful handling of potential errors within methods of a class?  
  
 A. Ignoring potential exceptions and letting the program crash.  
 B. Including `try...except` blocks within class methods to catch and handle specific exceptions.  
 C. Using global error handling outside the class.  
 D. Re-raising exceptions without any specific handling.  
 \*\*(Answer: B)\*\*

---  
Watermark: TeachMate AI | Version 1.0