1. **What is a class in Python?**
    a. A variable that stores data.
    b. A collection of functions.
    c. [✓ A blueprint for creating objects.]
    d. An instance of an object.

2. **What is an object in Python?**
    a. A special type of function.
    b. [✓ An instance of a class.]
    c. A reserved keyword.
    d. A type of loop.

3. **What is inheritance in Python?**
    a. The process of creating a new class.
    b. [✓ The process of reusing code from an existing class.]
    c. The process of instantiating an object.
    d. The process of defining a class.

4. **In Python, if a class B inherits from class A, and both classes have a method with the same name, which method will be called when invoked on an object of class B?**
    a. Method of class A.
    b. [✓ Method of class B.]
    c. Both methods will be called simultaneously.
    d. It depends on the order of inheritance.

5. **What is the purpose of the __init__ method in Python classes?**
    a. [✓ To initialize a new object.]
    b. To define a new class.
    c. To create a new instance of a module.
    d. To execute the class methods.

6. **In Python, what is encapsulation?**
    a. The process of creating a new class.
    b. The process of reusing code from an existing class.
    c. The process of bundling data and methods that operate on the data within a single unit.
    d. The process of defining a class.

7. **Which of the following access modifiers in Python allows an attribute or method to be accessible only within its own class and any class derived from it?**
    a. Public
    b. Private
    c. Protected
    d. None of the above

8. **In Python, what is the purpose of the super() function when used in a derived class?**
    a. It refers to the superclass itself.
    b. It refers to the current class.
    c. It calls the constructor of the superclass.
    d. It creates a new instance of the superclass.

9. **What is method overloading in Python?**
    a. Defining multiple methods with the same name in a class.
    b. Defining multiple methods with different names in a class.
    c. The process of reusing code from an existing class.
    d. The process of creating a new class.

10. **Which statement is correct about multiple inheritance in Python?**
    a. Python does not support multiple inheritance.
    b. It is a process of creating multiple objects of a class.
    c. A class can inherit from more than one class.
    d. It is the process of creating a new class.

11. **Consider the following class definition:**

```
class Parent:
   def display(self):
      return "Parent class"

class Child(Parent):
   def display(self):
      return "Child class"
```

If you create an object “obj = Child()”, and then call “obj.display()”, what will be the output?

a. "Parent class"
[✓ b. "Child class"]
c. "Parent class Child class"
d. “Parent Child”

12. **Consider the following class definition:**

```
class Animal:
   def make_sound(self):
      return "Generic animal sound"

class Dog(Animal):
   def make_sound(self):
      return "Woof!"

class Cat(Animal):
   def make_sound(self):
      return "Meow!"
```

If you create an object “dog = Dog()”, and then call “dog.make_sound()”, what will be the output?

a. "Generic animal sound"
[✓ b. "Woof!"]
c. "Meow!"
d. “Me”

13. **Consider the following class definition:**

```
class Clothing:
   def wear(self):
      return "Generic clothing style"

class Shirt(Clothing):
   def wear(self):
      return "Shirt style"

class Pants(Clothing):
   def wear(self):
      return "Pants style"
```

If you create an object “clothing = Clothing()”, and then call “clothing.wear()”, what will be the output?

[✓ a. "Generic clothing style"]
b. "Shirt style"
c. "Pants style"
d. “Generic”

14. **Consider the following class definition:**

```
class Plant:
   def grow(self):
      return "Generic plant growth"

class Rose(Plant):
   def grow(self):
      return "Rose produces beautiful blooms"

class OakTree(Plant):
   def grow(self):
      return "Oak tree develops sturdy branches"
```

If you create an object flora = Rose(), and then call flora.grow(), what will be the output?

A. "Generic plant growth"

B. "Rose produces beautiful blooms"

C. "Oak tree develops sturdy branches"

D. Error

15. **Consider the following class definition:**

```
class Furniture:
   def use(self):
      return "Generic furniture usage"

class Chair(Furniture):
   def use(self):
      return "Chair provides seating comfort"

class Desk(Furniture):
   def use(self):
      return "Desk supports work and study"
```

If you create an object item = Desk(), and then call item.use(), what will be the output?

A. "Generic furniture usage"

B. "Chair provides seating comfort"

C. "Desk supports work and study"

D. Error

16. **Consider the following class definition:**

```
class KitchenAppliance:
    def operate(self):
        return "Generic appliance operation"

class Refrigerator(KitchenAppliance):
    def operate(self):
        return "Refrigerator cools and preserves food"

class MicrowaveOven(KitchenAppliance):
    def operate(self):
        return "Microwave oven heats and cooks food"
```

If you create an object appliance = MicrowaveOven(), and then call appliance.operate(), what will be the output?

A. "Generic appliance operation"

B. "Refrigerator cools and preserves food"

C. "Microwave oven heats and cooks food"

D. Error

17. **In Python, what does the term "polymorphism" refer to?**

A. The ability of a class to inherit from multiple classes.

B. The ability of a method to have multiple implementations.

C. The process of creating a new class.

D. The process of defining a class.

18. In Python, what is the purpose of using private attributes in a class?

A. To make the attributes accessible from outside the class.

B. To restrict access to the attributes within the class.

C. To allow a class to inherit from multiple classes.

D. To increase code complexity.

19. What is the role of access modifiers in encapsulation?

A. To allow unrestricted access to class members.

B. To restrict access to class members based on certain conditions.

C. To define the size of the window in a GUI application.

D. To determine the order of inheritance.

20. Consider the following class hierarchy:

```
class Animal:
    def make_sound(self):
        return "Generic animal sound"

class Dog(Animal):
    def make_sound(self):
        return "Woof!"

class Cat(Animal):
    def make_sound(self):
        return "Meow!"
```

If you want to add a new class, say *Bird*, which inherits from the Animal class, what advantage of OOPs does this demonstrate?

A. Code redundancy
B. Code complexity

C. Code reusability
D. Code obfuscation