Python OOP Assignment

Q1. What is the purpose of Python's OOP?

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.

Q2. Where does an inheritance search look for an attribute?

An inheritance search looks for an attribute first in the instance object, then in the class the instance was created from, then in all higher superclasses, progressing from left to right (by default). The search stops at the first place the attribute is found.

Q3. How do you distinguish between a class object and an instance object?

The class is like a blue-print & object is like an actual thing built on that blue-print. Instance is a virtual copy of the object. Thus the difference between a class & instance object.

Q4. What makes the first argument in a class’s method function special?

The calling process is automatic while the receiving process is not (its explicit). This is the reason the first parameter of a function in class must be the object itself. Writing this parameter as self is merely a convention.

Q5. What is the purpose of the __init__ method?

The __init__ method lets the class initialize the object's attributes and serves no other purpose. It is only used within classes.

Q6. What is the process for creating a class instance?

To create instances of a class, you call the class using class name and pass in whatever arguments its __init__ method accepts.

Q7. What is the process for creating a class?

Declaration − A variable declaration with a variable name with an object type. Instantiation − The 'new' keyword is used to create the object. Initialization − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Q8. How would you define the superclasses of a class?

A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class.

Q9. What is the relationship between classes and modules?

Modules are collections of methods and constants. They cannot generate instances. Classes may generate instances (objects), and have per-instance state (instance variables).

Q10. How do you make instances and classes?
To create instances of a class, you call the class using class name and pass in whatever arguments its __init__ method accepts.

Q11. Where and how should be class attributes created?

Q12. Where and how are instance attributes created?
Instance attributes are defined in the constructor. Defined directly inside a class. Defined inside a constructor using the self parameter.

Q13. What does the term "self" in a Python class mean?

Q14. How does a Python class handle operator overloading?
The operator overloading in Python means provide extended meaning beyond their predefined operational meaning. Such as, we use the "+" operator for adding two integers as well as joining two strings or merging two lists. We can achieve this as the "+" operator is overloaded by the "int" class and "str" class.

Q15. When do you consider allowing operator overloading of your classes?
Operator overloading is mostly useful when you're making a new class that falls into an existing "Abstract Base Class" (ABC) -- indeed, many of the ABCs in standard library module collections rely on the presence of certain special methods (and special methods, one with names starting and ending with double underscores

Q16. What is the most popular form of operator overloading?
A very popular and convenient example is the Addition (+) operator. Just think how the '+' operator operates on two numbers and the same operator operates on two strings. It performs “Addition” on numbers whereas it performs “Concatenation” on strings.
Q17. What are the two most important concepts to grasp in order to comprehend Python OOP code?

Q18. Describe three applications for exception processing.

Q19. What happens if you don’t do something extra to treat an exception?
If you don't handle exceptions
When an exception occurred, if you don't handle it, the program terminates abruptly and the code past the line that caused the exception will not get executed.

Q20. What are your options for recovering from an exception in your script?
You can also provide a generic except clause, which handles any exception

Q21. Describe two methods for triggering exceptions in your script.
To avoid such a scenario, there are two methods to handle Python exceptions: Try – This method catches the exceptions raised by the program. Raise – Triggers an exception manually using custom exceptions.

Q22. Identify two methods for specifying actions to be executed at termination time, regardless of
whether or not an exception exists.

Q23. What is the purpose of the try statement?
The try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.
Q24. What are the two most popular try statement variations?
There are two other optional segments to a try block: else and finally . Both of these optional blocks will come after the try and the except .

Q25. What is the purpose of the raise statement?
The RAISE statement stops normal execution of a PL/SQL block or subprogram and transfers control to an exception handler.

Q26. What does the assert statement do, and what other statement is it like?
The assert keyword is used when debugging code. The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError.

Q27. What is the purpose of the with/as argument, and what other statement is it like?
The with statement is a replacement for commonly used try/finally error-handling statements.

Q28. What are *args, **kwargs?
*args and **kwargs allow you to pass multiple arguments or keyword arguments to a function.

Q29. How can I pass optional or keyword parameters from one function to another?
Users can either pass their values or can pretend the function to use theirs default values which are specified. In this way, the user can call the function by either passing those optional parameters or just passing the required parameters. Without using keyword arguments. By using keyword arguments.

Q30. What are Lambda Functions?
Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging.

Q31. Explain Inheritance in Python with an example?
Inheritance relationship defines the classes that inherit from other classes as derived, subclass, or sub-type classes. For example, you have a Base class of “Animal,” and a “Lion” is a Derived class. The inheritance will be Lion is an Animal.

Q32. Suppose class C inherits from classes A and B as class C(A,B).Classes A and B both have their own versions of method func(). If we call func() from an object of class C, which version gets invoked?
C(A) would be invoked.

Q33. Which methods/functions do we use to determine the type of instance and inheritance?
Use isinstance() to check an instance's type: isinstance(obj, int) will be True only if obj.__class__ is int or some class derived from int .
Use issubclass() to check class inheritance: issubclass(bool, int) is True since bool is a subclass of int .

Q34.Explain the use of the 'nonlocal' keyword in Python.
The nonlocal keyword is used to work with variables inside nested functions, where the variable should not belong to the inner function. Use the keyword nonlocal to declare that the variable is not local.

Q35. What is the global keyword?
The global keyword is used to create global variables from a no-global scope, e.g. inside a function.