**Python OOP Assignment**

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

The Purpose of OOP is to implement real world paradigm like inheritance, polymorphism, encapsulation etc.

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 super classes, 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 a the blue print. The Object is an actual thing that is built based on the 'blue print' (like the house). An instance is a virtual copy (but not a real copy) of the object.

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

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. It is not a keyword and has no special meaning in Python.

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?

To define a class, use the class keyword, then give your class a name. Then inside curly brackets {}, write the **body** of the class. The body consists of the variables, constructors, and functions you want inside the class.

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. In the above example, Vehicle is the Superclass and its subclasses are Car, Truck and Motorcycle

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?

Class attributes are the variables defined directly in the class that are shared by all objects of the class. Instance attributes are attributes or properties attached to an instance of a class. Instance attributes are defined in the constructor. Defined directly inside a class.

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?

self **represents the instance of the class**. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments. The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes.

Q14. How does a Python class handle operator overloading?

To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator, the magic method \_\_add\_\_ is automatically invoked in which the operation for + operator is defined.

Q15. When do you consider allowing operator overloading of your classes?

Consider that we have two objects which are a physical representation of a class (user-defined data type) and we have to add two objects with binary '+' operator it throws an error, because compiler don't know how to add two objects. So we define a method for an operator and that process is called operator overloading.

Q16. What is the most popular form of operator overloading?

A very popular and convenient example is the Addition (+) operator

Q17. What are the two most important concepts to grasp in order to comprehend Python OOP code?

In this article, we will elaborate on two key concepts of OOP which are inheritance and polymorphism. Both inheritance and polymorphism are key ingredients for designing robust, flexible, and easy-to-maintain software. These concepts are best explained via examples.

Q18. Describe three applications for exception processing.

Zero Division Error, Floating Point Error, Overflow Error

Q19. What happens if you don't do something extra to treat an exception?

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. After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception. The else-block is a good place for code that does not need the try: block's protection.

Q21. Describe two methods for triggering exceptions in your script.

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.

* Division by Zero
* Accessing a file which does not exist.

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.

Q24. What are the two most popular try statement variations?

What are the two most popular try statement variations in Python?

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 . Also, there's nothing stopping you from using both else and finally in a single statement — but keep them in that order if you do

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. RAISE statements can raise predefined exceptions, such as ZERO\_DIVIDE or NO\_DATA\_FOUND , or user-defined exceptions whose names you decide.

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. You can write a message to be written if the code returns False, check the example below.

Q27. What is the purpose of the with/as argument, and what other statement is it like?

In Python, with statement is used in exception handling to make the code cleaner and much more readable. It simplifies the management of common resources like file streams

Q28. What are \*args, \*\*kwargs?

\*args and \*\*kwargs allow you to pass multiple arguments or keyword arguments to a function. Consider the following example. This is a simple function that takes two arguments and returns their sum: def my\_sum(a, b): return a + b. This function works fine, but it's limited to only two arguments.

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?

A lambda function is an anonymous function (i.e., defined without a name) that can take any number of arguments but, unlike normal functions, evaluates and returns only one expression.

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. Base class remains to be the source from which a subclass inherits. 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?

Class A as that order comes first

Q33. Which methods/functions do we use to determine the type of instance and inheritance?

**isinstance() and issubclass()** The isinstance() method checks whether an object is an instance of a class whereas issubclass() method asks whether one class is a subclass of another class (or other classes).

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?

Global keyword is used when we want to read or write any global variable value inside the function.