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?

From an pre-existing classes, inheritance searches look for an attribute

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

A class is a template for creating objects in a program, whereas the object is an instance of a class**.** A class is a logical entity, while an object is a physical entity. A class does not allocate memory space; on the other hand, an object allocates memory

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

Q5. What is the purpose of the init method?

The __init__ method allows you to accept arguments to your class.

More importantly, the __init__ method allows you to assign initial values to various attributes on your class instances.

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?

A class can be created by using the keyword class, followed by the class name.

Syntax: **class** ClassName:

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).

Modules may be mixed in to classes and other modules. The mixed in module's constants and methods blend into that class's own, augmenting the class's functionality. Classes, however, cannot be mixed in to anything.

A class may inherit from another class, but not from a module.

A module may not inherit from anything.

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?

A class attribute is shared by all instances of the class. To define a class attribute, you place it outside of the __init__() method.

Q12. Where and how are instance attributes created?

Instance attributes are defined in the __init__() function.

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?

Python operators work for built-in classes**.** But the same operator behaves differently with different types. For example, the + operator will perform arithmetic addition on two numbers, merge two lists, or concatenate two strings.

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

Operator Overloading means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because '+' operator is overloaded by int class and str class. You might have noticed that the same built-in operator or function shows different behavior for objects of different classes, this is called *Operator Overloading*.

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?

Both **inheritance** and **polymorphism** are key ingredients for designing robust, flexible, and easy-to-maintain software.

Q18. Describe three applications for exception processing.

Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

ry and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

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

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

Generally, an array is of fixed size and each element is accessed using the indices. For example, we have created an array with size 7. Then the valid expressions to access the elements of this array will be a[0] to a[6] (length-1).

Whenever, you used an –ve value or, the value greater than or equal to the size of the array, then the ArrayIndexOutOfBoundsException is thrown.

Q20. What are your options for recovering from an exception in your script?

In Python, exceptions can be handled using a try statement. The critical operation which can raise an exception is placed inside the try clause. The code that handles the exceptions is written in the except clause. We can thus choose what operations to perform once we have caught the exception.

Q21. Describe two methods for triggering exception

Try – This method catches the exceptions raised by the program. Raise – Triggers an exception manually using custom exceptions in your script.

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

We can specify which exceptions an except clause should catch. A try clause can have any number of except clauses to handle different exceptions, however, only one will be executed in case an exception occurs. We can use a tuple of values to specify multiple exceptions in an except clause.

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?

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

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?

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.

Q28. What are *args, **kwargs?

*args specifies the number of non-keyworded arguments that can be passed and the operations that can be performed on the function in Python whereas **kwargs is a variable number of keyworded arguments that can be passed to a function that can perform dictionary operations.

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.

Q30. What are Lambda Functions?

A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have 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?

Python also has a super() function that will make the child class inherit all the methods and properties from its parent

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?

Global keyword is **used to modify the global variable outside its current scope and meaning**. It is used to make changes in the global variable in a local context. The keyword 'Global' is also used to create or declare a global variable inside a function.