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).

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 space.

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?

"\_\_init\_\_" is a reseved method in python classes. It is known as a constructor in object oriented concepts. This method called when an object is created from the class and it allow the class to initialize the attributes of a class

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

To create a class that inherits from another class, after the class name you'll put parentheses and then list any classes that your class inherits from. In a function definition, parentheses after the function name represent arguments that the function accepts.

Q7. What is the process for creating a class?

In Python, a class can be created by using the keyword class, followed by the class name.

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?

So a module in python is simply a way to organize the code, and it contains either python classes or just functions. If you need those classes or functions in your project, you just import them. For instance, the math module in python contains just a bunch of functions, and you just call those needed

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?

Instance attributes are defined in the \_\_init\_\_() function.

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?

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?

This feature in Python that allows the same operator to have different meaning according to the context 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 fundamental concepts of object oriented programming. These concepts help us to create code that can be extended and easily maintainable.

Q18. Describe three applications for exception processing.

There are mainly three kinds of distinguishable errors in Python: syntax errors, exceptions and logical errors

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

An exception object is created when a Python script raises an exception. If the script explicitly doesn't handle the exception, the program will be forced to terminate abruptly.

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.

An unhandled exception displays an error message and the program suddenly crashes. To avoid such a scenario, there are two methods to handle Python exceptions:

1. Try – This method catches the exceptions raised by the program
2. 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.

1. try
2. catch
3. finally
4. throw

Q23. What is the purpose of the try statement?

The try block lets you test a block of code for errors. The except block lets you handle the error. The else block lets you execute code when there is no error.

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

The Different Try/Except Variations. So far we've used a try / except and even a try / except / except , but this is only two-thirds of the story. 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?

Python raise Keyword is used to raise exceptions or errors. The raise keyword raises an error and stops the control flow of the program. It is used to bring up the current exception in an exception handler so that it can be handled further up the call stack.

Q26. What does the assert statement do, and what other statement is it like?

assert statement takes an expression and optional message. assert statement is used to check types, values of argument and the output of the function. assert statement is used as debugging tool as it halts the program at the point where an error occurs.

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

n 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 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. By using keyword arguments

Q30. What are Lambda Functions?

A lambda function evaluates an expression for a given argument. You give the function a value (argument) and then provide the operation (expression). The keyword lambda must come first. A full colon (:) separates the argument and the expression. In the example code below, x is the argument and x+x is the 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?

Mostly Class A

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

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?

In Python, the global keyword allows us to modify the variable outside of the current scope. It is used to create a global variable and make changes to the variable in a local context.