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

Ans: 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?**

**Ans:** 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?**

**Ans:** The class is 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?**

**Ans:**

**Q5. What is the purpose of the init method?**

**Ans:** 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?**

**Ans:** 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?**

**Ans:**

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

**Ans:** A superclass is the class from which many subclasses can be created.

**Q9. What is the relationship between classes and modules?**

**Ans:** The difference between a class and a module is that a class is used to define a blueprint for a given object, whereas a module is used to reuse a given piece of code inside another program.

**Q10. How do you make instances and classes?**

**Ans:** 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?**

**Ans:** Class attributes belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top, for legibility.

**Q12. Where and how are instance attributes created?**

**Ans;**  Instance attributes, which are defined in the \_\_init\_\_() function, are class variables that allow us to define different values for each object of a class. 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?**

**Ans:** The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

**Q14. How does a Python class handle operator overloading?**

**Ans**: In Python, overloading is achieved by overriding the method which is specifically for that operator, in the user-defined class. Such as, we use the "+" operator for adding two integers as well as joining two strings or merging two lists.

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

**Ans:** Ensures that objects of a class behave consistently with built-in types and other user-defined types.

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

**Ans:** The most frequent instance is the adding up operator '+', where it can be used for the usual addition and also for combining two different strings.

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

**Ans:** 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.**

**Ans: 1)TypeError**: This exception is raised when an operation or function is applied to an object of the wrong type, such as adding a string to an integer.

**2) NameError**: This exception is raised when a variable or function name is not found in the current scope.

**3) IndexError**: This exception is raised when an index is out of range for a list, tuple, or other sequence types.

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

**Ans:** 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?**

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

**Ans:** The two methods for triggering exceptions in your script are:

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?**

**Ans**: 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?**

**Ans:** There are two other optional segments to a try block: **else and finally**

**Q25. What is the purpose of the raise statement?**

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

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

**Ans: 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. Observe the following code example on how the use of with statement makes code cleaner.

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

**Ans:** \*args, \*\*kwargs are Special Symbols Used for passing arguments:-

* \*args (Non-Keyword Arguments)
* \*\*kwargs (Keyword Arguments)

**Q29. How can I pass optional or keyword parameters from one function to another?**

**Ans**: We can pass optional or keyword parameters from one function to anotherby using keyword arguments.

**Q30. What are Lambda Functions?**

**Ans:** lambda is a keyword in Python for defining the anonymous function. Lambda helps you use a function only once, and hence, avoids cluttering up the code with function definitions.

**Q31. Explain Inheritance in Python with an example?**

**Ans:** 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.

**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?**

**Ans:**

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

**Ans:** Python has two built-in functions that work with 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.**

**Ans:** The nonlocal keyword is used to work with variables inside nested functions, where the variable should not belong to the inner function.

**Q35. What is the global keyword?**

**Ans:**  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.