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

A1. Object Oriented Programming (OOP) is a programming paradigm that uses object and classes with attributes and methods in programming. Its goal is to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.

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

A2. The 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?

A3. When we create a class in python then a class object is created so whenever python finds a class statement in the whole program then it creates a class object and assigns a name to that object i.e. class name whereas when we call a class, it creates an instance object of that class from which the object has been created.

Example: Class- class Fruit:

  pass

Instance- Apple= Fruit()

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

A4. The self keyword is used to represent an instance (object) of the given class. The first parameter of a function in class must be the object itself. By using the self we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

Q5. What is the purpose of the init method?

A5. 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?

A6. 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?

A7. To create a class, the keyword class needs to be used. Then different methods are put into it and also use the \_\_init\_\_() function to assign values to object properties.

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

A8. The class from which a class inherits is called the parent or superclass The class from which the subclass is derived is called a superclass (also a base class or a parent class). It is a part of the inheritance process.

Q9. What is the relationship between classes and modules?

A9. Classes may generate instances (objects), and have per-instance state (instance variables). 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. Modules are collections of methods and constants. They cannot generate instances.

Q10. How do you make instances and classes?

A10. To create instances of a class, you call the class using class name and pass in whatever arguments its \_\_init\_\_ method accepts. While to create a class, the keyword class needs to be used. Then different methods are put into it and also use the \_\_init\_\_() function to assign values to object properties.

Example: Class- class Fruit:

  Pass

Instance- Apple= Fruit()

Q11. Where and how should be class attributes created?

A11. Class attributes are the variables defined directly in the class that are shared by all objects of the class.

Q12. Where and how are instance attributes created?

A12. 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.

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

A13. 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?

A14. In Python class, we can make the derived class have the same methods as base class by making certain changes. When an instance is created for the derived class and we call the method that is common to both derived and base class, the appropriate method gets called. That’s how operator overloading is handled in Python class.

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

A15. Operator overloading is mostly useful when you're making a new class that falls into an existing base class, many of them in standard library module collections rely on the presence of certain special methods.

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

A16. 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?

A17. Inheritance and encapsulation are the two most important concepts to grasp in order to comprehend Python OOP code.

Q18. Describe three applications for exception processing.

A18. The three applications for exception processing are:

1. ZeroDivisionError – It is raised when division or modulo by zero takes place for all numeric types
2. IndexError – It is raised when an index is not found in a sequence.
3. KeyError – It is raised when the specified key is not found in the dictionary.

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

A19. If you don’t handle 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?

A20. Exceptions can be recovered by handling them with the help of try-catch blocks.

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

A21. The two methods for triggering exceptions in the script are:

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.

A22. The two methods for specifying actions to be executed at termination time are: Try and Finally.

Q23. What is the purpose of the try statement?

A23. 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?

A24. The two most popular try statement variations are:

1. try:  
     print(x)  
   except:  
     print("An exception occurred")
2. try:  
     print("Hello")  
   except:  
     print("Something went wrong")  
   else:  
     print("Nothing went wrong")

Q25. What is the purpose of the raise statement?

A25. 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?

A26. 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?

A27. 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. It replaces a try-catch block with a concise shorthand.

The with statement is a replacement for commonly used try/finally error-handling statements. A common example of using the with statement is opening a file.

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

A28. The \*args and \*\*kargs are special symbols for passing arguments. The special syntax *\*args* in function definitions in python is used to pass a variable number of arguments to a function. It is used to pass a non-key worded, variable-length argument list. The special syntax *\*\*kwargs* in function definitions in python is used to pass a keyworded, variable-length argument list. We use the name *kwargs* with the double star. The reason is that the double star allows us to pass through keyword arguments (and any number of them).

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

A29. To pass optional or keyword parameters from one function to another, collect the arguments using the \* and \*\* specifiers in the function’s parameter list.

Q30. What are Lambda Functions?

A30. Lambda is a keyword in Python for defining the anonymous function. Argument(s) is a placeholder, that is a variable that will be used to hold the value you want to pass into the function expression. A lambda function can have multiple variables depending on what you want to achieve.

Q31. Explain Inheritance in Python with an example?

A31. Inheritance in Python is one of the core concepts in object-oriented programming (OOP) languages. It is a mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class. Inheritance is the capability of one class to derive or inherit the properties from another class.

Example:

**class** Person(object):

**def** \_\_init\_\_(self, name):

        self.name **=** name

**def** getName(self):

**return** self.name

**def** isEmployee(self):

**return** False

**class** Employee(Person):

**def** isEmployee(self):

**return** True

emp **=** Person("Anita")  # An Object of Person

print(emp.getName(), emp.isEmployee())

emp **=** Employee("Shubhi")  # An Object of Employee

print(emp.getName(), emp.isEmployee())

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?

A32. The version of method func() from Class A gets invoked as it is placed before B in C(A,B).

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

* A33.
* isinstance() is used to check an instance's type: isinstance(obj, int) will be True only if obj.\_\_class\_\_ is int or some class derived from int .
* issubclass() is used 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.

A34. The nonlocal keyword is used to work with variables inside nested functions, where the variable should not belong to the inner function. It is used to declare that the variable is not local.

Q35. What is the global keyword?

A35. The global keyword is used to create global variables from a no-global scope, e.g. inside a function.