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

* **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**

**Main Concepts of Object-Oriented Programming (OOPs)**

* Class
* Objects
* Polymorphism
* Encapsulation
* Inheritance
* Data Abstraction

**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?**

* class object:
* 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. As we know in python, everything is an object so the class itself is an object and is the instance of [metaclasses](https://artificialintelligencestechnology.com/python/metaclasses-in-python/). Look at the following example

|  |  |
| --- | --- |
|  | class MyClass:   pass |

* above code will generate a class object and name it ‘MyClass’. From this class object, we will create **instance objects**.

## Instance object:

when we **call**a class, it creates an instance object of that class from which the object has been created. For example when we call the above-created class then it will create an instance object like this.

|  |  |
| --- | --- |
| 1 | Obj1=MyClass() |
|  |  |

the above statement creates an object and names it to **Obj1**which is an instance of MyClass.

**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 create a class, use the keyword class:**

Create a class named MyClass, with a property named x:

class MyClass:  
  x = 5

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

* **The class whose properties gets inherited by another class is known as superclass or parent class and the class which inherits the properties of another class is known as the subclass.**

**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 (math.sin). whereas Classes are blueprints that allow you to create instances with attributes and bound functionality. Classes support inheritance, metaclasses, and descriptors.**

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

* **The *class* statement creates a new class definition. The name of the class immediately follows the keyword *class* followed by a colon as follows –**

**Class classname:**

* **To create instances of a class, you call the class using class name and pass in whatever arguments its *\_\_init\_\_* method accepts.**

**Emp1=Employee(“Rahul”,25)**

**Q11. Where and how should be class attributes created?**

* To define a class attribute, you place it outside of the \_\_init\_\_() method. For example, the following defines pi as a class attribute:
* class Circle:
* pi = 3.14159
* def \_\_init\_\_(self, radius):
* self.radius = radius
* def area(self):
* return self.pi \* self.radius\*\*2
* def circumference(self):
* return 2 \* self.pi \* self.radius

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

* **An instance attribute is a Python variable belonging to one, and only one, object. This variable is only accessible in the scope of this object, and it’s defined inside the constructor function**

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

* **After creating a class in Python, we need to access its attributes and methods. In Python, the word self is explicitly included as a first parameter of methods.**

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

* **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*.**
* **# Python program to show use of**
* **# + operator for different purposes.**
* **print(1 + 2)**
* **# concatenate two strings**
* **print("rahul "+"jain")**
* **# Product two numbers**
* **print(3 \* 4)**
* **# Repeat the String**
* **print("rahul"\*4)**

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

* **Operator overloading is mostly useful when you're making a new class that falls into an existing "Abstract Base Class" (ABC) -- indeed, many of the ABCs in standard library module**[**collections**](http://docs.python.org/library/collections.html#module-collections)**rely on the presence of certain special methods (and special methods, one with names starting and ending with double underscores**

**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. These concepts are best explained via examples**

**Q18. Describe three applications for exception processing.**

**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?**

* **In Python, we catch exceptions and handle them using try and except code blocks. The try clause contains the code that can raise an exception, while the except clause contains the code lines that handle the exception. Let's see if we can access the index from the array, which is more than the array's length, and handle the resulting exception.**

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

* **To avoid such a scenario, there are two methods to handle Python exceptions: 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.**

* **Finally block always executes irrespective of an exception being thrown or not. The final keyword allows you to create a block of code that follows a try-catch block.**

**Q23. What is the purpose of the try statement?**

**The try block lets you test a block of code for errors.** **The try block is used to check some code for errors i.e the code inside the try block we execute when there is no error in the program.**

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

* **The raise keyword is used to raise an exception.**
* **You can define what kind of error to raise, and the text to print to the user**.
* x = "hello"  
    
  if not type(x) is int:  
    raise TypeError("Only integers are allowed")

**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.**

**x = "hello"**

**#if condition returns False, AssertionError is raised:**

**assert x == "goodbye", "x should be 'hello'"**

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

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

* **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?**

* **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?**

Lambda Functions in Python are anonymous functions, implying they don't have a name. The def keyword is needed to create a typical function in Python, as we already know. We can also use the lambda keyword in Python to define an unnamed function.

### Syntax of Python Lambda Function

* **lambda** arguments: expression

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

* **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.**
* Class BaseClass:
* {Body}
* Class DerivedClass(BaseClass):
* {Body}

**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 version will be invoked**

**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?**

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

**def myfunction():  
  global x  
  x = "hello"  
  
#execute the function:  
myfunction()  
  
#x should now be global, and accessible in the global scope.  
print(x)**