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

Ans-  **provides a means of structuring programs so that properties and behaviors are bundled into individual objects**.

Reuse of code through inheritance.

Flexibility through polymorphism. ..

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

Ans- **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- **A class is a blueprint which you use to create objects.** **An object is an instance of a class** - it's a concrete 'thing' that you made using a specific class. So, 'object' and 'instance' are the same thing, but the word 'instance' indicates the relationship of an object to its class.

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

Ans- 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?

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?

class Employee:

    def \_\_init\_\_(self,name,age):

        self.emp\_name=name

        self.emp\_age=age

    def displayEmp\_details(self):

        print("employee name: ",self.emp\_name,"age: ",self.emp\_age)

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

Ans- class Employee:

    def \_\_init\_\_(self,name,age):

        self.emp\_name=name

        self.emp\_age=age

    def displayEmp\_details(self):

        print("employee name: ",self.emp\_name,"age: ",self.emp\_age)

class Person(Employee):

    p1=Employee('prakhar', 24)

    p1.displayEmp\_details()

Q9. What is the relationship between classes and modules?

Ans- **a module in python is simply a way to organize the code, and it contains either python classes or just functions**.

Q10. How do you make instances and classes?

class Employee:

    def \_\_init\_\_(self,name,age):

        self.emp\_name=name

        self.emp\_age=age

Q11. Where and how should be class attributes created?

class Employee:

    def \_\_init\_\_(self,name,age):

        self.emp\_name=name

        self.emp\_age=age

name and age are class variable.

Q12. Where and how are instance attributes created?

Ans- class Employee:

    def \_\_init\_\_(self,name,age):

        self.emp\_name=name

        self.emp\_age=age

emp\_name,Emp\_age is instance attribute.

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

Ans-it acts as pointer point towards current object.

Q14. How does a Python class handle operator overloading?

Ans-

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

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

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

Q18. Describe three applications for exception processing.

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

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

Q21. Describe two methods for triggering 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.

Q23. What is the purpose of the try statement?

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

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?

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

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

Q30. What are Lambda Functions?

Q31. Explain Inheritance in Python with an example?

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?

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

Q34.Explain the use of the 'nonlocal' keyword in Python.

Q35. What is the global keyword?