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

- ➔ It's used to implement real life entities in the form of program.

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

- ➔ Inheritance searches first look for instance objects, then the class from which the instance was created, and then super classes, going from bottom to top and left to right.

Q3. How do you distinguish between a class object and an instance object?

- ➔ Class object are created when we define a class with keyword class. example -> class iNeuron:
- ➔ Instance object are created when we want to call the class. example -> ob = iNeuron()

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

- ➔ Self is the first argument in a class method.
- ➔ It is special because it represents the instance of the class.
- ➔ It's not compulsory to use self only we can use different names also like (this) but this is not universally followed.
- ➔ Whenever we pass an value from object it get converted internally and become class.method_name(ob,arg) so here self is used to store the object of the class.

Q5. What is the purpose of the init method?

- ➔ Init method is called when we create an instance of class and used to initialize value to the variable.

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

- ➔ Create a class object and passing the values that are there in __init__ method.

Q7. What is the process for creating a class?

- ➔ My using class keyword we can create a class followed by name of the class. example -> class iNeuron:

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

- ➔ Super classes are the class which is called parent class which will be inherited by the child class known sub-classes.

Q9. What is the relationship between classes and modules?

- ➔ A module in python contains either python classes or functions. If we need them, we can import them into our programs.

Q10. How do you make instances and classes?

- ➔ To make instances we need to call class and pass the values there in __init__ method.
- ➔ Classes are created using keyword class followed by class name and semicolon.

Q11. Where and how should be class attributes created?

- ➔ Class attributes are global variable defined inside a class and shared by all the objects.
- ➔ class iNeuron:

```
        count = 0 // this is class attributes.
```

Q12. Where and how are instance attributes created?

- ➔ Instance are defined inside a method and can accessed using object.

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

- ➔ Self is the first argument that is explicitly defined in a method.
- ➔ In __init__ method "self "works as a newly created object.
- ➔ And in other method it refers to the object whose method is called.

Q14. How does a Python class handle operator overloading?

- ➔ By using + operator we can handle operator overloading.

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?

- ➔ IN OOP inheritance and polymorphism. Both inheritance and polymorphism are key ingredients for designing robust, flexible, and easy-to-maintain software.

Q18. Describe three applications for exception processing.

- ➔ 1. Exception

  Base class for all exceptions

  2. StopIteration

  Raised when the next() method of an iterator does not point to any object.

  3. SystemExit

  Raised by the sys.exit() function.

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?

➔ 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. The finally block lets you execute code, regardless of the result of the try- and except blocks.

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

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?

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

➔ 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.

```
  # file handling
1. without using with statement
  file = open('file_path', 'w')
  file.write('hello world!')
  file.close()

2. without using with statement
  file = open('file_path', 'w')
  try:
     file.write('hello world')
  finally:
     file.close()
```

```
# using with statement

with open('file_path', 'w') as file:
file.write('hello world !')
```

Q28. What are *args,* *kwargs?

➔ When you are not clear how many arguments you need to pass to a particular function, then we use *args and* *kwargs.

The *args keyword represents a varied number of arguments. It is used to add together the values of multiple arguments

The kwargs keyword represents an arbitrary number of arguments that are passed to a function**.** kwargs keywords are stored in a dictionary. You can access each item by referring to the keyword you associated with an argument when you passed the argument.

args Python Example:

```
def sum(*args):

 total = 0

  for a in args:

   total = total + a print(total)

sum(1,2,3,4,5)

Output: 15
```

**kwargs Python Example:

```
def show(**kwargs):

        print(kwargs)

show(A=1,B=2,C=3)

Output: {'A': 1, 'B': 2, 'C': 3}
```

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

- ➔ we can pass a functional key parameter by using *args and* *kwargs specifiers.

Q30. What are Lambda Functions?

- ➔ Python Lambda Functions are anonymous function means that the function is without a name. As we already know that the def keyword is used to define a normal function in Python. Similarly, the lambda keyword is used to define an anonymous function in Python.

Q31. Explain Inheritance in Python with an example?

- ➔ Inheritance is a powerful feature in object-oriented programming.

It refers to defining a new class with little or no modification to an existing class. The new class is called derived (or child) class and the one from which it inherits is called the base (or parent) class.

```
class Person(object):

  # Constructor
  def __init__(self, name, id):
    self.name = name
    self.id = id

  # To check if this person is an employee(method)
  def display(self):
    print(self.name, self.id)


# object
emp = Person("iNeuron", 101) # An Object of Person
emp.display()
iNeuron 101
```

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?

- ➔ 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.

- ➔ 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.

```
def myfunc1():
  x = "John"
  def myfunc2():
    x = "hello"
  myfunc2()
  return x

print(myfunc1())
```

```
John
```

Q35. What is the global keyword?

- ➔ The global keyword is used to create global variables from a no-global scope.
  Global keyword is used to modify the global variable outside its current scope and meaning. It is used to make changes in the global variable in a local context. The keyword 'Global' is also used to create or declare a global variable 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)
```

```
hello
```