Q1. What is the purpose of PythonOOP?

A:

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. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

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

A: 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?

A:

|  |  |
| --- | --- |
| **Class** | **Object** |
| A class is a template for creating objects in program. | The object is an instance of a class. |
| A class is a logical entity | Object is a physical entity |
| A class does not allocate memory space when it is created. | Object allocates memory space whenever they are created. |

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

A:

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?

A: "\_\_init\_\_" is a reserved method in python classes. It is called as a constructor in object-oriented terminology. This method is called when an object is created from a class, and it allows the class to initialize the attributes of the class.

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

A: class Person:

# init method or constructor

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

self.name = name

# Sample Method

def say\_hi(self):

print('Hello, my name is', self.name)

p = Person('Nikhil')

p.say\_hi()

Q7. What is the process for creating a class?

A:

class name:

print("hello")

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

A: class name:

def func(self):

print(" I am part of class name")

class name1(name):

def func1(self):

print(" I am part of class name1")

a=name1()

a.func1()

a.func()