Name – Krishna Gupta

Phone number- 9993153109

ANSWER

“  
  
  
Here’s a \*\*complete list of Python OOPs questions with brief answers\*\* for quick revision or interview prep:

---

### 🔹 \*\*Basics\*\*

\*\*MH What is Object-Oriented Programming (OOP)?\*\*

OOP is a programming paradigm based on the concept of "objects", which contain both data (attributes) and methods (functions). It promotes code reusability, modularity, and abstraction.

\*\*H What is a class in OOP?\*\*

A class is a blueprint for creating objects. It defines attributes and methods that the created objects (instances) will have.

\*\*3H What is an object in OOP?\*\*

An object is an instance of a class. It is a self-contained unit with attributes and behavior defined by its class.

---

### 🔹 \*\*Core Concepts\*\*

\*\*\&H What is the difference between abstraction and encapsulation?\*\*

\* \*\*Abstraction\*\* hides complex implementation and shows only essential features.

\* \*\*Encapsulation\*\* binds data and methods together and hides the internal state from outside access.

\*\*\$H What are dunder methods in Python?\*\*

Dunder (double underscore) methods like `\_\_init\_\_`, `\_\_str\_\_`, `\_\_len\_\_` are special methods with double underscores. They provide built-in behavior for operators and functions.

\*\*"H Explain the concept of inheritance in OOP.\*\*

Inheritance allows a class (child) to inherit attributes and methods from another class (parent), promoting code reuse.

\*\*#H What is polymorphism in OOP?\*\*

Polymorphism allows methods to perform differently based on the calling object. In Python, it works via method overriding and duck typing.

---

### 🔹 \*\*Implementation in Python\*\*

\*\*9H How is encapsulation achieved in Python?\*\*

Encapsulation is achieved using access modifiers:

\* `\_protected` for protected

\* `\_\_private` for private variables (name mangling)

\*\*>H What is a constructor in Python?\*\*

A constructor is defined by `\_\_init\_\_()`. It runs automatically when a new object is created.

\*\*M H What are class and static methods in Python?\*\*

\* `@classmethod`: Method that takes `cls` as the first argument, operates on class variables.

\* `@staticmethod`: Doesn’t take `self` or `cls`, behaves like a regular function inside a class.

\*\*MMH What is method overloading in Python?\*\*

Python doesn't support traditional overloading. Instead, you can use default arguments or `\*args`/`\*\*kwargs` to simulate it.

\*\*M(H What is method overriding in OOP?\*\*

It allows a child class to provide a specific implementation of a method already defined in its parent class.

\*\*M3H What is a property decorator in Python?\*\*

`@property` lets you define a method that acts like an attribute. Used for getter/setter without breaking encapsulation.

---

### 🔹 \*\*Deep Concepts\*\*

\*\*M\&H Why is polymorphism important in OOP?\*\*

It enables flexibility and extensibility in code. Same interface, different implementations.

\*\*M\$H What is an abstract class in Python?\*\*

An abstract class (from `abc` module) cannot be instantiated directly and may contain abstract methods that must be implemented by subclasses.

\*\*M"H What are the advantages of OOP?\*\*

\* Code reusability

\* Modularity

\* Maintainability

\* Real-world modeling

\* Easier debugging and scaling

\*\*M#H What is the difference between a class variable and an instance variable?\*\*

\* \*\*Class variable\*\*: Shared across all instances (`ClassName.var`).

\* \*\*Instance variable\*\*: Unique to each object (`self.var`).

\*\*M9H What is multiple inheritance in Python?\*\*

A class can inherit from multiple base classes:

```python

class A: pass

class B: pass

class C(A, B): pass

```

---

### 🔹 \*\*Special Methods & Keywords\*\*

\*\*M>H Explain the purpose of `\_\_str\_\_` and `\_\_repr\_\_` methods in Python.\*\*

\* `\_\_str\_\_`: Returns a readable string for users (`print(obj)`).

\* `\_\_repr\_\_`: Returns a developer-oriented representation (`repr(obj)`).

\*\*( H What is the significance of the `super()` function in Python?\*\*

`super()` gives access to methods of the parent class. Useful in inheritance, especially with multiple inheritance.

\*\*(MH What is the significance of the `\_\_del\_\_` method in Python?\*\*

It’s the destructor method, called when an object is deleted or goes out of scope. Used for cleanup actions.

\*\*((H What is the difference between `@staticmethod` and `@classmethod` in Python?\*\*

\* `@staticmethod`: No access to class or instance.

\* `@classmethod`: Access to the class, but not the instance.

---

### 🔹 \*\*Advanced Behavior\*\*

\*\*(3H How does polymorphism work in Python with inheritance?\*\*

Through method overriding. Python checks the object’s class at runtime and calls the appropriate method.

\*\*(\&H What is method chaining in Python OOP?\*\*

Calling multiple methods on the same object in a single line. Each method returns `self`.

```python

obj.method1().method2().method3()

```

\*\*(\$H What is the purpose of the `\_\_call\_\_` method in Python?\*\*

It lets an object be called like a function. Example:

```python

class A:

    def \_\_call\_\_(self): print("Called!")

a = A(); a()  # prints "Called!"

```

---

Would you like these compiled into a PDF or formatted for flashcards?

”