# OOP Concepts in Python – Interview Guide

## Q1. What is Abstraction?

Abstraction is the concept of exposing only essential features of an object while hiding implementation details. It focuses on what an object does, not how it does it.
**Example:**

```
from abc import ABC, abstractmethod

class Animal(ABC):
    @abstractmethod
    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return "Bark"
```

## Q2. What is Encapsulation?

Encapsulation is the practice of bundling data and methods together and restricting direct access to the data. It is used to protect an object's internal state.
**Example:**

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # private variable

    def get_balance(self):
        return self.__balance
```

## Q3. What is Polymorphism?

Polymorphism allows different objects to respond to the same method name in different ways. It enables flexibility and extensibility in code.
**Example:**

```
class Dog:
    def speak(self):
        return "Bark"

class Cat:
    def speak(self):
        return "Meow"

def make_sound(animal):
    print(animal.speak())
```

## Q4. How are Abstraction and Polymorphism related?

Abstraction defines a common interface or contract, while polymorphism allows multiple implementations of that interface. Together, they enable loosely coupled and scalable designs.

## Q5. How is Encapsulation different from Abstraction?

Abstraction hides implementation details and focuses on design, while encapsulation hides data and focuses on data protection. Encapsulation is implemented using access control, whereas abstraction is implemented using interfaces or abstract classes.

## One-line Interview Summary

Abstraction defines what an object should do, Polymorphism defines how differently it can do it, and Encapsulation protects the internal state while doing it.