**1. What are the main features of Python?**

**Definition:**  
Python is a **high-level, interpreted, object-oriented, dynamically typed programming language** known for its **simplicity, readability, and versatility**.

**Main Features:**

1. **Easy to Learn & Readable** → Syntax is close to natural language.
2. **Interpreted** → No compilation step; code is executed line by line.
3. **Dynamically Typed** → No need to declare variable types explicitly.
4. **Cross-platform** → Runs on different operating systems without changes.
5. **Extensive Standard Library** → Built-in modules for various tasks.
6. **Supports Multiple Paradigms** → Object-oriented, functional, and procedural programming.
7. **Large Community & Ecosystem** → Many frameworks and libraries (Django, Flask, NumPy, etc.).

**2. Why choose Python over other languages?**

**Definition:**  
Python is chosen for its **rapid development speed, ease of use, and strong ecosystem**, especially in AI, data science, automation, and web development.

**Reasons:**

* **Shorter code, faster results** compared to C++ or Java.
* **Beginner-friendly syntax** yet powerful for experts.
* **Strong in modern fields** (AI, ML, IoT, Web Dev).
* **Cross-industry usage** (Google, Netflix, NASA).

**3. OOP Concepts in Python**

**Definition:**  
Object-Oriented Programming (OOP) is a **programming paradigm** where programs are organized into **objects** that contain **data (attributes)** and **functions (methods)**.

**Core OOP Concepts:**

1. **Class** → A blueprint for creating objects.
2. **Object** → An instance of a class.
3. **Encapsulation** → Restricting direct access to internal data.
4. **Inheritance** → Acquiring properties/methods from another class.
5. **Polymorphism** → Same method name behaving differently for different objects.
6. **Abstraction** → Hiding complex implementation details.

**Real-world Analogy (Car Factory):**

* **Class**: Car design blueprint.
* **Object**: The actual car built.
* **Encapsulation**: Driver doesn’t need to know the engine’s inner mechanics.
* **Inheritance**: ElectricCar inherits features from Car but adds battery.
* **Polymorphism**: “Start” method works differently for petrol vs. electric cars.
* **Abstraction**: Using the brakes without knowing hydraulic mechanics.

**4. Method Overloading vs Method Overriding**

**Definition:**

* **Method Overloading** → Same method name but different parameter lists (not directly supported in Python; simulated using default arguments or \*args).
* **Method Overriding** → Subclass provides a new implementation for a method defined in the parent class.

**Example:**

python

CopyEdit

# Overloading simulation

class MathOps:

def add(self, a, b=0, c=0):

return a + b + c

print(MathOps().add(2, 3))

print(MathOps().add(2, 3, 4))

# Overriding

class Parent:

def greet(self):

print("Hello from Parent")

class Child(Parent):

def greet(self):

print("Hello from Child")

Child().greet()

**5. Constructor in Python**

**Definition:**  
A **constructor** is a special method called \_\_init\_\_ that is executed automatically when an object is created, used for **initializing object attributes**.

**Example:**

python

CopyEdit

class Car:

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

self.brand = brand

print(f"Car created: {brand}")

mycar = Car("Tesla")

**6. Abstraction in Python**

**Definition:**  
Abstraction is the **process of hiding implementation details** and showing only necessary functionality, often implemented using **Abstract Base Classes**.

**Example:**

python

CopyEdit

from abc import ABC, abstractmethod

class Vehicle(ABC):

@abstractmethod

def start(self):

pass

class Car(Vehicle):

def start(self):

print("Car engine started")

Car().start()

**Real-world analogy:** You use a TV remote without knowing the internal circuit design.

**7. Array Concept in Python**

**Definition:**  
An **array** is a collection of elements stored in contiguous memory locations, usually of the **same data type**.

**Key Points:**

* In Python, you can use the built-in array module for fixed-type arrays.
* **Lists** can store mixed types but are slower for numerical operations.
* **NumPy arrays** are faster and more efficient for large datasets.

**Example:**

python

CopyEdit

import array

arr = array.array('i', [1, 2, 3])

print(arr[0]) # Output: 1

**8. Types of Looping Statements in Python**

**Definition:**  
Loops are **control structures** that repeat a block of code until a condition is met.

**Types:**

1. **for loop** → Iterates over sequences (lists, tuples, strings).
2. **while loop** → Repeats until a condition becomes false.
3. **Nested loops** → Loop inside another loop.
4. **Loop control statements** →
   * break → Exit loop early.
   * continue → Skip current iteration.
   * pass → Placeholder with no action.

**Example:**

python

CopyEdit

for i in range(5):

if i == 3:

continue

print(i)

**9. Bonus – Common Interview Follow-ups**

* **Is Python pass-by-value or pass-by-reference?**  
  → Neither; it’s **pass-by-object-reference**.
* **Shallow copy vs Deep copy?**  
  → Shallow copies copy references; deep copies create independent copies.
* **How is memory managed in Python?**  
  → Automatic garbage collection + reference counting.