**Python Learning Notes**

**1️ Data Types in Python**

**Python offers built-in data types for handling various kinds of data.**

* **Numeric: int, float, complex**
* **Sequence: list, tuple, range**
* **Text: str**
* **Set: set, frozenset**
* **Mapping: dict**
* **Boolean: bool**
* **Binary: bytes, bytearray, memoryview**

**🔹 Examples:**

**x = 10 # int**

**y = 3.14 # float**

**z = "Hello" # str**

**nums = [1, 2, 3] # list**

**coords = (10, 20) # tuple**

**data = {"name": "Alice", "age": 25} # dict**

**flag = True # bool**

**2️Object-Oriented Programming (OOP)**

**OOP organizes code into classes and objects, promoting modularity and reusability.**

**🔹 Four Pillars of OOP:**

* **Encapsulation**
* **Inheritance**
* **Polymorphism**
* **Abstraction**

**3️ Constructors (\_\_init\_\_)**

**Used to initialize object attributes when an instance is created.**

**class Student:**

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

**self.name = name**

**self.age = age**

**s1 = Student("John", 21)**

**print(s1.name, s1.age)**

**4️ Inheritance**

**Allows a child class to inherit methods and properties from a parent class.**

**class Animal:**

**def speak(self):**

**print("This is an animal")**

**class Dog(Animal):**

**def speak(self):**

**print("Bark")**

**d = Dog()**

**d.speak()**

**5️ Abstraction**

**Hides implementation details and exposes only essential features.**

**from abc import ABC, abstractmethod**

**class Vehicle(ABC):**

**@abstractmethod**

**def start(self):**

**pass**

**class Car(Vehicle):**

**def start(self):**

**print("Car started")**

**c = Car()**

**c.start()**

**6️ Polymorphism**

**Same method behaves differently depending on the object calling it.**

**class Bird:**

**def fly(self):**

**print("Most birds can fly")**

**class Penguin(Bird):**

**def fly(self):**

**print("Penguins cannot fly")**

**b1 = Bird()**

**p1 = Penguin()**

**b1.fly()**

**p1.fly()**

**7️ Importance of OOP in Python**

* **Encapsulation: Protects internal data**
* **Inheritance: Promotes code reuse**
* **Polymorphism: Enhances flexibility**
* **Abstraction: Simplifies complex systems**

**OOP makes Python code more structured, maintainable, and scalable.**

**8️ Practical Applications of Python OOP**

* **Web Development: Django, Flask use OOP for models/views**
* **Game Development: Characters and interactions modeled as objects**
* **Data Science: Classes for datasets and ML models**
* **Automation: Modular scripts using reusable components**