PRESENTATION

By Rishabh Dubey

AGENDA

- Functional Programming
- Object-Oriented Programming
- File Handling

QUICK REVISION QUIZ (MCQS & TRUE/FALSE)

1 List is a mutable data type

True

False

QUICK REVISION QUIZ (MCQS & TRUE/FALSE)

2️⃣ Which of the following is an immutable data type?

a) List
b) Dictionary
c) Tuple
d) Set

QUICK REVISION QUIZ (MCQS & TRUE/FALSE)

3 Tuples allow item reassignment

True

False

Answer: False)

Functional Programming

INTRODUCTION TO FUNCTIONAL PROGRAMMING

✅ Uses pure functions, immutability, and higher-order functions

✅ Helps in cleaner, reusable, and efficient code

```
def add(a, b):
    return a + b
print(add(5, 10))
```

FUNCTIONAL PROGRAMMING VS OBJECT-ORIENTED PROGRAMMING

Feature	Functional Programming (FP)	Object-Oriented Programming (OOP)
Paradigm	Declarative (what to do)	Imperative (how to do)
Structure	Uses functions	Uses classes & objects
State Management	No shared state, avoids side effects	State is stored in objects
Key Focus	Data transformation & pure functions	Data encapsulation & behavior
Example Use Cases	Data processing, mathematical operations	Building software models, GUI applications
Example Code	map(), filter(), lambda	class, objects, inheritance

EXAMPLE COMPARISON:

✅ Functional Approach:

```
def add(a, b): return a + b
print(add(5, 10))
```

✅ OOP Approach

```
class Calculator:
    def add(self, a, b):
        return a + b
calc = Calculator()
print(calc.add(5, 10))  # Output: 15
```

PRACTICE QUESTION

📝 Task:

- 1️⃣ Function to find max of two numbers
- you can only use functional approach

LAMBDA FUNCTIONS

✅ Anonymous functions (no def, one-liner)

- Syntax: **lambda arguments: expression**
- Example:

```
square = lambda x: x ** 2
print(square(4))  # Output: 16
```

A lambda function in Python is an anonymous function (without a name) that can have any number of arguments but only one expression.

LAMBDA FUNCTIONS

💡 Questions (Lambda Functions)

1️⃣ Write a lambda function to check if a number is a multiple of 3.

2️⃣ Write a lambda function to find the square of a number.

MAP, FILTER, REDUCE

- ✅ map() – Applies a function to all elements
- ✅ filter() – Filters elements based on a condition
- ✅ reduce() – Reduces elements to a single value

Object-Oriented Programming

(ENCAPSULATION , INHERITANCE , POLYMORPHISM , ABSTRACTION)

INTRODUCTION TO OOP

✅ OOP – Organizes code into classes & objects

✅ Key Principles:

1. **Encapsulation** – Restrict access to methods & variables
2. **Inheritance** – Child class inherits parent class properties
3. **Polymorphism** – Same method, different behavior
4. **Abstraction** – Hide complex implementation

CLASSES & OBJECTS

📌 What are Classes & Objects?

✅ Class → A blueprint for creating objects.

✅ Object → An instance of a class with unique attributes and behaviors.

📌 Key Concepts in Class & Objects

✅ __init__() Constructor → Initializes attributes when an object is created.

✅ Attributes (Variables) → Store object data (self.brand, self.model).

✅ Methods (Functions inside a Class) → Define object behaviors (display()).

✅ self Keyword → Refers to the current instance of the class.

🔹 Example: Real-world Analogy

Class: Car (Blueprint)

Objects: Tesla Model S, BMW X5 (Specific Cars)

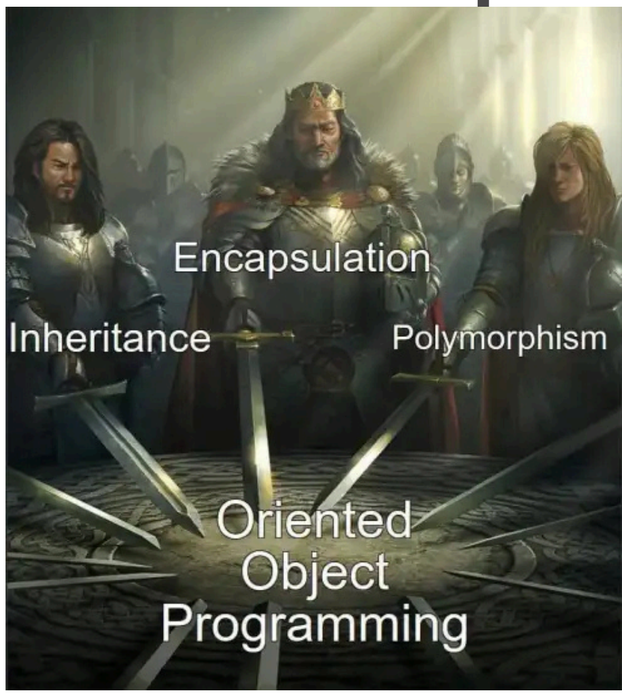
INHERITANCE & POLYMORPHISM

✅ Inheritance: Allows a child class to use methods from the parent class
✅ Polymorphism: Overriding methods for different behavior

```
class Animal:
    def speak(self):
        print("Animal speaks")

class Dog(Animal):
    def speak(self):
        print("Dog barks")

d = Dog()
d.speak()  # Output: Dog barks
```



File Handling

READING & WRITING FILES

INTRODUCTION TO FILE HANDLING

✅ What is File Handling?

- Reading & writing data to external files.
- Supports different file types: Text (.txt), CSV (.csv), JSON (.json), Binary (.bin).
- ✅ Why is it Important?
- Store and retrieve data persistently.
- Process large datasets efficiently.

READING & WRITING TEXT FILES (.TXT)

✅ Opening a File: open(filename, mode)

"r" → Read
"w" → Write (Overwrites file)
"a" → Append
"x" → Create

- Example - Writing to a File:

```
with open("example.txt", "w") as f:
    f.write("Hello, World!")
```

- Example - Reading a File:

```
with open("example.txt", "r") as f:
    print(f.read())  # Output: Hello, World
```

EXCEPTION HANDLING IN FILE OPERATIONS

✅ Preventing Errors Using try-except

```
try:
    with open("missing_file.txt", "r") as f:
        print(f.read())
except FileNotFoundError:
    print("File not found!")
```

✅ Common File Handling Errors:

- FileNotFoundError – Trying to read a non-existent file.
- PermissionError – No access to the file.
- IOError – Issues in reading/writing.

COMMON EXCEPTIONS IN PYTHON

Exception Name	Description
BaseException	The base class for all built-in exceptions.
Exception	The base class for all non-exit exceptions.
ArithmeticError	Base class for all errors related to arithmetic operations.
ZeroDivisionError	Raised when a division or modulo operation is performed with zero as the divisor.
OverflowError	Raised when a numerical operation exceeds the maximum limit of a data type.

THANK YOU