Object-Oriented Programming (OOP) vs Functional-Oriented Programming (FOP)

🔧What is OOP?

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "objects". These objects contain both **data (attributes)** and **functions (methods)**.

- Focuses on: **Objects and Classes**
- Common in: Python, Java, C++, C#

Example in Python:

```
class Car:
    def __init__(self, brand):
        self.brand = brand

    def drive(self):
        print(f"Driving a {self.brand} car")

my_car = Car("Toyota")
my_car.drive()
```

🕖What is FOP?

Functional-Oriented Programming (FOP) is a programming paradigm where the program is built using **functions** and **procedures**. There is **no concept of classes or objects**.

- Focuses on: **Functions and Data Flow**
- Common in: C, Haskell, Lisp, early Python

Example in Python:

```
def drive(brand):
    print(f"Driving a {brand} car")

drive("Toyota")
```

📄Key Differences Between OOP and FOP

Feature	OOP	FOP
Basic Unit	Object	Function
Approach	Bottom-Up (combine objects to build apps)	Top-Down (break down tasks into functions)
Focus	Data and behavior together	Functions and flow of data
Data Handling	Encapsulated within objects	Passed between functions
Reusability	Achieved using Inheritance and Polymorphism	Achieved through function reuse
Ease of Maintenance	Easier (modular and structured)	Can become difficult for large projects
Examples	Python, Java, C++, Ruby	C, Haskell, JavaScript (in functional style)

🕡When to Use What?

Use OOP When:

- You want to model real-world entities.
- Your application is large and needs a clear structure.
- You want to reuse and extend code easily.

Use FOP When:

- The task is small and simple.
- You are doing mathematical or data transformation tasks.
- You want to focus purely on logic and avoid state changes.

👙Summary

- **OOP** helps organize code using real-world concepts like objects and classes.
- **FOP** focuses on creating reusable functions and logical flow.

Both are powerful. Python even allows **mixing both OOP and FOP**, giving you flexibility based on your needs.

Would you like to see a mixed (hybrid) example using both OOP and FOP in the same program?