Comparative Analysis of Programming Paradigms

Objective: To examine the six primary programming paradigms, analyzing their core philosophies, methodologies, and representative languages.

---

1. Imperative Paradigm

- **Philosophy:** Focuses on **how** a task should be accomplished through explicit, step-by-step instructions.
- **Key Concept:** Programs consist of sequences of statements that change the program's state. It is closely tied to the Von Neumann architecture.
- **Common Languages:** Fortran, COBOL, C.

2. Procedural Paradigm

- **Philosophy:** A derivation of imperative programming that organizes code into reusable, modular units called **procedures** or functions.
- **Key Concept:** It emphasizes modularization and code reusability by breaking down complex algorithms into smaller sub-tasks.
- **Common Languages:** Pascal, BASIC, C.

3. Object-Oriented Paradigm (OOP)

- **Philosophy:** Views a program as a collection of interacting **objects** that encapsulate both data (attributes) and behavior (methods).
- **Key Concept:** Relies on four primary pillars: Encapsulation, Abstraction, Inheritance, and Polymorphism.
- **Common Languages:** Java, C++, Python, C#.

4. Functional Paradigm

- **Philosophy:** Treats computation as the evaluation of mathematical functions, avoiding state changes and mutable data.
- **Key Concept:** Emphasizes **Pure Functions** (where the output depends only on the input) and Higher-Order Functions.
- **Common Languages:** Haskell, Lisp, Erlang, F#.

5. Declarative Paradigm

- **Philosophy:** Focuses on **what** the program should achieve rather than the specific control flow of how to do it.
- **Key Concept:** The programmer describes the desired result or logic, and the underlying system handles the implementation details.
- **Common Languages:** SQL, HTML, CSS.

6. Logic Paradigm

- **Philosophy:** Based on formal logic; programs are expressed as sets of **facts and rules**.
- **Key Concept:** The system uses logical inference and backtracking to find solutions to queries based on the provided rules.
- **Common Languages:** Prolog, Mercury.

Summary Comparison Table

Paradigm	Primary Focus	Main Goal	Typical Usage
Imperative	Sequence of commands	Change state	Low-level systems
Procedural	Procedures/Routines	Modularity	General computation
OOP	Objects/Entities	Data encapsulation	Large-scale software
Functional	Math functions	Avoiding side effects	Data processing
Declarative	Logic/Relationships	Outcome-focused	Data querying
Logic	Facts and Rules	Theorem proving	Artificial Intelligence

s