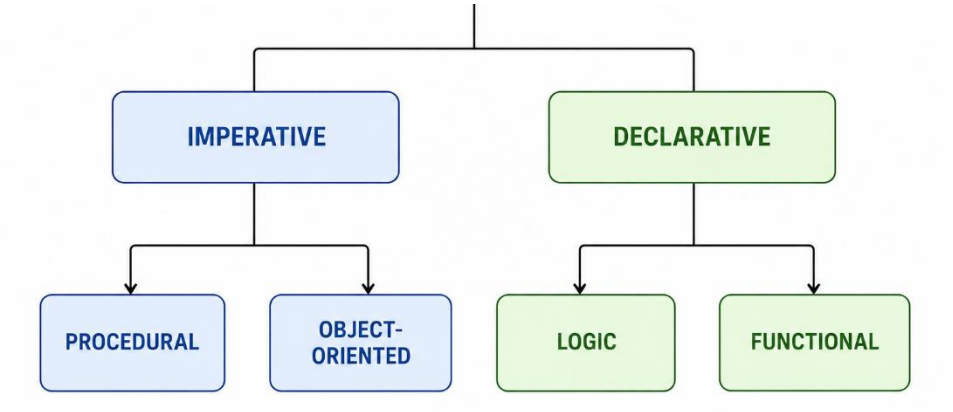
# 1.2.4 Programming Paradigms

## Programming Paradigms

A programming paradigm is a style or approach to programming that defines how programs are structured and written.




High-level programming languages can be split into two paradigms:

- **Imperative languages** – use statements to control how a program executes and changes its state.
- **Declarative languages** – use statements describe the result without describing its control flow.

| Imperative Languages | Declarative Languages |
|---|---|
| Describe how a program should reach an outcome. | Describe what result is wanted rather than how to achieve it. |
| Can change the state of the program. | The programmer does not explicitly describe the control flow used to achieve the result. |
| The order of program statements dictates the outcome. | The order in which statements are written is generally less important than in imperative programming. |
| Variables are mutable | Variables are immutable. |

These paradigms can be further reduced into smaller groups:

- Procedural languages
- Object oriented languages
- Logic languages
- Functional languages

## Procedural Languages

A procedural language is an imperative programming paradigm in which a program is organised into procedures or subroutines.

Procedural languages lay out the code as a series of statements such as:

- Sequence
- Selection
- Iteration

Procedures and functions can be reused, supporting modular program design.

Examples of procedural languages are:

- Fortran
- C
- Pascal

## Object-oriented Languages

An object-oriented language is an imperative programming paradigm that uses classes and objects to group related data and behaviour using attributes and methods.

Object oriented languages make use of techniques such as:

- Encapsulation
- Inheritance
- Polymorphism

A class can be used to create multiple objects (instances) with the same structure and behaviour.

Examples of object oriented languages are:

- C Sharp
- Python
- Java