PARADIGMS OF PROGRAMING

What is a Programming Paradigm?

Programming paradigms are different ways or styles in which a given program or programming language can be organized. Each paradigm consists of certain structures, features, and opinions about how common programming problems should be tackled.

Imperative Programming

Imperative programming consists of sets of detailed instructions that are given to the computer to execute in a given order. It's called "imperative" because as programmers we dictate exactly what the computer has to do, in a very specific way.

Imperative programming focuses on describing how a program operates, step by step.

Procedural Programming

Procedural programming is a derivation of imperative programming, adding to it the feature of functions (also known as "procedures" or "subroutines").

In procedural programming, the user is encouraged to subdivide the program execution into functions, as a way of improving modularity and organization.

Functional Programming

Functional programming takes the concept of functions a little bit further.

In functional programming, functions are treated as **first-class citizens**, meaning that they can be assigned to variables, passed as arguments, and returned from other functions.

Another key concept is the idea of **pure functions**. A **pure** function is one that relies only on its inputs to generate its result. And given the same input, it will always produce the same result. Besides, it produces no side effects (any change outside the function's environment).

With these concepts in mind, functional programming encourages programs written mostly with functions (surprise 😲). It also defends the idea that code modularity and the absence of side effects makes it easier to identify and separate responsibilities within the codebase. This therefore improves the code maintainability

Declarative Programming

Declarative programming is all about hiding away complexity and bringing programming languages closer to human language and thinking. It's the direct opposite of imperative programming in the sense that the programmer doesn't give instructions about how the computer should execute the task, but rather on what result is needed.

Object-Oriented Programming

One of the most popular programming paradigms is object-oriented programming (OOP).

The core concept of OOP is to separate concerns into entities which are coded as objects. Each entity will group a given set of information (properties) and actions (methods) that can be performed by the entity.

OOP makes heavy usage of classes (which are a way of creating new objects starting out from a blueprint or boilerplate that the programmer sets). Objects that are created from a class are called instances.