1) Introduction:-

A programming language is a set of commands, instructions, and other syntax used to create a software program.
Languages that programmers used to write code are called "high-level languages;" which are then compiled into "Low-level language, recognized directly by the computer hardware.

1.1) Definition of a Programming Paradigm:

- A programming paradigm is a fundamental style of computer programming.
- A multi-paradigm programming language is a programming language that supports more than one programming paradigm.

The design goal of such languages is to allow programmers to use the best tool for a job, admitting that no one paradigm solves all problems in the easiest or most efficient way.

- A programming paradigm is a fundamental style of programming regarding how solutions to problems are to be formulated in a programming language.

1.2. Types of Programming Paradigm:-

(I) Imperative Programming.
(II) Declarative Programming.
(III) Functional Programming.
(IV) Logic Programming.
(V) Concurrent Programming.
(VI) Object-oriented programming.

(I) Imperative Programming.

- Action oriented languages, consists of a series of actions which produce a result.
  - C
  - Pascal
  - Basic
  - Fortran
- Imperative programming is a programming paradigm that describes computation in terms of statements that change a program state.
- Imperative programs define sequences of commands for the computer to perform.
- Imperative languages like Fortran, BASIC and C were abstractions of an assembly language.
- Procedural programming is imperative programming in which the program is built from one or more procedures. (also known as subroutines or functions.
- Heavily procedural programming, in which state changes are localized to procedures or restricted to explicit arguments and returns

from procedures, is known as structured programming.

(11) Declarative Programming :- eg: SQL

- It is a non imperative stype of programming in which programs describe the desired results of a program, without explicitly listing comm and or steps that need to be carried out to achieve the results.
- Declarative programming, expresses what needs to be done, without prescribing how to do it in terms of sequences of actions to be taken.
- A program that describes what computation should be performed and not how to compute it
- Declarative programming is an umbrella term that includes a number of better-known programming paradigms.

#. Declarative programming is a programming paradigm that expresses the logic of a computation without describing its control flow

KP:
- Non-imperative nature
- Focus on what, not how
- SQL-Example
- Avoids control flow.
- Expresses logic of computation.
- Its an umbrella term.

LMP: Declarative programming improves readability and maintainability, as the code is often shorter and easier to understand.

Variable ko value dena ya badalna

Assignment allowed nahi hota.
Value ek baar set hoti hai, fir change nahi hoti.

(iii) Functional programming:-

- Functional programming is a subset of declarative programming.
- It treats problems as mathematical functions and avoids changing states or using assignments.
- Pure functional programming does not use assignments, which means no variable reassignment.
- It aims to eliminate side effects and make code more predictable and reliable.
- Functional programs discourage changes in variable values, and prefer recursion over loops.
- In functional programming, functions behave like mathematical functions, producing the same result for the same input.
- Examples of functional programming languages
  - LISP
  - Scheme
  - Standard ML
  - Haskell
- Pure functional languages like Haskell only use functions that transforms inputs and do not modify program state.

Mypoint: Functional programming is a style of programming where programs are built using functions just like in mathematics. It avoids changing values and does not use assignments. Instead, it uses pure functions and recursion.

Logic programming meh hum rules aur facts likhte hain, aur computer logic lagaker answer nikalta hai

(IV) Logic Programming

- Logic programming uses mathematical logic to solve problems.
- Programs are written using facts and rules (called logical statements)
- It is both, declarative and procedural in nature
- Example language: Prolog.
- Instead of telling how to solve a problem, logic programming asks the system to find the solution using reasoning.
- It uses queries and rules, and the system searches for proofs to answer the query.
- The process is based on backward reasoning (goal reduction).
- Logic programs run by applying inference rules to a set of known facts.
- SQL and Prolog can use the logic-based techniques to get answers.

MN. Logic programming is a style of programming where we write rules and facts, and the computer uses logic to find answers.

Concurrent programming matlab ek saath kai kaam chalana, har kaam apne alag Thread mein chalta hain

Thread ek chhoti line of kaam hoti hai jo program ke andar se aur saath-saath chalti hai

(v) Concurrent programming :-

- Concurrent programming allows multiple computations to run at the same time.
- Each computation has its own thread of control.
- It supports multiple threads of execution in a program.
- Example: one thread handles user input, another handles database access.
- Used to make programs faster and more responsive.
- Popular concurrent languages:
  - Ada
  - Java
  - Concurrent Pascal

Concurrent programming means running many tasks at once using different threads.

OOP ek style hai jisme code objects ke through likha jaata hai - jisme data aur function dono hote hain.

(VI) Object-oriented programming (OOP)

- OOP is a programming paradigm that uses "objects" - data structures consisting of data fields and methods.
- OOP features include encapsulation, abstraction, polymorphism, modularity, and inheritance.

Encapsulation:
Wrapping data and methods together in one unit (object)
Data aur function ko ek saath ek object meh bandhna

Abstraction:
Showing only important details, hiding the inner working.
Sirf zaroori chiz dikhana, baaki details chhupana

Polymorphism:
One thing (function or object) can behave differently in different situations.
Ek hi function alag-alag tarike se kaam kare

Modularity:
Breaking a program into small parts (modules) for easy understanding and testing.
Program ko chhote-chhote parts mein todna, jisse samajhna aur test karna easy ho.

Inheritance:
One class can reuse or inherit features (data/functions) from another class.
Ek class dusri class ke features ko use kar sakti hai.

- An object bundles data and functions together, like a real-world entity. (eg: bank account or player).
- Other code accesses the object only through allowed functions (methods), not directly.
- In OOP, a program is a collection of interacting objects.
- Object-oriented languages (like Java, C++, Smalltalk) group data + methods as a single unit called an object.
- Users can access object data only through methods; internal working is hidden (information hiding)
- Polymorphism was developed to solve the problem of needing every object to have all methods.
- OOP is a paradigm, not a language - even assembler languages like HLA can follow OOP.
- Some OOP languages are also imperative. (eg: C++)

OOP is a programming style where code is written using objects that contain both data and functions.

ADT mein data ko direct access nahi hota. Sirf fix kiye gaye functions (interfaces) se hi access hota hai.

1.3 Abstract Data Type (ADT)

- ADT (Abstract Data Type) is a specification of a ~~collection~~ of a data type along with a set of operations that can be performed on the data.
- It defines what operations can be done, not how they are implemented.
- ADT hides the internal details (data structure), and exposes only interface (set of operations)
- The interface is the only way to access and modify the data.
- ADTs are used to define a new type from which multiple instances (objects) can be created
- ADTs are used in object-oriented programming as classes, and their instances are called objects
- Each ADT has two parts:
  - Data: Structure of the data.
  - Operations: Valid functions that define how to interact with the data.
- ADTs can include constructor (to create an instance) and destructor (to delete it).
- ADTs focus on:
  - The involved data.
  - The allowed operations.
  - Axioms (rules)
  - Preconditions and postconditions (required conditions for operations).
- Operations may change the state of the ADT. So, order of execution matters.
- An Abstract data type (ADT) is characterized by the following properties:

(i) It exports a type.
(ii) It exports a set of operations (called interface)

(iii) Only operations of the interface can access or modify the data.
(iv) Axioms and preconditions define how the type behaves in its domain.

Examples of ADT:-
Stack, Queue, List, Tree, Graph.

ADT is a model where data is accessed only through fixed operations, not directly.