Welcome to Python Programming

Unit 1: Problem Solving using Flowchart and Algorithm
Lecture 1: Introduction, Steps for problem-solving

Agenda

Introduction to Problem Solving
The Role of Computers
Systematic Approach
Steps in Problem Solving
Summary

What is Problem Solving?

Problem solving is the process of formulating a problem, finding a solution, and expressing the solution.
In computer science, it means transforming a real-world problem into a sequence of instructions a computer can execute.
It is language-independent at the initial stages.

The Role of Computers

Computers are fast, accurate, but fundamentally dumb.
They require explicit, step-by-step instructions.
The programmer's job is to bridge the gap between human intent and machine execution.
Garbage In, Garbage Out (GIGO) principle.

Why a Systematic Approach?

Reduces complexity of large problems.
Makes debugging easier and more predictable.
Allows for team collaboration and modularity.
Ensures edge cases are not missed.

Step 1: Problem Definition

Clearly understand what the problem is.
Identify the exact requirements.
Determine what the output should be.
Example: 'Find the largest of three numbers' requires clearly defining what 'largest' means (e.g., handling equal numbers).

Step 2: Input/Output Specification

Identify the inputs: What data is available?
Identify the outputs: What is the expected result?
Determine the data types (e.g., integers, strings, floating-point numbers).
Example: Input -¿ 3 integers, Output -¿ 1 integer (the largest).

Step 3: Algorithm Development

Create a logical sequence of steps to solve the problem.
Use natural language, pseudocode, or flowcharts.
Focus on the logic, not syntax.
Iterative process: refine the steps until they are unambiguous.

Example: Algorithm for Largest of Three

1. Read three numbers: A, B, C
2. If A ¿= B and A ¿= C, then Largest is A
3. Else if B ¿= A and B ¿= C, then Largest is B
4. Else Largest is C
5. Print Largest

Step 4: Implementation (Coding)

Translate the algorithm into a specific programming language (e.g., Python).
Follow the syntax rules of the language.
Write readable code (use comments, meaningful variable names).
This step is mechanical if the algorithm is solid.

Step 5: Testing and Debugging

Run the program with various test cases.
Include normal cases, edge cases (e.g., negative numbers), and invalid inputs.
Debugging: finding and fixing logical or syntax errors.
Iterative refinement.

Summary

Problem solving is a structured process.
Key steps: Define, Specify I/O, Develop Algorithm, Code, Test.
Planning (Steps 1-3) is more critical than Coding (Step 4).