What is HTML?

HTML (HyperText Markup Language) is the standard language used to create and design web pages.
It tells the web browser **how to display text, images, links, and other elements** on a webpage.

- **HyperText** means text that links to other documents.
- **Markup Language** means a set of tags used to “mark up” text to define its structure and presentation.

📄 Uses of HTML

1. To **create and structure web pages**.
2. To **insert text, images, videos, and links**.
3. To **create forms** for user input.
4. To **format text** (bold, italic, underline, headings).
5. To **define page layout** (paragraphs, tables, divisions).
6. To **link multiple web pages** together using hyperlinks.

🏷 HTML TAGS

HTML uses **tags** enclosed in angle brackets < > to define elements.

Example:

```
<p>This is a paragraph.</p>
```

Here:

- `<p>` → Opening tag
- `</p>` → Closing tag
- Everything between them → Element content

🌐 Basic Structure of an HTML Document

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Webpage</title>
</head>
<body>
    <h1>Welcome to HTML</h1>
    <p>This is my first paragraph.</p>
</body>
</html>
```

◆ Important HTML Tags

1. `<head>` Tag

- Contains information **about** the web page (metadata).
- Not displayed on the page.
- Includes title, CSS links, and scripts.

```
<head>
    <title>My Page</title>
</head>
```

---

2. `<title>` Tag

- Sets the **title of the webpage** (appears on the browser tab).

```
<title>Home Page</title>
```

---

3. `<body>` Tag

- Contains all **visible content** (text, images, links, etc.)

```
<body>
    <p>Hello, this is my webpage.</p>
</body>
```

---

4. Empty Tags

- Tags that **don't need a closing tag**.
  Examples:

```
<br>   <!-- Inserts a line break -->
<hr>   <!-- Inserts a horizontal line -->
```

---

5. Container Tags

- Tags that **have both opening and closing parts**.
  Examples:

```
<p>Paragraph text</p>
<b>Bold text</b>
<i>Italic text</i>
```

---

6. Heading Tags (`<h1>` to `<h6>`)

- Used for headings and subheadings.
- `<h1>` is the largest; `<h6>` is the smallest.

```
<h1>Main Heading</h1>
<h2>Subheading</h2>
<h3>Smaller heading</h3>
```

---

7. `<br>` Tag

- Adds a **line break** (moves text to the next line).

```
<p>This is line one.<br>This is line two.</p>
```

---

8. `<hr>` Tag

- Inserts a **horizontal line** to separate sections.

```
<hr>
```

---

9. HTML Comments

- Used to write notes in the code; not shown on the webpage.

```
<!-- This is a comment -->
```

🎨 Formatting and Style Tags

🖌 Background Colour

Use the `bgcolor` attribute in the `<body>` tag (HTML4) or use **CSS (recommended)**.

Example (HTML4 style):

```
<body bgcolor="lightblue">
```

Example (CSS style):

```
<body style="background-color: lightblue;">
```

✍ Font, Bold, and Italic

Bold Text

```
<b>This is bold text</b>
```

Italic Text

```
<i>This is italic text</i>
```

Font (Old method using `<font>`, though CSS is preferred)

```
<font face="Arial" size="4" color="blue">This is styled text</font>
```

📏 Text Alignment and Paragraph

Paragraph Tag

```
<p>This is a paragraph.</p>
```

Text Alignment

Use the `align` attribute inside `<p>` or `<div>`:

```
<p align="left">Left aligned text</p>
<p align="center">Centered text</p>
<p align="right">Right aligned text</p>
```

□ Example: Complete HTML Page

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Example</title>
</head>
<body style="background-color: #f2f2f2;">

    <h1 align="center">Welcome to HTML</h1>
    <hr>

    <p align="justify">
        <b>HTML</b> stands for <i>HyperText Markup Language</i>.
        It is used to design web pages and display content on the
Internet.<br>
        This is an example of using different HTML tags.
    </p>

    <!-- This is a comment and won't appear in the browser -->

</body>
</html>
```

🌐 HTML LISTS, IMAGES, AND LINKS

▯ LISTS IN HTML

HTML provides three main types of lists to organize content:

1 Ordered List (`<ol>`)

- Displays items in a **numbered (ordered)** format.
- Each item is written inside the `<li>` (list item) tag.
- Numbers appear automatically (1, 2, 3...).

Example:

```
<h3>Steps to Start a Computer</h3>
<ol>
    <li>Press the Power button</li>
    <li>Wait for the system to load</li>
    <li>Login with your password</li>
    <li>Start working</li>
</ol>
```

Output:

1. Press the Power button
2. Wait for the system to load
3. Login with your password
4. Start working

Attributes:
You can change numbering style using the `type` attribute:

```
<ol type="A"> → A, B, C
<ol type="a"> → a, b, c
<ol type="I"> → I, II, III
<ol type="i"> → i, ii, iii
```

Example:

```
<ol type="A">
    <li>India</li>
    <li>USA</li>
    <li>UK</li>
</ol>
```

---

2 Unordered List (`<ul>`)

- Displays items in **bulleted (unordered)** format.

- Used when the order of items doesn't matter.

Example:

```
<h3>Fruits</h3>
<ul>
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
</ul>
```

Output:
• Apple
• Mango
• Banana

Attributes:
You can change bullet type using the `type` attribute:

```
<ul type="disc">    → ●
<ul type="circle">  → ○
<ul type="square">  → ■
```

---

3 Definition List (`<dl>`)

- Used to display terms and their definitions (like a dictionary).

Tags Used:

- `<dl>` → definition list
- `<dt>` → definition term
- `<dd>` → definition description

Example:

```
<h3>Computer Terms</h3>
<dl>
    <dt>HTML</dt>
    <dd>HyperText Markup Language – used to create web pages.</dd>

    <dt>CSS</dt>
    <dd>Cascading Style Sheets – used for webpage styling.</dd>
</dl>
```

Output:
HTML
   HyperText Markup Language – used to create web pages.
CSS
   Cascading Style Sheets – used for webpage styling.

🖼 BACKGROUND IMAGE

You can set a **background image** for a web page using the `background` attribute (old method) or CSS (modern method).

◆ HTML4 (older) Method:

```
<body background="img/background.jpg">
```

◆ CSS (recommended) Method:

```
<body style="background-image: url('img/background.jpg');
             background-size: cover;
             background-repeat: no-repeat;">
```

- `background-size: cover;` → image fills the screen
- `background-repeat: no-repeat;` → image does not repeat

🏞 INSERTING AN IMAGE (`<img>` Tag)

The `<img>` tag is used to display images on a webpage.
It is an **empty tag** (no closing tag).

Syntax:

```
<img src="path" alt="text" width="..." height="...">
```

Attributes:

- `src` → Source (path of the image file)
- `alt` → Alternate text (shown if image not found)
- `width` / `height` → Set image size

Example:

```
<img src="img/work01.png" alt="Sample image" width="300" height="200">
```

Output:
🖼 Displays the image named *work01.png* at 300×200 pixels.

🔗 ANCHOR TAG (`<a>`) – HYPERLINK

The `<a>` tag is used to **create links** between web pages, files, or websites.
It can link to:

- Another webpage
- A section of the same page
- An email address

Syntax:

```
<a href="URL">Link Text</a>
```

Example:

```
<a href="https://www.google.com">Visit Google</a>
```

Open link in a new tab

Use the `target="_blank"` attribute:

```
<a href="https://www.google.com" target="_blank">Open Google in new tab</a>
```

Link to a local file or another page

```
<a href="about.html">Go to About Page</a>
```

Email Link

```
<a href="mailto:info@example.com">Send Email</a>
```

💡 Complete Example

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Lists and Links Example</title>
</head>
<body style="background-image: url('img/slide01.png'); background-size:
cover;">

    <h1 align="center">HTML Lists, Images, and Links</h1>
    <hr>

    <h2>1. Ordered List</h2>
    <ol type="1">
        <li>Wake up</li>
        <li>Brush your teeth</li>
        <li>Have breakfast</li>
    </ol>
    <h2>2. Unordered List</h2>
    <ul type="square">
        <li>Pen</li>
        <li>Pencil</li>
        <li>Eraser</li>
    </ul>

    <h2>3. Definition List</h2>
    <dl>
        <dt>CPU</dt>
        <dd>Central Processing Unit - brain of the computer.</dd>
        <dt>RAM</dt>
        <dd>Random Access Memory - temporary memory.</dd>
    </dl>

    <h2>4. Image Example</h2>
    <img src="img/work02.png" alt="Sample Image" width="250" height="150">

    <h2>5. Hyperlink Example</h2>
    <a href="https://www.wikipedia.org" target="_blank">Visit Wikipedia</a>

</body>
</html>
```

1. Introduction to Problem Solving & Programming

Problem solving in programming means **finding a logical and systematic way** to solve a given problem using a computer.

- A **problem**: Any task or situation that needs a solution.
- **Programming**: Writing instructions (code) that a computer can understand and execute to solve a problem.

💡 **Goal**: Convert a real-life problem into a step-by-step logical solution.

---

2. Steps for Problem Solving

These steps help in developing correct and efficient programs:

1. **Problem Definition**
   Understand what is given and what is required.
2. **Analysis**
   Identify inputs, outputs, and processing steps.
3. **Algorithm Design**
   Prepare step-by-step instructions to solve the problem.
4. **Flowchart / Pseudocode**
   Represent the logic visually or in structured text.
5. **Coding**
   Write the program using a programming language.
6. **Testing & Debugging**
   Find and remove errors.
7. **Documentation & Maintenance**
   Explain the program and update when needed.

---

3. Introduction of Algorithm

An **Algorithm** is a **finite set of well-defined steps** to solve a problem.

Characteristics of a Good Algorithm:

- **Input**: Takes zero or more inputs
- **Output**: Produces at least one output
- **Definiteness**: Steps are clear and unambiguous
- **Finiteness**: Ends after a limited number of steps
- **Effectiveness**: Steps are simple and executable

Example:

Algorithm to add two numbers:

1. Start
2. Read A, B
3. Sum = A + B
4. Print Sum
5. Stop

---

4. Understanding and Preparing Flowcharts

A **Flowchart** is a **graphical representation** of an algorithm using symbols.

Common Flowchart Symbols:

Symbol	Meaning
Oval	Start / Stop
Parallelogram	Input / Output
Rectangle	Process
Diamond	Decision
Arrow	Flow of control

Advantages:

- Easy to understand
- Helps in debugging
- Visual representation of logic

---

5. Evolution of Programming Languages

Programming languages evolved to make programming easier and more user-friendly.

Generations of Programming Languages:

1. **1GL (Machine Language)**
   Binary code (0s and 1s)
2. **2GL (Assembly Language)**
   Uses mnemonics (ADD, SUB)
3. **3GL (High-Level Languages)**
   C, C++, Java, Python
4. **4GL (Very High-Level Languages)**
   SQL, MATLAB
5. **5GL (AI-Based Languages)**
   Prolog, AI-based languages

---

6. Types of Programming Languages

Based on usage and level:

a) Low-Level Languages

- Machine language
- Assembly language

b) High-Level Languages

- C, C++, Java, Python
- Easy to learn and use

c) Object-Oriented Languages

- Java, C++, Python
- Uses objects and classes

d) Scripting Languages

- Python, JavaScript, PHP

---

7. Pseudocoding

Pseudocode is a **simple, English-like description** of program logic.

Features:

- Not language-specific
- Easy to understand
- Helps in converting logic into code

Example:

```
START
READ A, B
IF A > B
   PRINT A
ELSE
   PRINT B
END
STOP
```

---

8. Program, Compiler, and Interpreter

Program

A **program** is a set of instructions written to perform a specific task.

Compiler

A **compiler** translates the **entire program at once** into machine language.

Examples: C, C++

Advantages:

- Faster execution
- Errors shown after compilation

Interpreter

An **interpreter** translates and executes the program **line by line**.

Examples: Python, JavaScript

Advantages:

- Easy debugging
- No separate compilation step

Difference Between Compiler and Interpreter:

Compiler	Interpreter
Translates whole program	Translates line by line
Faster execution	Slower execution
Errors shown after compilation	Errors shown line by line
Example: C, C++	Example: Python

Introduction to Problem Solving & Programming

Problem solving in programming means **finding a logical and systematic way** to solve a given problem using a computer.

- A **problem**: Any task or situation that needs a solution.
- **Programming**: Writing instructions (code) that a computer can understand and execute to solve a problem.

Steps for Problem Solving

These steps help in developing correct and efficient programs:

1. **Problem Definition**
   Understand what is given and what is required.
2. **Analysis**
   Identify inputs, outputs, and processing steps.
3. **Algorithm Design**
   Prepare step-by-step instructions to solve the problem.
4. **Flowchart / Pseudocode**
   Represent the logic visually or in structured text.
5. **Coding**
   Write the program using a programming language.
6. **Testing & Debugging**
   Find and remove errors.
7. **Documentation & Maintenance**
   Explain the program and update when needed.

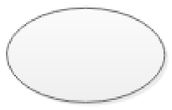
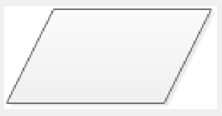
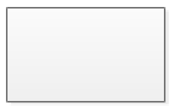
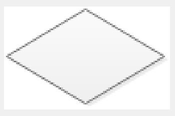
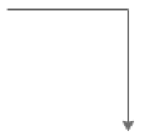
Definition of Algorithm

Writing a logical step-by-step method to solve the problem is called the algorithm. In other words, an algorithm is a procedure for solving problems. In order to solve a mathematical or computer problem, this is the first step in the process.

An algorithm includes calculations, reasoning, and data processing. Algorithms can be presented by natural languages, pseudocode, and flowcharts, etc.

Definition of Flowchart

A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes, and arrows to demonstrate a process or a program. With algorithms, we can easily understand a program. The main purpose of using a flowchart is to analyse different methods. Several standard symbols are applied in a flowchart:

Terminal Box - Start / End	
Input / Output	
Process / Instruction	
Decision	
Connector / Arrow	

The symbols above represent different parts of a flowchart. The process in a flowchart can be expressed through boxes and arrows with different sizes and colours. In a flowchart, we can easily highlight certain elements and the relationships between each part.

Difference between Algorithm and Flowchart

If you compare a flowchart to a movie, then an algorithm is the story of that movie. In other words, **an algorithm is the core of a flowchart**. Actually, in the field of computer programming, there are many differences between algorithm and flowchart regarding various aspects, such as the accuracy, the way they display, and the way people feel about them. Below is a table illustrating the differences between them in detail.

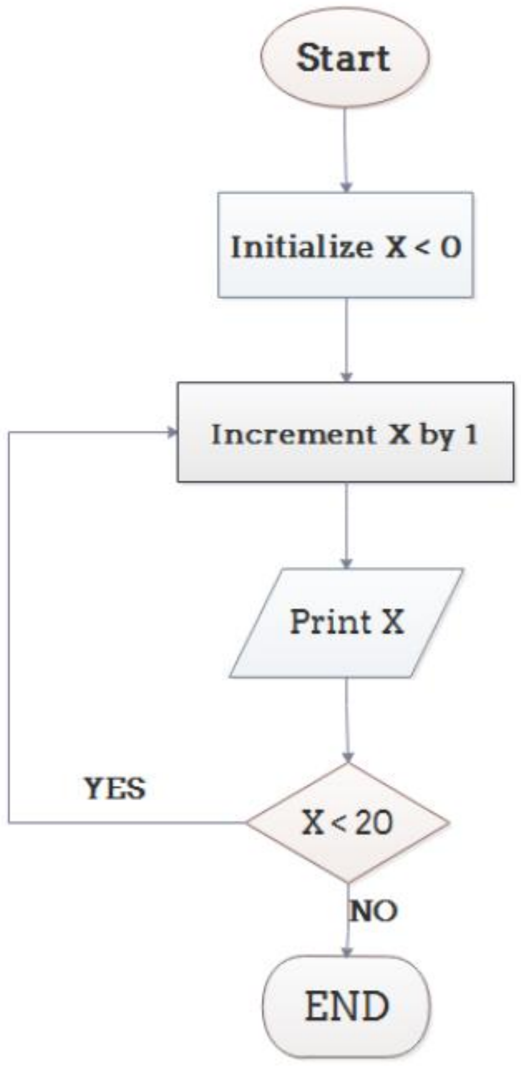
Algorithm	Flowchart
It is a procedure for solving problems.	It is a graphic representation of a process.
The process is shown in step-by-step instruction.	The process is shown in block-by-block information diagram.
It is complex and difficult to understand.	It is intuitive and easy to understand.
It is convenient to debug errors.	It is hard to debug errors.
The solution is showcased in natural language.	The solution is showcased in pictorial format.
It is somewhat easier to solve complex problem.	It is hard to solve complex problem.
It costs more time to create an algorithm.	It costs less time to create a flowchart.

Example 1: Print 1 to 20:

Algorithm:

- Step 1: Initialize X as 0,
- Step 2: Increment X by 1,
- Step 3: Print X,
- Step 4: If X is less than 20 then go back to step 2.

Flowchart:


Start
Initialize X < 0
Increment X by 1
Print X
X < 20
YES
NO
END