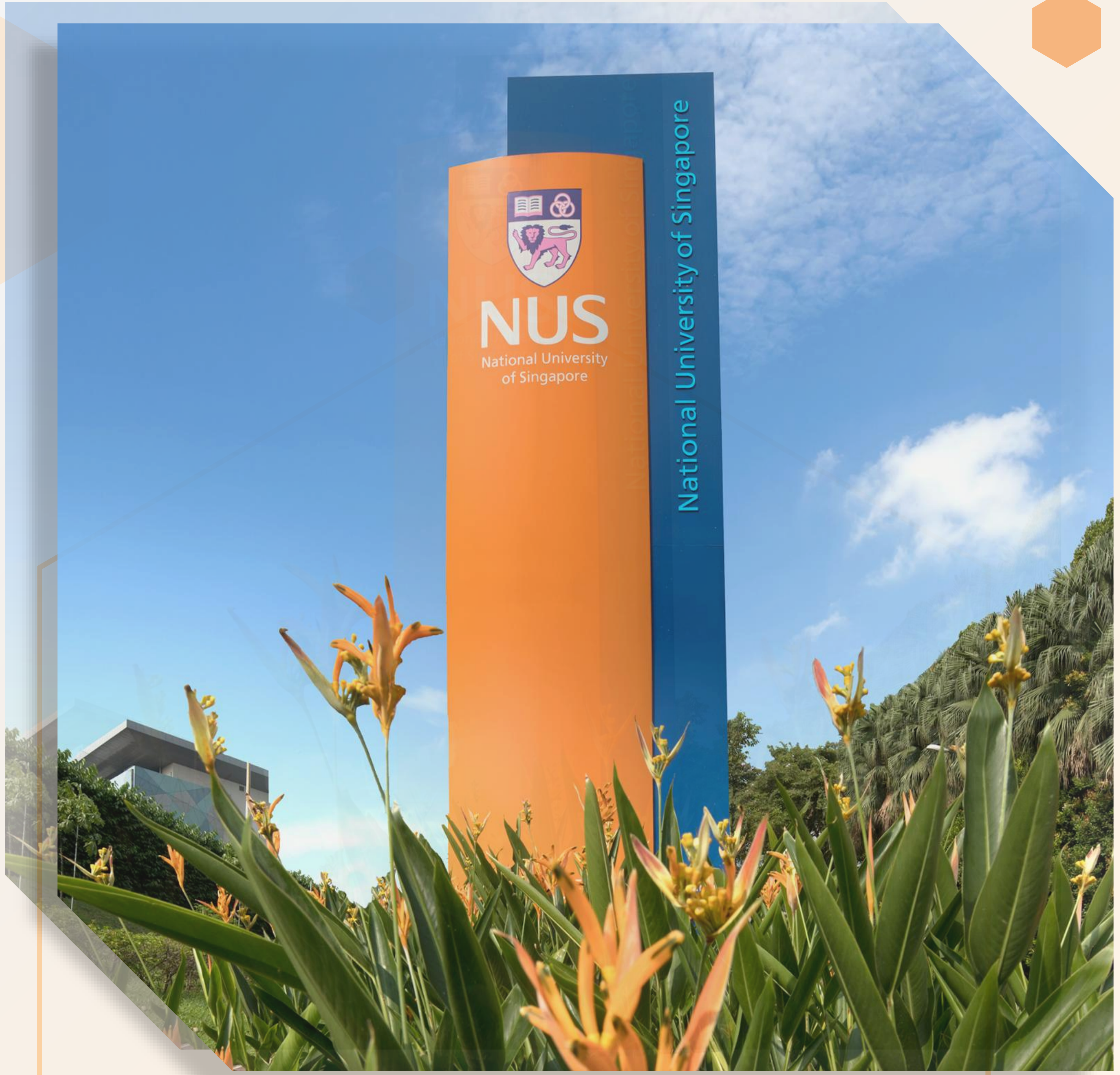
Full Stack Development with AI Programme

Session-1

Manali Dubal

Friday, 3 Jan 2025


NUS
National University of Singapore
National University of Singapore

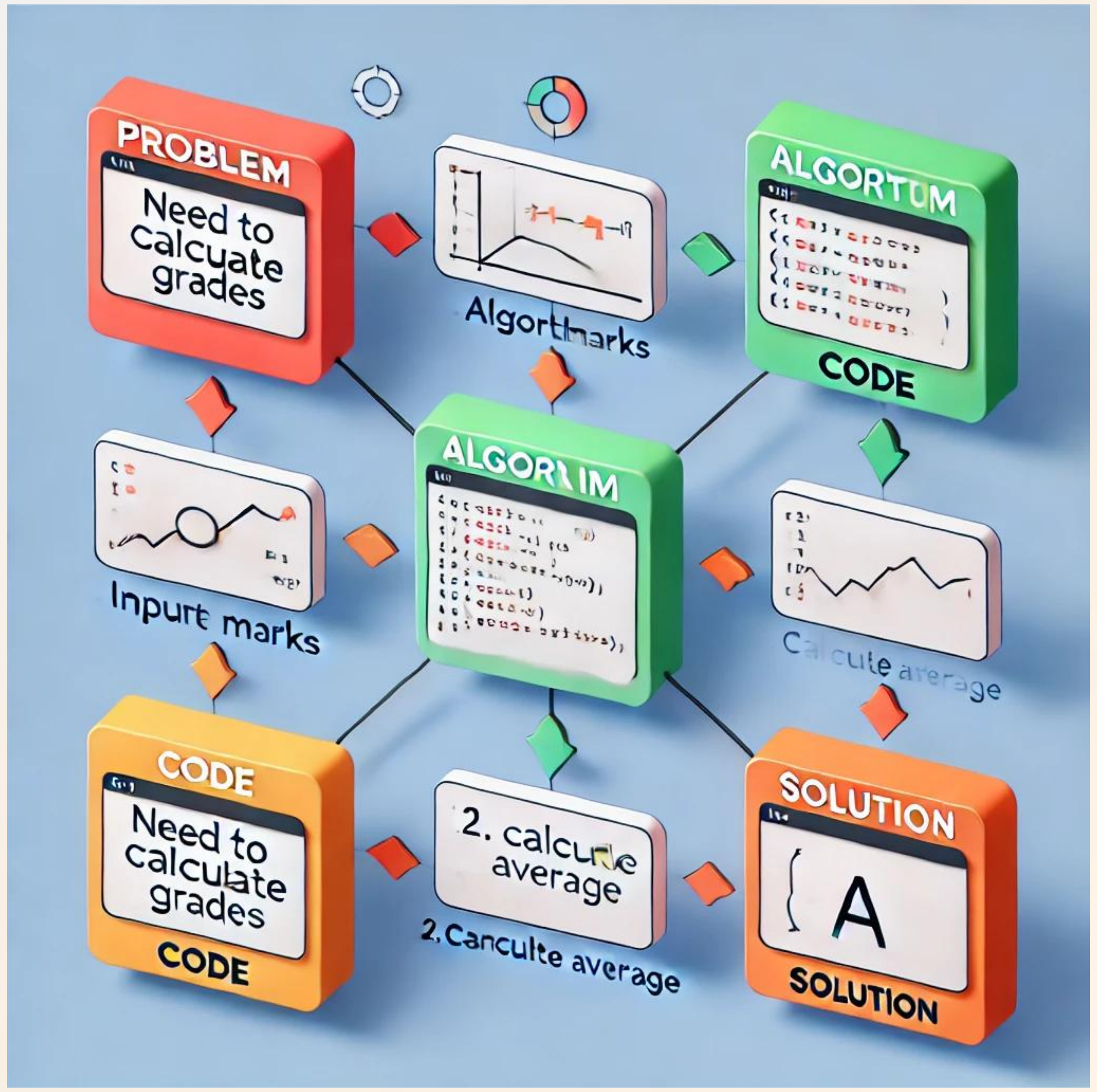
Advanced Computing for Executives
School of Computing

Agenda

- What is Programming
- High-level and Low-level Languages
- Algorithmic thinking
- Variables
- Operators
- Control Flow
- Conditional statements
- Loops
- Functions and its types
- Overview of web development technologies
- Static and Dynamic web pages

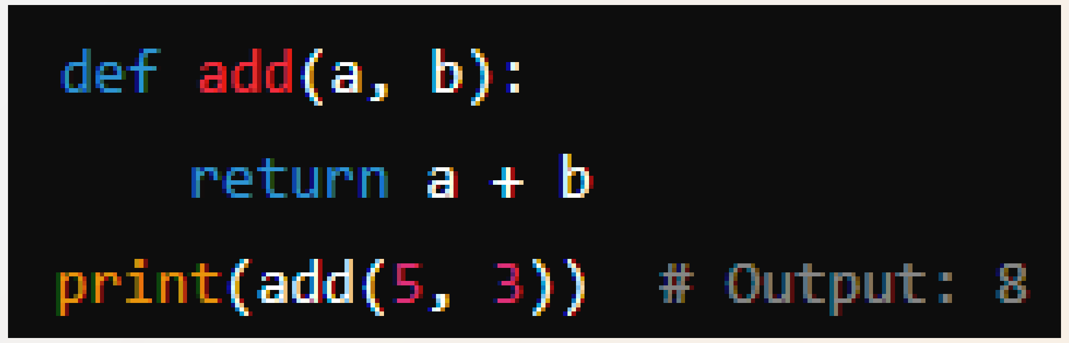
What is Programming?

- Definition: Programming is the process of creating instructions for a computer to perform tasks
- Purpose: Automate tasks, solve problems, and build solutions like apps, websites, or games
- Real-life examples:
  - Automating a to-do list application
  - Developing a weather forecast website
  - Building a chatbot for customer support

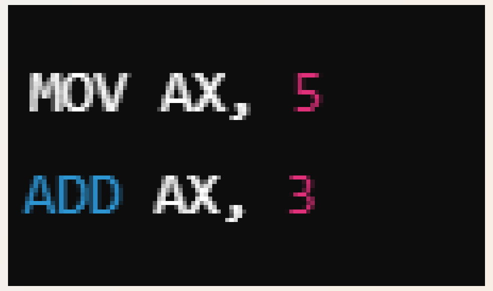


High-Level vs Low-Level Languages

- High-Level Languages:
  - Close to human language; easier to write and understand.
  - Examples: Python, Java, JavaScript.
  - Use Case: Writing a calculator program in Python.
- Low-Level Languages:
  - Close to machine code; faster and more efficient.
  - Examples: Assembly Language, Machine Code.
  - Use Case: Simple addition in Assembly
- Key Differences:
  - High-level: Developer-friendly, slower execution.
  - Low-level: Machine-friendly, faster execution.



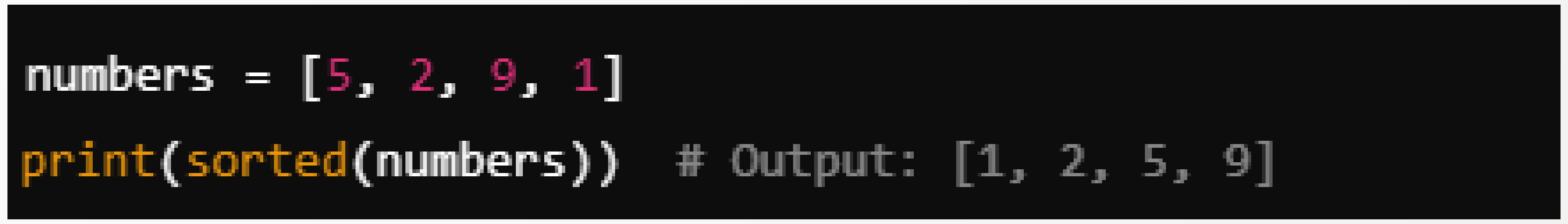
```
def add(a, b):
    return a + b
print(add(5, 3))  # Output: 8
```



```
MOV AX, 5
ADD AX, 3
```

Algorithmic Thinking & Development Tools

- Algorithmic Thinking:
  - Breaking down problems into clear, step-by-step solutions.



```python
numbers = [5, 2, 9, 1]
print(sorted(numbers))  # Output: [1, 2, 5, 9]
```

- Development Tools:
  - Text Editors: VS Code, Sublime Text.
  - IDEs: PyCharm, Eclipse (Example: Debugging Python code).
  - Version Control: Git/GitHub for managing code versions.
  - Debugging Tools: Breakpoints, Loggers.

Introduction to Variables

- What are Variables?
  - Containers for storing data values.
  - Allow dynamic data manipulation.
  - Declaring variables: Use let, const, or var. // avoid using var, deprecated.

Operators Overview

- Types of Operators:
  - Arithmetic Operators: Perform math
    - +, -, *, /, %
  - Comparison Operators: Compare values
    - ==, ===, !=, !==, <, >
  - Logical Operators: Combine conditions
    - && (AND), || (OR), ! (NOT)

Variables and Operators in Action

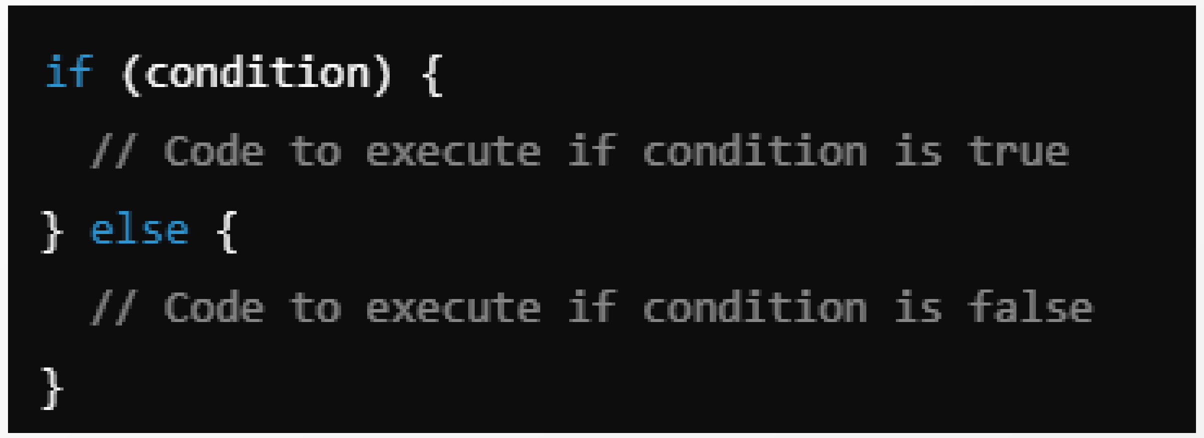
- Example 1: Simple Calculation
- Example 2: Conditional Logic with Operators
- Example 3: Combining Strings (Concatenation)

Introduction to Control Flows

- Control flows determine the order in which instructions are executed in a program.
- Why Use Control Flows?
  - Enable decision-making.
  - Repeat tasks with loops.
  - Create dynamic and responsive programs.
- Types of Control Flows:
  - Conditional Statements (e.g., if, else, switch)
  - Loops (e.g., for, while, do...while)

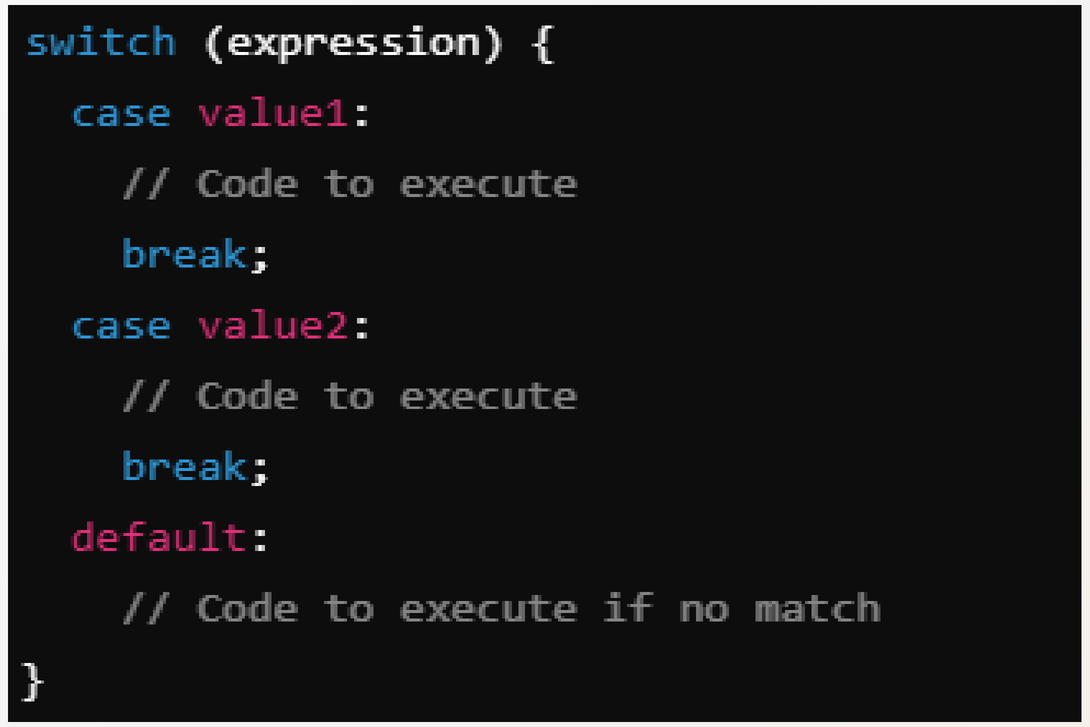
Conditional Statements

- if...else Statement



```
if (condition) {
  // Code to execute if condition is true
} else {
  // Code to execute if condition is false
}
```

- switch Statement:



```
switch (expression) {
  case value1:
    // Code to execute
    break;
  case value2:
    // Code to execute
    break;
  default:
    // Code to execute if no match
}
```

Loops in JavaScript

- Types of Loops:
  - for Loop

```
for (initialization; condition; increment/decrement) {
  // Code to execute repeatedly
}
```

  - while Loop

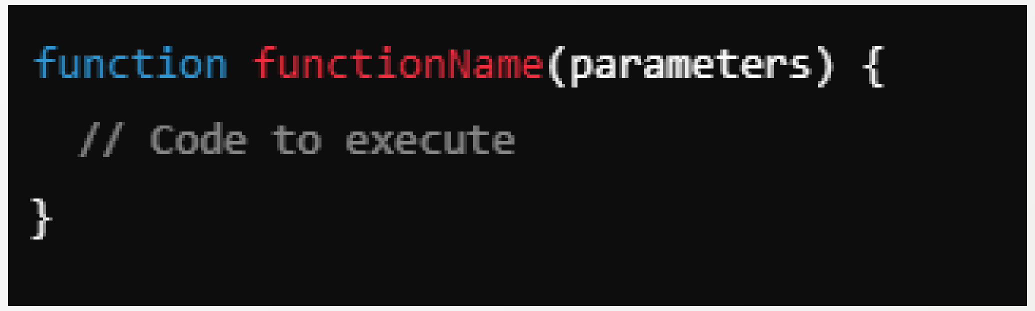
```
while (condition) {
  // Code to execute while the condition is true
}
```

  - do...while Loop

```
do {
  // Code to execute at least once
} while (condition);
```

Introduction to Functions

- A function is a block of reusable code designed to perform a specific task.
- Why use functions?
  - Reusability: Write once, use multiple times.
  - Modularity: Break complex tasks into manageable parts.
  - Readability: Make code cleaner and easier to understand.
  - Maintainability: Centralize code changes.
- Syntax:



```
function functionName(parameters) {
  // Code to execute
}
```

Types of Functions

- Function Declaration:

```
function add(a, b) {
  return a + b;
}
console.log(add(3, 5)); // Output: 8
```

- Function Expression:

```
const subtract = function(a, b) {
  return a - b;
};
console.log(subtract(10, 4)); // Output: 6
```

Functions with Parameters and Return Values

- Parameters: Allow functions to accept inputs.
- Return Values: Provide output from functions.

Overview of Web Development Technologies

- Web development involves creating, building, and maintaining websites.
- Key Components:
    - Frontend Development: User-facing interface.
    - Backend Development: Server-side logic and databases.
    - Full-Stack Development: Combination of both frontend and backend.

Core Technologies for Web Development

- The Building Blocks of the Web:
  - HTML (HyperText Markup Language): Defines the structure of web pages.
  - CSS (Cascading Style Sheets): Adds style and layout to web pages.
  - JavaScript: Adds interactivity and dynamic behavior.

Static vs. Dynamic Web Pages

- Static Web Pages:
  - Content is fixed and served as-is.
  - Built using only HTML and CSS.
  - Example Use Case: Portfolio websites.
- Dynamic Web Pages:
  - Content changes based on user interaction or server response.
  - Requires server-side logic (e.g., PHP, Node.js) or client-side scripting (e.g., JavaScript).
  - Example Use Case: E-commerce platforms.

NUS National University of Singapore | Advanced Computing for Executives School of Computing

LIVE SESSION EXPERIENCE SURVEY

Before we proceed to Q&A
Take 2 minutes to share your feedback with Us!

NUS National University of Singapore | Advanced Computing for Executives
School of Computing

THANK YOU