LESSON PLAN MODULE 1

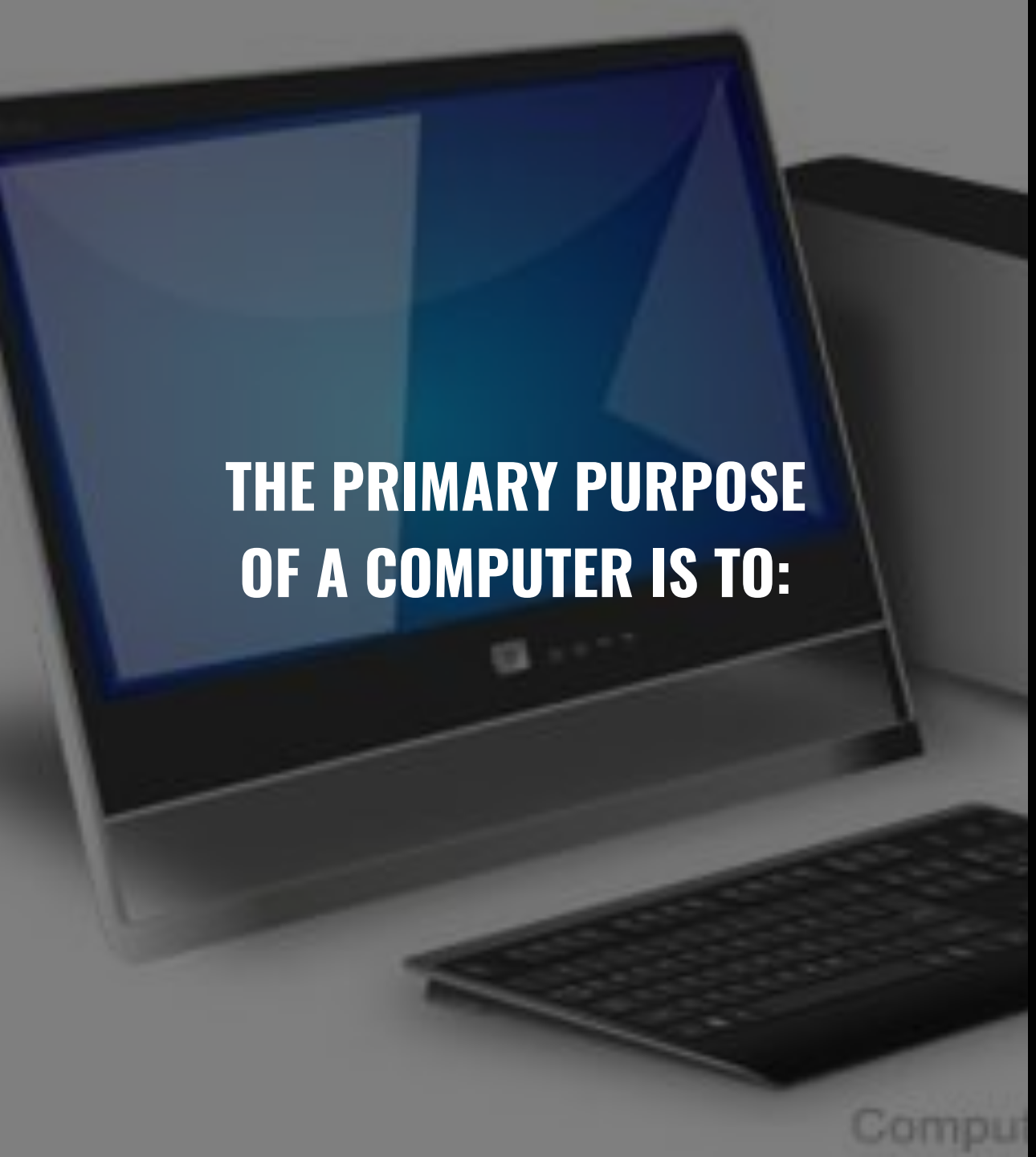
- 1.Lessons 1-4: Introduction to Computers and Programming
    1. Lesson 1: How Computers Work
    2. Lesson 2: Binary and Hexadecimal Systems
    3. Lesson 3: What is Programming?

1. Lessons 4-6: VSCode, Git, and Markdown
    2. Lesson 4: Installing and Using Visual Studio Code (VSCode)
    3. Lesson 5: Introduction to Markdown
    4. Lesson 6: Introduction to Git and GitHub

LESSON 1: HOW COMPUTERS WORK

WHAT IS A COMPUTER?

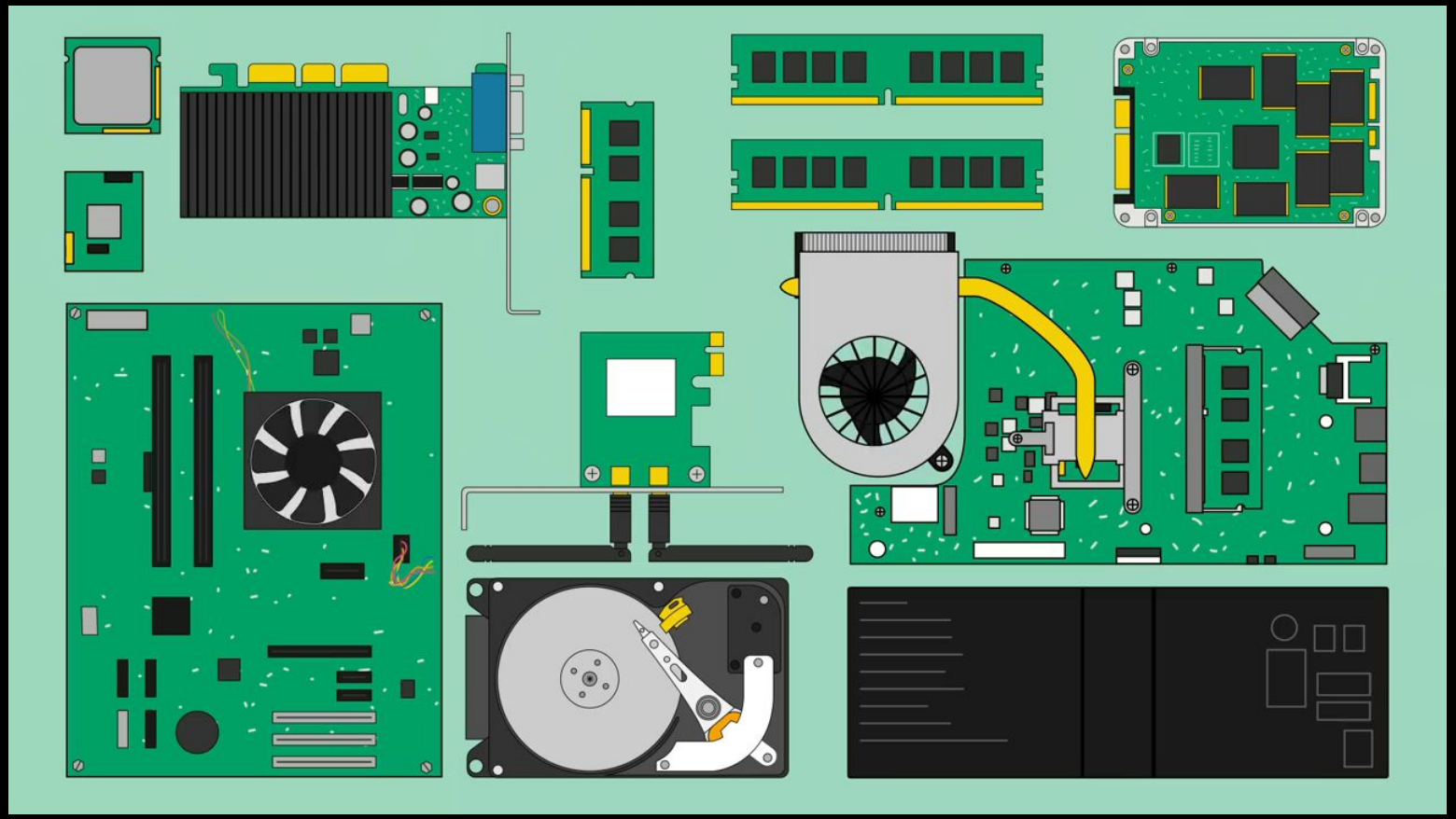
A **computer** is an electronic device that manipulates information or data. It has the ability to store, retrieve, and process data. Computers can perform a wide variety of tasks, from simple calculations to complex operations. They are designed to follow a set of instructions called a program, which enables them to carry out a wide range of applications.

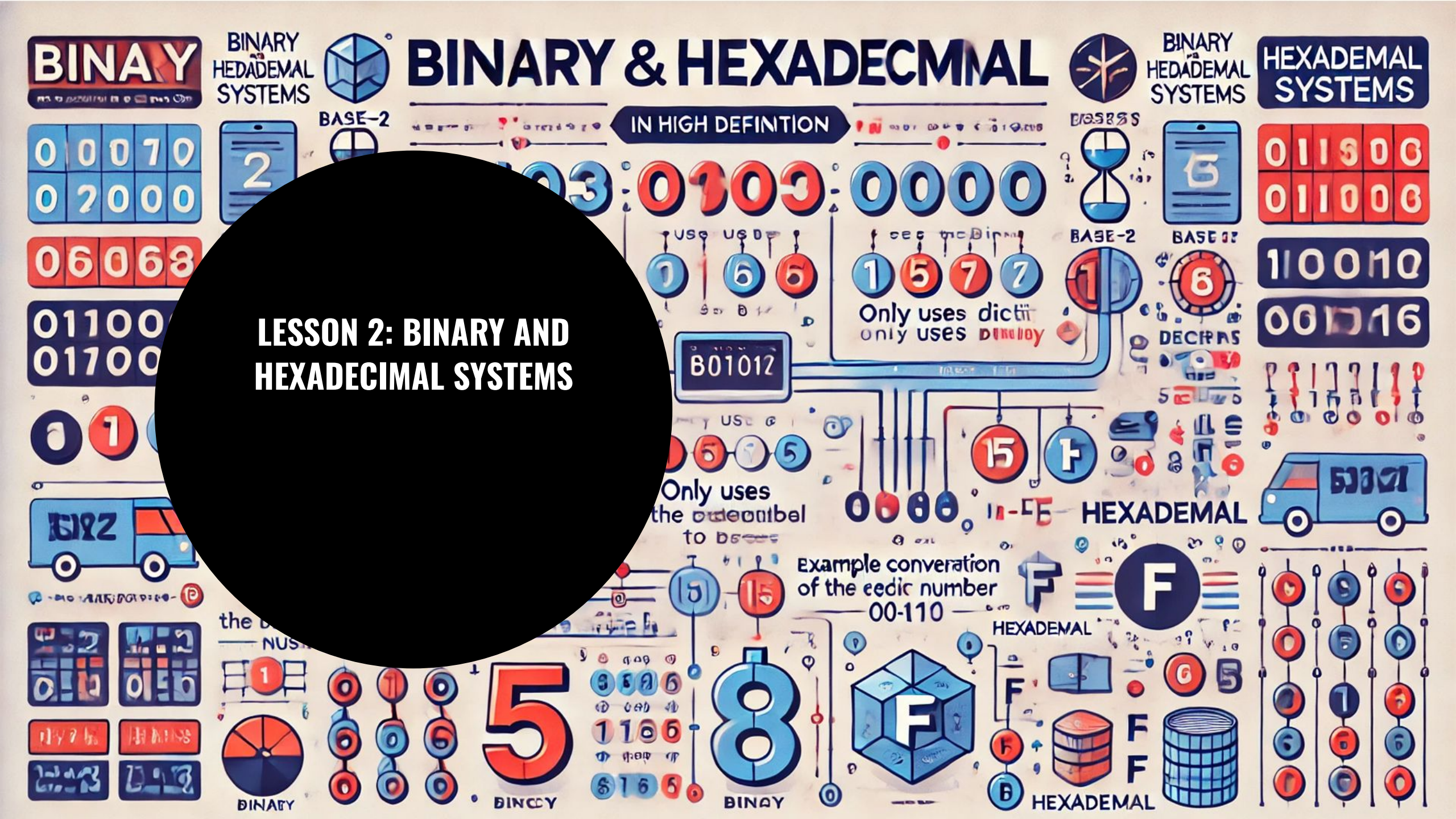


THE PRIMARY PURPOSE OF A COMPUTER IS TO:

- The primary purpose of a computer is to:
- **Process Data**: Computers take raw data and process it into useful information.
- **Store Information**: Computers can store vast amounts of data and retrieve it quickly.
- **Perform Calculations**: Computers can perform complex mathematical operations at high speeds.
- **Automate Tasks**: Computers can automate repetitive tasks, increasing efficiency and productivity.

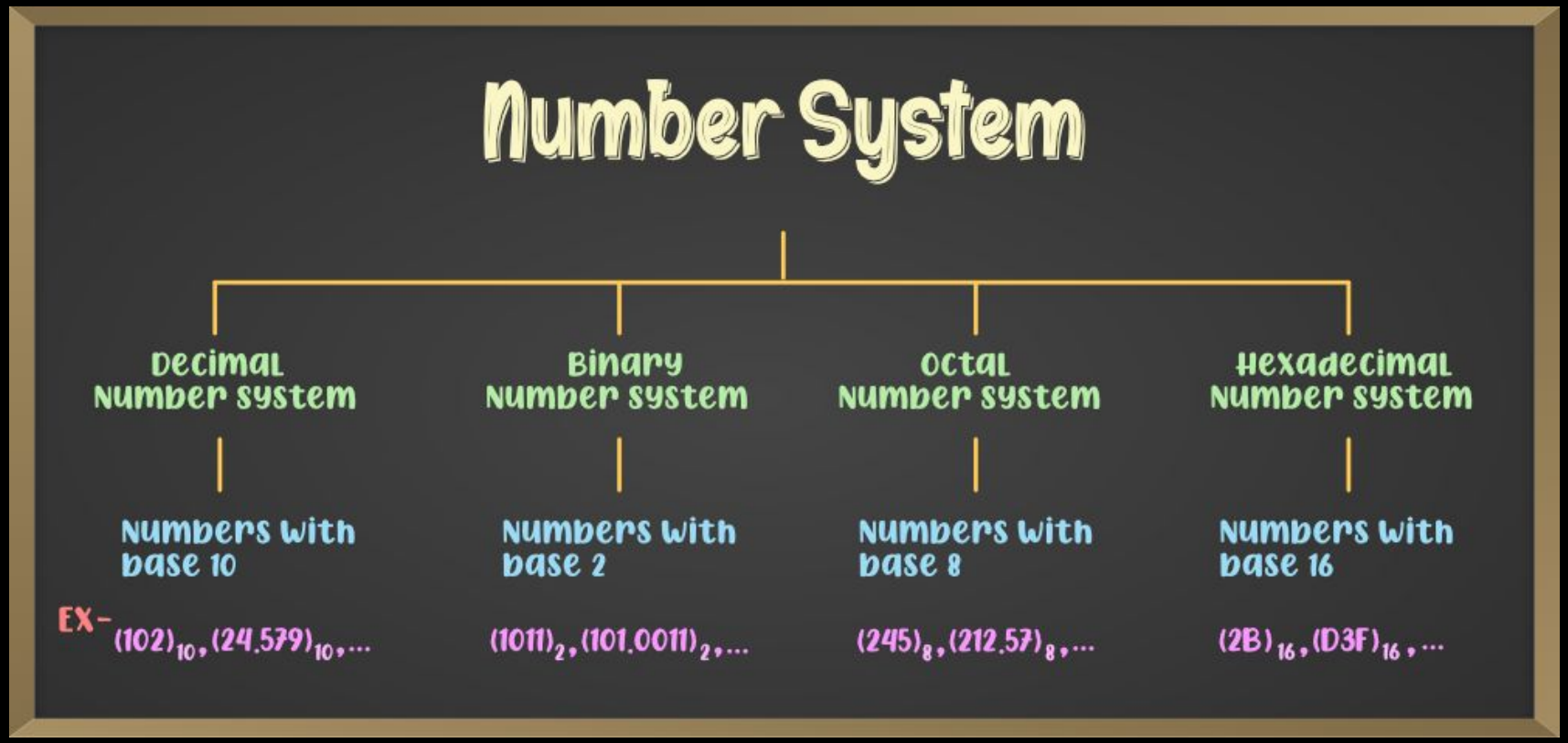
HOW DOES A COMPUTER WORK





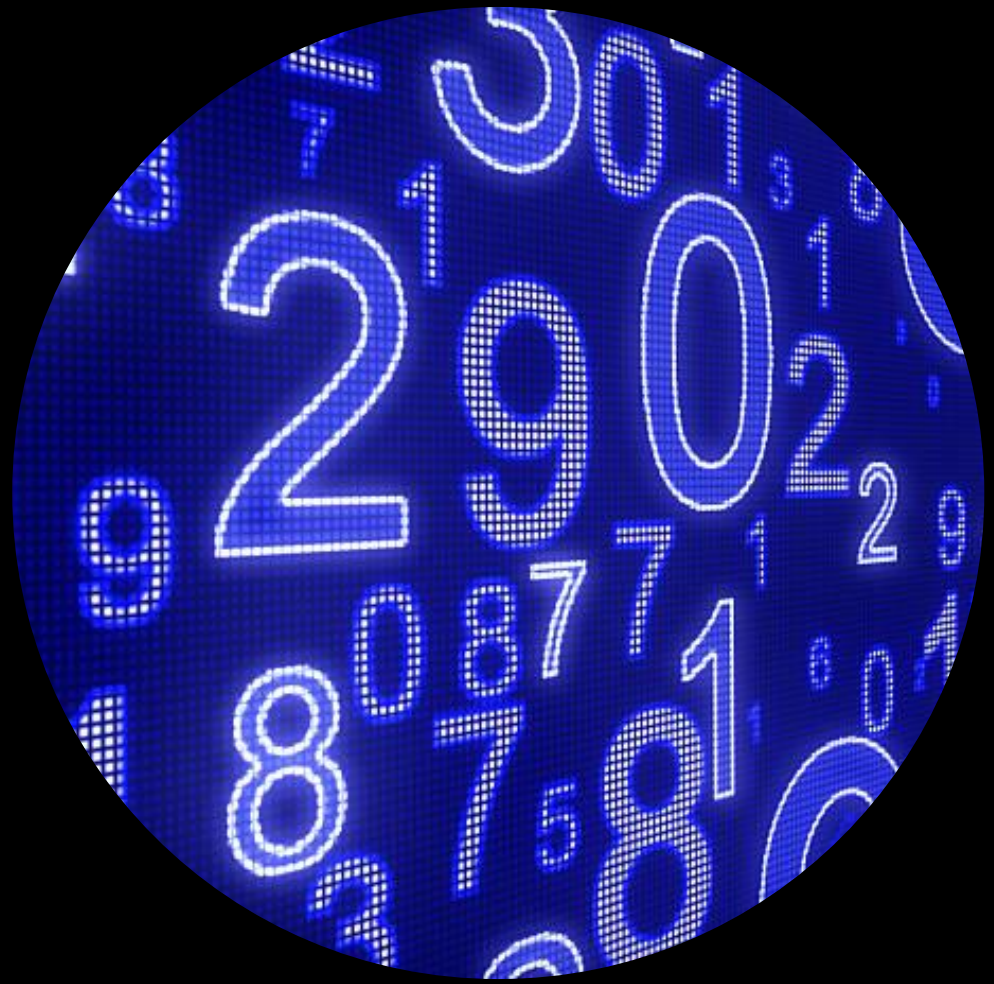
LESSON 2: BINARY AND HEXADECIMAL SYSTEMS

NUMBERING SYSTEMS



DECIMAL NUMBERING SYSTEM

Decimal system, in mathematics, positional numeral system employing 10 as the base and requiring 10 different numerals, the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. It also requires a dot (decimal point) to represent decimal fractions.



BINARY NUMBERING SYSTEM

- The binary system is a base-2 numeral system using two symbols: 0 and 1. It is the foundation of all binary code, crucial for encoding data electronically.
- Binary Digits (Bits): 0 and 1
  - Bit: The smallest unit of data in computing, representing a single binary digit (0 or 1).
  - Byte: A group of 8 bits.

CONVERTING DECIMAL NUMBERS TO BINARY

- To convert a decimal number to binary:
  1. Divide the number by 2.
  2. Record the remainder.
  3. Repeat with the quotient until you reach 0.
  4. Read the remainders in reverse order.

CONVERTING DECIMAL NUMBERS TO BINARY

- **Example**: Convert 13 to binary:
  1. 13 ÷ 2 = 6, remainder 1
  2. 6 ÷ 2 = 3, remainder 0
  3. 3 ÷ 2 = 1, remainder 1
  4. 1 ÷ 2 = 0, remainder 1
  5. Result: 1101

OCTAL NUMBERING SYSTEM

A number system with its base as 'eight' is known as an Octal number system and uses numbers from 0 to 7 i.e., 0, 1, 2, 3, 4, 5, 6, and 7. We can take an example, to understand the concept better. Like we already know, any number that has a base 8 is called an octal number like 248, 1098, 558, etc.

CONVERTING BINARY NUMBERS TO OCTAL

Steps to Convert Binary to Octal

1. **Group the binary digits in sets of three**, starting from the right. If the number of digits is not a multiple of three, add extra zeros to the left (most significant bit) to form complete groups.
2. **Convert each group of three binary digits to its equivalent octal digit**.
3. **Combine the octal digits** to form the final octal number.

CONVERTING BINARY NUMBERS TO OCTAL

Steps to Convert Binary to Octal

1. **Group the binary digits in sets of three**, starting from the right. If the number of digits is not a multiple of three, add extra zeros to the left (most significant bit) to form complete groups.
2. **Convert each group of three binary digits to its equivalent octal digit**.
3. **Combine the octal digits** to form the final octal number.

CONVERTING OCTAL NUMBERS TO BINARY

Steps to Convert Octal to Binary

1. **Convert each octal digit to its equivalent 3-bit binary representation**.
2. **Combine all the binary groups** to form the final binary number.

CONVERTING OCTAL NUMBERS TO BINARY

Example

Let's convert the octal number 732 to its binary equivalent.

Step 1: Convert Each Octal Digit to Binary

`7` in octal = `111` in binary
`3` in octal = `011`in binary
`2` in octal = `010` in binary

HEXADECIMAL NUMBERING SYSTEM

- The hexadecimal system is a base-16 numeral system that uses sixteen symbols: 0-9 and A-F. It is commonly used in computing for a more human-friendly representation of binary-coded values.
- Hexadecimal Digits: 0-9 and A-F
  - Digits 0-9: Represent values 0 to 9.
  - Digits A-F: Represent values 10 to 15.

CONVERTING BINARY TO HEXADECIMAL

- To convert binary to hexadecimal:
  1. Group the binary digits into sets of four from right to left.
  2. Convert each group to its corresponding hexadecimal digit.
- **Example**: Convert 11010111 to hexadecimal:
  1. Group: 1101 0111
  2. Convert: 1101 (D) and 0111 (7)

  Result: D7

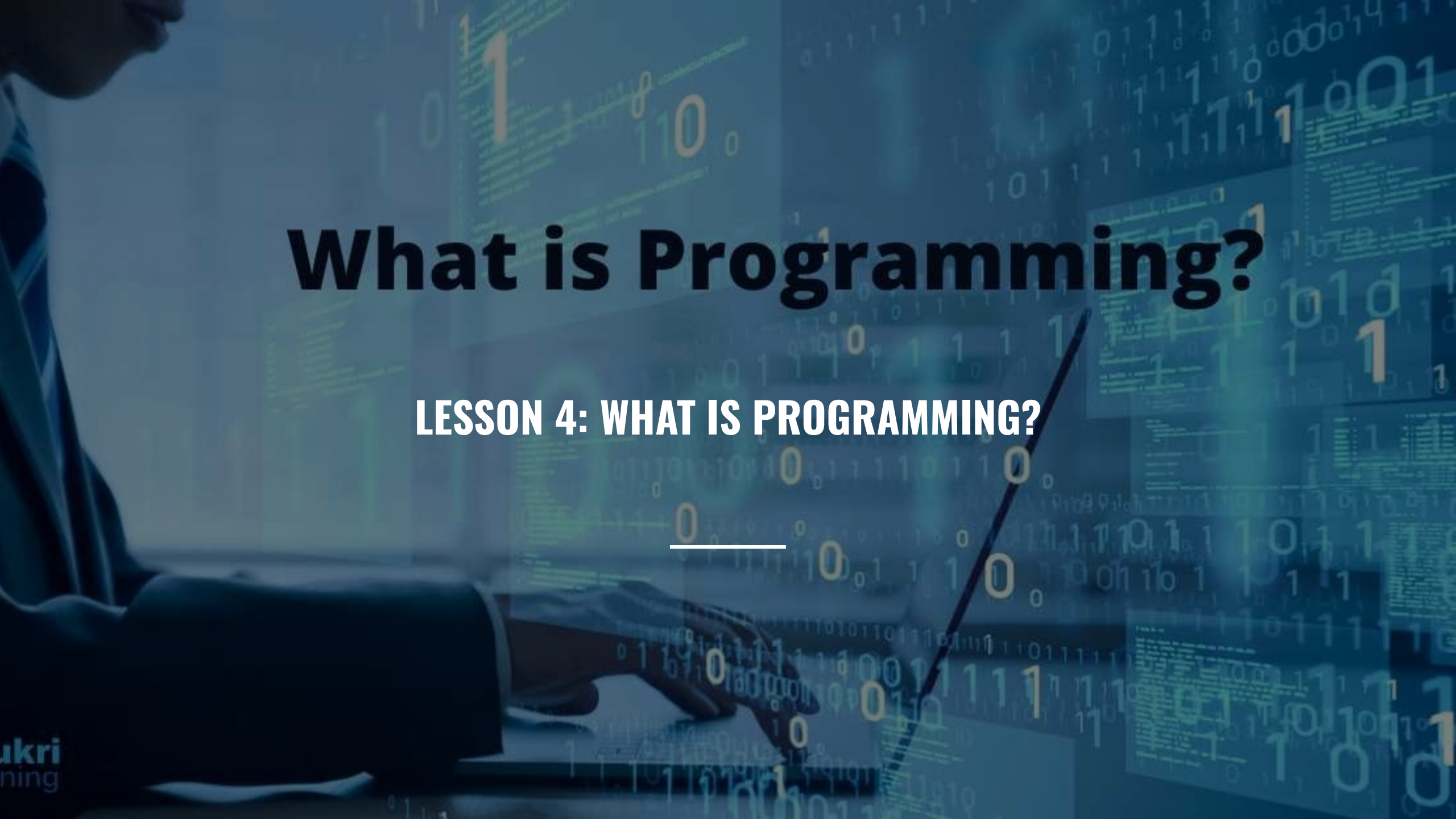
CONVERTING HEXADECIMAL TO BINARY

- To convert hexadecimal to binary:

  Convert each hexadecimal digit to its 4-bit binary equivalent.

  Example: Convert 2F to binary:

1. 2 (0010) and F (1111)
2. Result: 00101111


What is Programming?
LESSON 4: WHAT IS PROGRAMMING?

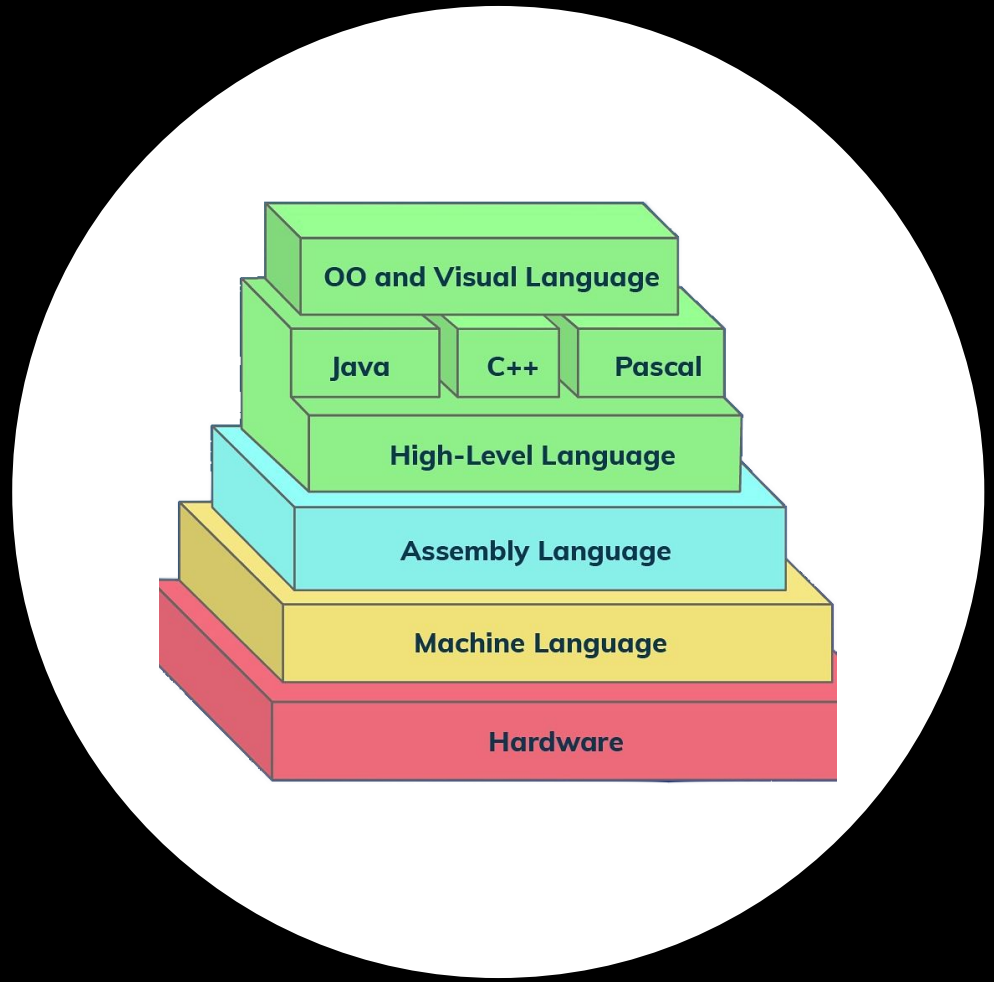
Computer Language Hierarchy

OO and Visual Language: High-level languages that support object-oriented programming and visual development, such as Java, C++, and visual languages.

High-Level Language: More abstract languages closer to human languages, such as Pascal.

Assembly Language: A low-level language that is a step above machine language, using mnemonic codes.

Machine Language: The lowest level of code, directly executed by the hardware, consisting of binary code.



Definition

Low-level languages are closer to the machine's hardware. They provide little or no abstraction from the computer's architecture, meaning they operate with the actual hardware's instructions.

Examples:

- **Machine Code**: The most basic form of code, consisting of binary (1s and 0s) that the computer's CPU can execute directly.
- **Assembly Language**: A step above machine code, it uses symbolic names (mnemonics) instead of binary instructions. Each assembly language instruction corresponds directly to a machine language instruction.

Characteristics:

- **Efficiency**: Programs written in low-level languages are often very fast and efficient because they operate directly with the hardware.
- **Complexity**: Writing in low-level languages is more complex and time-consuming because programmers need to manage the system's memory and understand the hardware's architecture.
- **Portability**: Low-level programs are less portable since they are closely tied to the specific hardware they were written for.

HIGH-LEVEL PROGRAMMING LANGUAGES:

Definition: High-level languages are more abstract, meaning they are further from the hardware and closer to human languages. These languages are designed to be easy to read, write, and maintain.

Example:

- Python
- Java
- C++
- Ruby
- JavaScript

Characteristics:

- Ease of Use: High-level languages are easier to learn and use because they use human-readable syntax and abstract away the complexities of the hardware.

- Portability: Programs written in high-level languages are generally more portable, as they can run on different hardware and operating systems with minimal modification.

- Less Control Over Hardware: While easier to use, high-level languages offer less direct control over hardware, which can lead to less optimized performance in some cases.

Interpreter

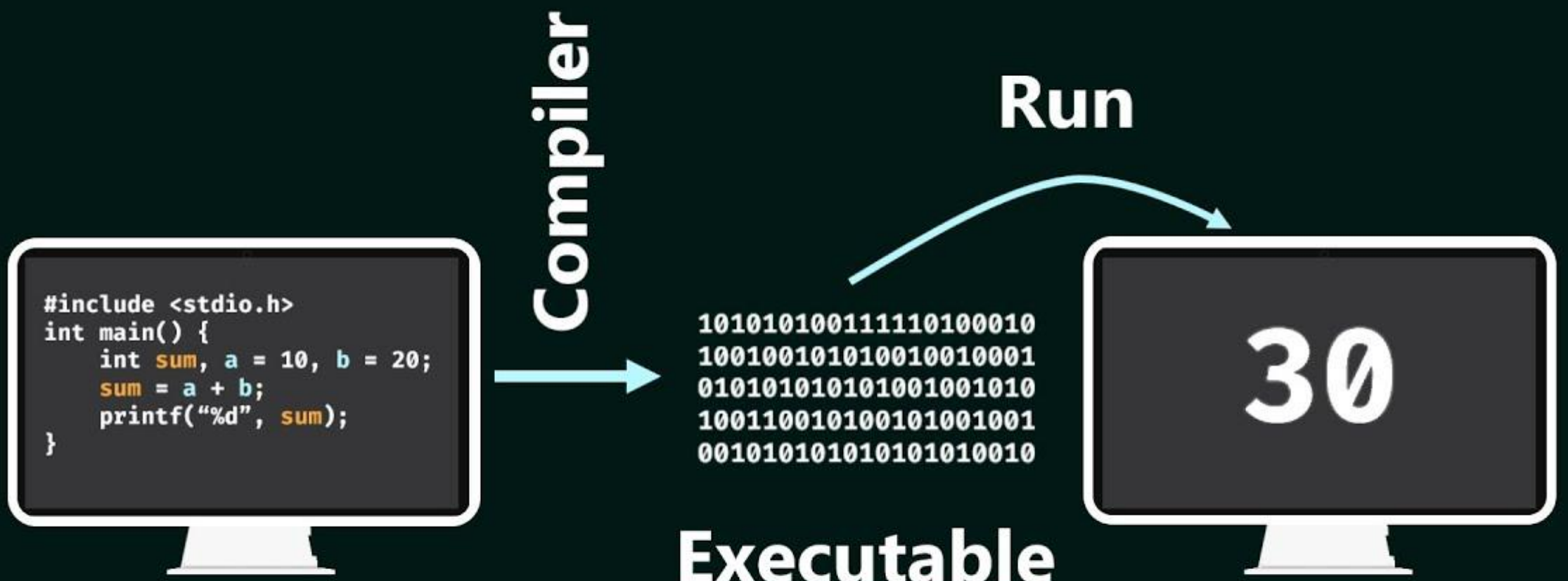
A B C D E F G H I J K L M N O P Q R
S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9
* / > < & # № ? ! @ % = + − $ € £ (. ,)

Vs

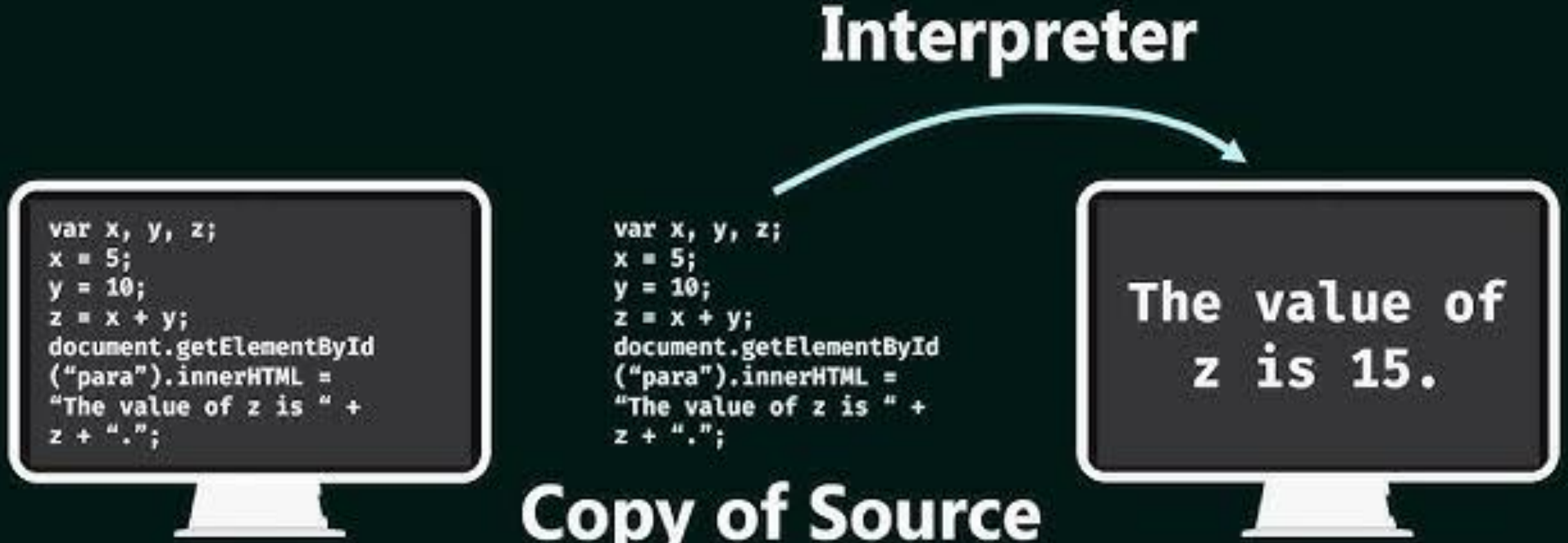
Compiler

A B C D E F G H I J K L M N O P Q R
S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9
* / > < & # № ? ! @ % = + − $ € £ (. ,)

What is a Compiler?



What is an Interpreter?



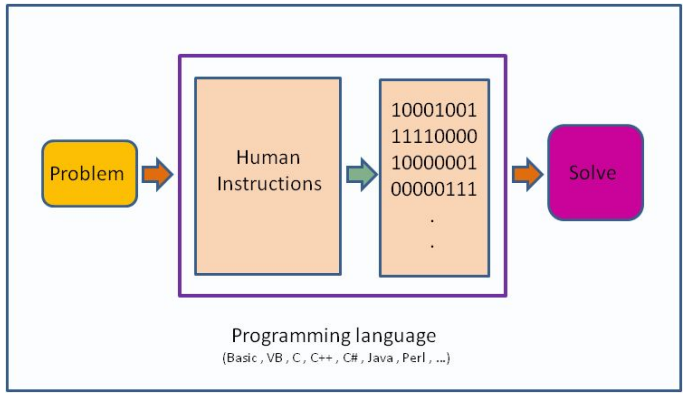
How Compiler Works

How Interpreter Works

DEFINITION OF PROGRAMMING:

Is the process of writing instructions for a computer to perform specific tasks.**Examples of Programming Languages**:

- Python: A high-level, easy-to-read language often used for web development, automation, and data analysis.
- JavaScript: A language commonly used for creating interactive effects within web browsers.
- C++: A powerful language used for system/software development and game programming.



BASIC CONCEPTS:

- **Algorithms:**

  A step-by-step procedure or formula for solving a problem.

  Example: A recipe for baking a cake, which involves specific steps in a particular order.

- **Syntax :**

  The set of rules that defines the combinations of symbols that are considered to be a correctly structured program in a language.

  Example: correct order of writing a sentence

- **Semantics :**

  The meaning of the symbols, expressions, and statements

  Example Hello World

- **Variables :**

  Named storage locations in memory for data that can change during program execution.

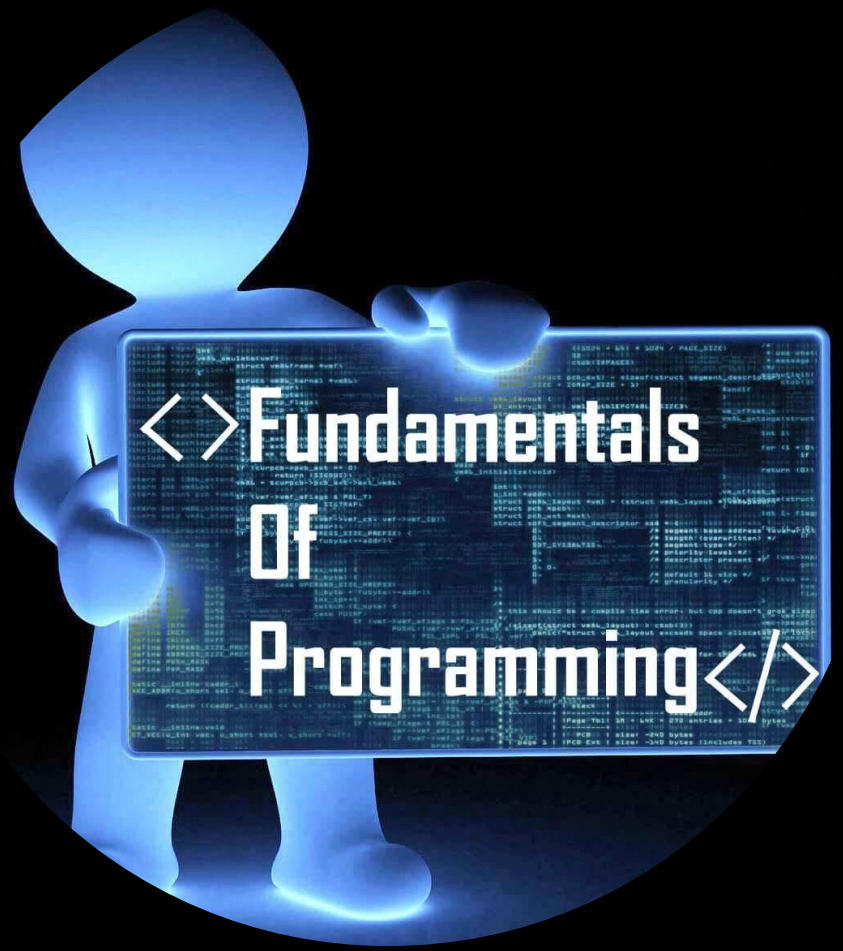
  Example: `x=5` assign 5 to variable me `x` which is a location in the memory

- **Data Types:**

  The kinds of values that variables can hold

  Examples

  - Integer □ whole numbers(1, 2, 4 ...)
  - Float □ decimal or fractional numbers (1.1, 2/6, 4.2 ....)
  - String □ textual data "Hello world"
  - Boolean □ True or False values



BASIC CONCEPTS:

- Control Structures:

Conditional Statements: Allow the program to make decisions based on certain conditions.

Example `if-else`

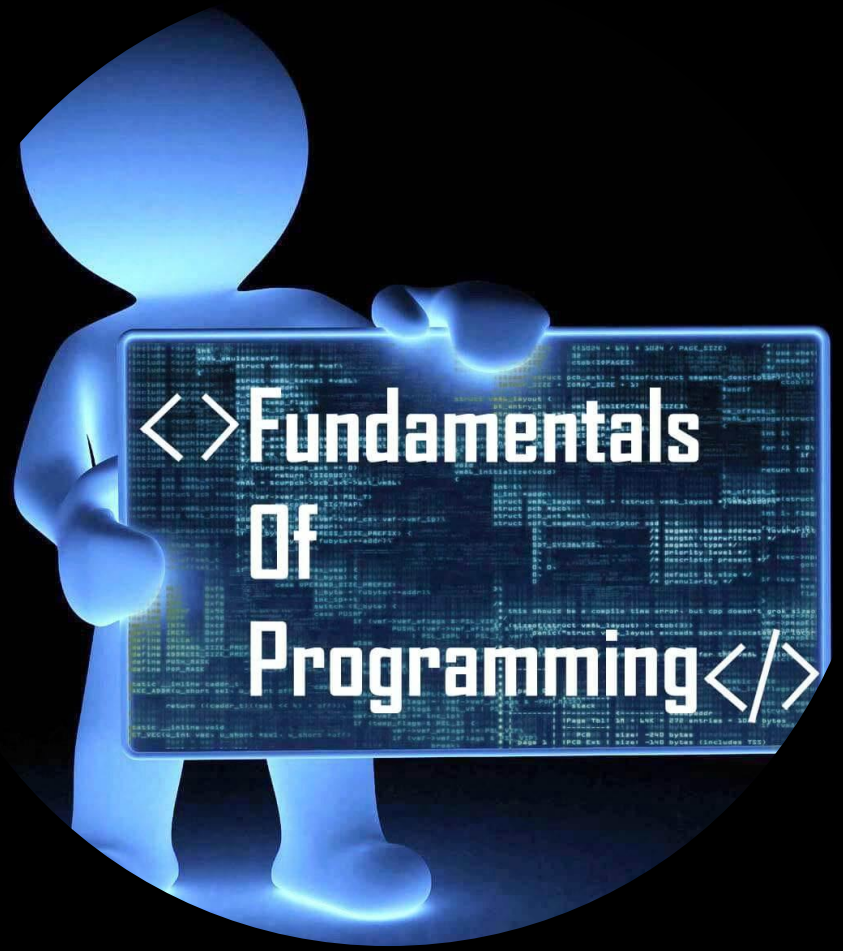
```
if x > 0:
    print("x is positive")
else:
    print("x is non-positive")
```

- Loops:

Conditional Statements: Allow the program to make decisions based on certain conditions.

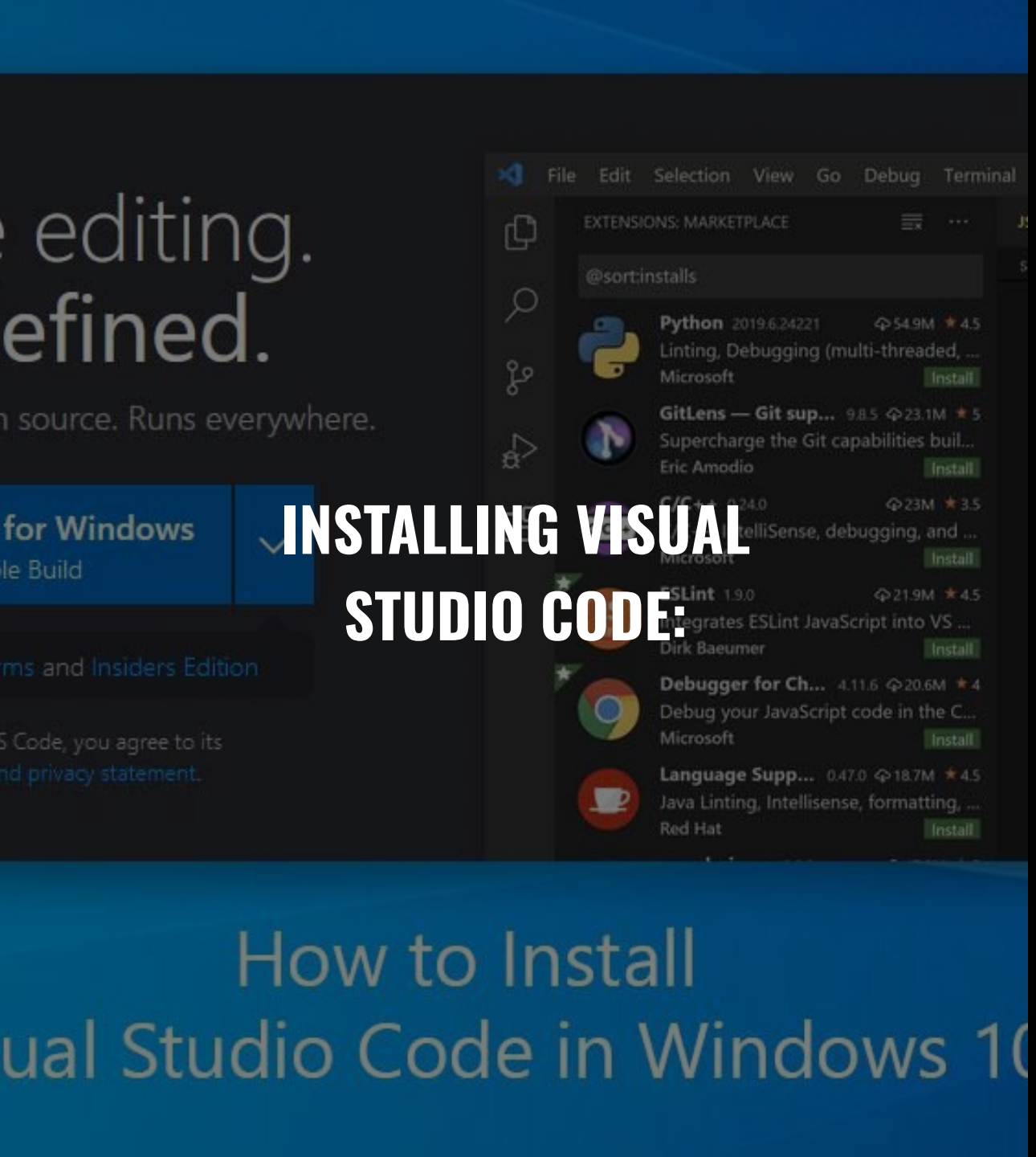
Example `for loop`

```
for i in range(5):
    print(i)
```



Lesson 5: Installing and Using Visual Studio Code (VSCode)

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. It is available for Windows, macOS, and Linux. VS Code is widely used by developers due to its rich feature set and flexibility.



INSTALLING VISUAL STUDIO CODE:

1. Download VSCode:
    1. Go to the official Visual Studio Code website.
    2. Click on the download button appropriate for your operating system (Windows, macOS, or Linux).
2. Install VSCode:
    1. Run the installer you downloaded.
    2. Follow the on-screen instructions to complete the installation.

BASIC FEATURES:

Creating a New File:

Click on File > New File or press Ctrl+N (Cmd+N on macOS)

Saving a File:

Click on File > Save or press Ctrl+S (Cmd+S on macOS).

Choose the file location and name it with an appropriate file extension (e.g., .md for Markdown)

Opening a Folder:

Click on File > Open Folder to open a directory and work on multiple files within that directory.

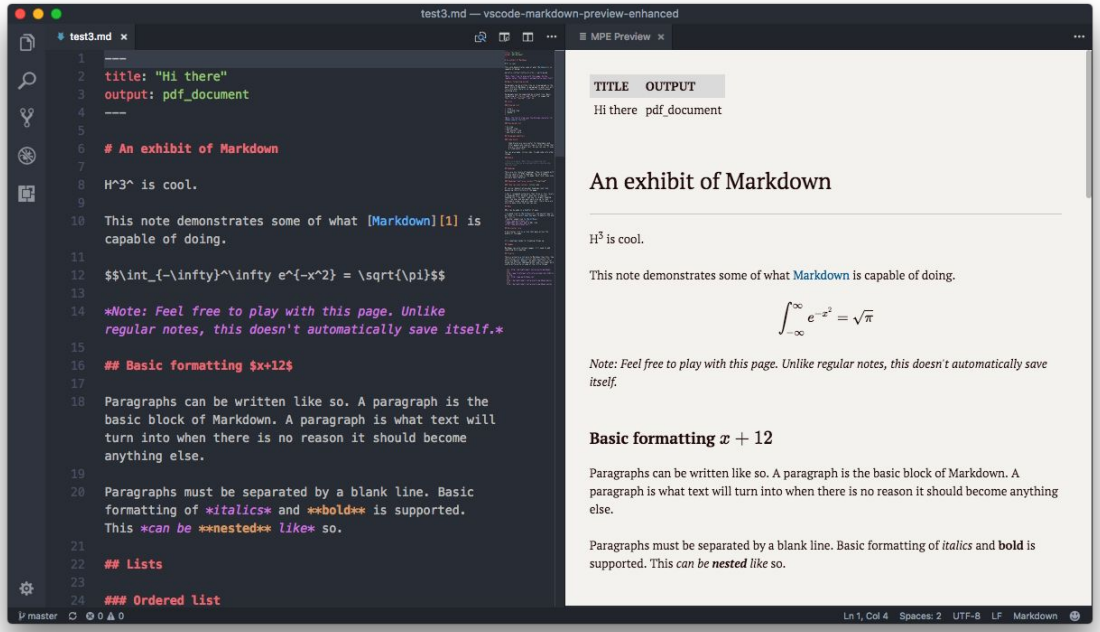
Using the Integrated Terminal:

Open the terminal by clicking on View > Terminal or press Ctrl+ (backtick).

Installing Extensions:

Click on the Extensions icon in the sidebar or press Ctrl+Shift+X (Cmd+Shift+X on macOS).

Search for extensions like Markdown All in One, and click Install to add them.

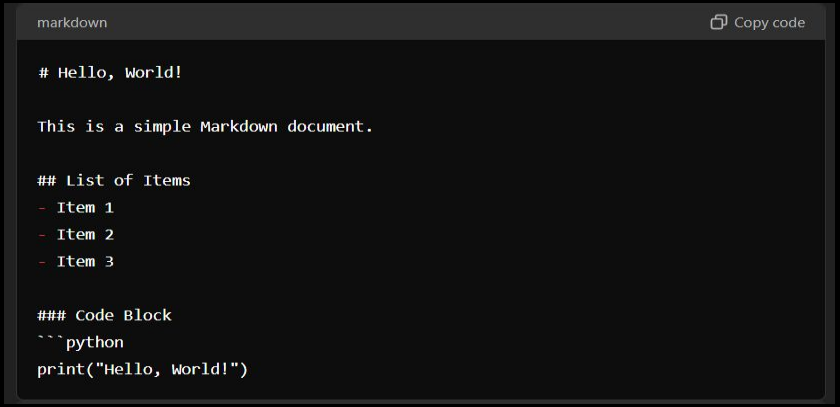


SIMPLE EXAMPLE:

Creating a Simple Markdown Document in VSCode

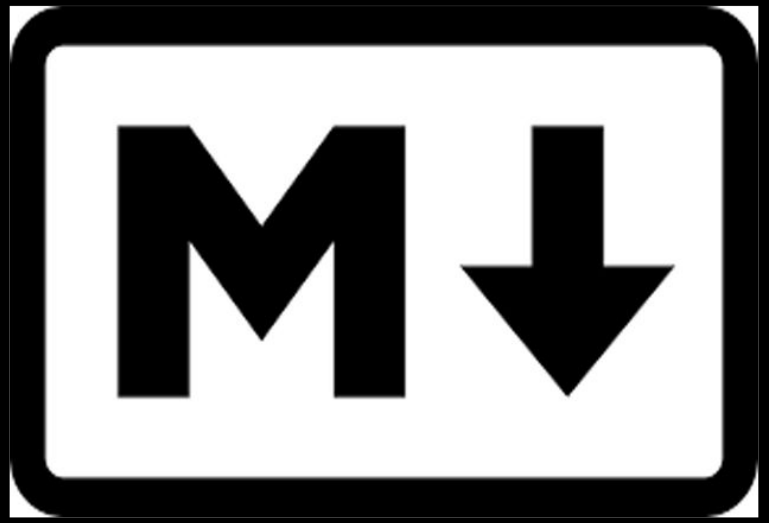
1. **Open VSCode**: Launch the application.

2. **Create a New File**:
   - Click File > New File.
   - Save the file as example.md.
3. **Write a Simple Markdown Document**:
   - Type the following content into the file



4. Preview the Markdown:
   - Click on the markdown file and then on the `Open preview` on the side bar

Lesson 7: Introduction to Markdown



Markdown is a lightweight markup language with plain text formatting syntax. It is designed to be easy to read and write, making it a popular choice for creating formatted text using a plain text editor. Markdown is widely used for writing documentation, web content, notes, and README files in software projects.

BASIC SYNTAX:

Headers:

- Create headers using the # symbol followed by a space

markdown Copy code

```
# This is an H1 header
## This is an H2 header
### This is an H3 header
```

Emphasis:

- Italic: Use * or _ around the tex

markdown Copy code

```
*italicized text* or _italicized text_
```

BASIC SYNTAX:

Emphasis:

Bold: Use ** or __ around the text.

markdown Copy code

```
*italicized text* or _italicized text_
```

Strikethrough: Use ~~ around the text.

markdown Copy code

```
**bold text** or __bold text__
```

Strikethrough: Use ~~ around the text

markdown Copy code

```
~~strikethrough text~~
```

BASIC SYNTAX:

Lists
:

Unordered Lists: Use -, *, or + followed by a space.

markdown Copy code

```
- Item 1
- Item 2
- Item 3
```

Ordered Lists: Use numbers followed by a period and a space.

markdown Copy code

```
1. First item
2. Second item
3. Third item
```

BASIC SYNTAX:

- **Links:**

Inline links: Use [text](URL).

```markdown
[OpenAI](https://www.openai.com)
```

- **Images:**

Use ![alt text](URL) for images.

```markdown
![Markdown Logo](https://markdown-here.com/img/icon256.png)
```

BASIC SYNTAX:

Blockquotes:

Use > before the text.

markdown Copy code

```
> This is a blockquote.
```

Code:

Inline code: Use backticks ` around the code

markdown Copy code

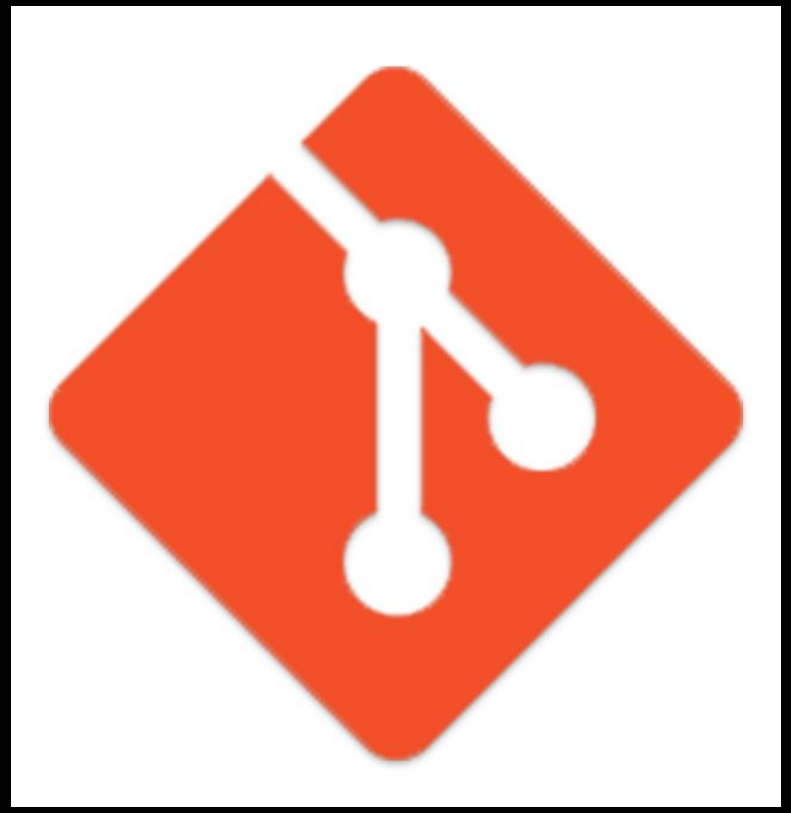
```
`code`
```

SIMPLE MARKDOWN DOCUMENT EXAMPLE:

markdown Copy code

```markdown
# Introduction to Markdown

Markdown is a lightweight markup language for creating formatted text using a plain-text e

## Basic Syntax

### Headers

# H1
## H2
### H3

### Emphasis

*Italic* or _Italic_

**Bold** or __Bold__

~~Strikethrough~~

### Lists

- Unordered
  - Sub-item

1. Ordered
2. List

### Links

[OpenAI](https://www.openai.com)

### Images

![Markdown Logo](https://markdown-here.com/img/icon256.png)

### Blockquotes

> Markdown is a great tool for writing documentation.

### Code

Inline `code`
```

Lesson 6: Introduction to Git and GitHub



Git is a distributed version control system that allows developers to track changes in their code, collaborate with others, and manage project history efficiently. It is widely used in software development for source code management and has become an industry standard due to its robust features and flexibility.

INSTALLING GIT:

- Windows:

1. Download Git:
    1. Go to git-scm.com and download the Windows installer.
2. Run the Installer:
    1. Follow the setup instructions, leaving the default options selected.

WHAT IS GITHUB?

- A web-based platform that uses Git for version control and provides a hosting service for Git repositories.
- Advantages:
  1. Facilitates collaboration by allowing multiple people to work on a project.
  2. Provides a user-friendly interface to manage repositories.
  3. Offers features like pull requests, issues, and project management tools.

CREATING A GITHUB ACCOUNT:

1.Go to GitHub:

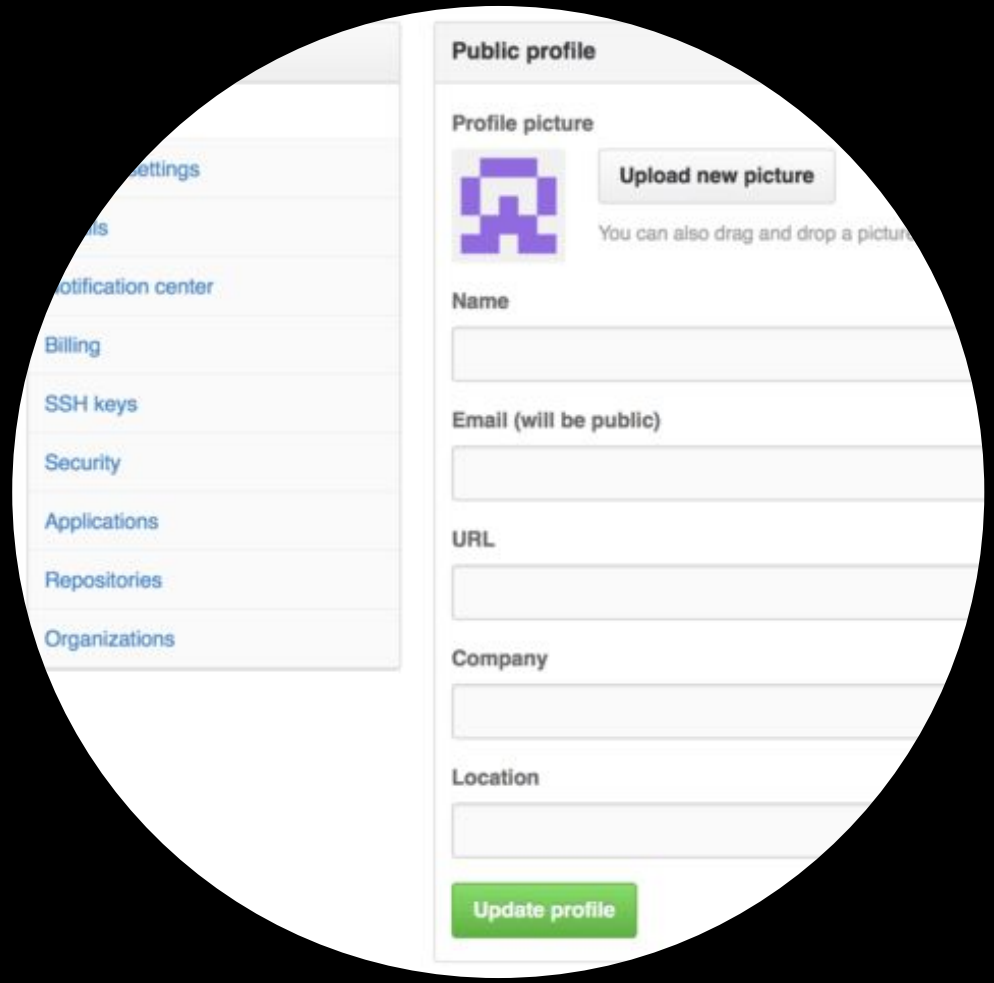
- Open your web browser and go to github.com.

2.Sign Up:

- Click on the Sign up button in the top right corner.
- Enter your email address and click Continue.
- Create a password and click Continue.
- Choose a username and click Continue.
- Follow the on-screen instructions to complete the sign-up process, including solving the puzzle to verify you're not a robot.

3.Verify Email:

- Check your email for a verification message from GitHub.
- Click the verification link in the email to activate your account.



BASIC GIT COMMANDS:

Initial Setup:

Configure Git: Set your username and email

```bash
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Common Commands:

Initializing a Repository: Create a new repository

```bash
git init
```

Cloning a Repository:

Clone an existing repository from GitHub

```bash
git clone https://github.com/username/repository.git
```

BASIC GIT COMMANDS:

Checking Repository Status:

Check the status of your files

bash Copy code

```
git status
```

Adding Changes:

Stage files for commit

bash Copy code

```
git add filename
```

Stage all changes

bash Copy code

```
git add .
```

Committing Changes:

Commit your staged changes

bash Copy code

```
git commit -m "Commit message"
```

BASIC GIT COMMANDS:

Pushing Changes:

Push your commits to the remote repository:

bash Copy code

```
git push origin branch-name
```

Pulling Changes:

Pull the latest changes from the remote repository:

bash Copy code

```
git pull origin branch-name
```

BRANCHES IN GIT:

- What is a Branch?
- **Branch**: A separate line of development in a Git repository. It allows you to work on different features or bug fixes independently of the main codebase.
- Advantages of Using Branches:
  - Isolation: Work on new features or bug fixes without affecting the main codebase.
  - Collaboration: Multiple developers can work on different features simultaneously.
  - Version Control: Easily track changes and manage multiple versions of your project.

BASIC BRANCH COMMANDS:

Creating a New Branch:

```bash
git branch new-branch-name
```

Creating and Switching to a Branch:

```bash
git checkout -b new-branch-name
```

Switching to a Branch:

```bash
git checkout new-branch-name
```

Merging a Branch:

```bash
git checkout main
git merge new-branch-name
```

Deleting a Branch:

```bash
git branch -d new-branch-name
```

USING GITHUB:

Creating a Repository:

1.Create a New Repository:

- Go to GitHub.
- Click the + icon in the top right corner and select New repository.
- Fill in the repository name, description, and choose whether it should be public or private.
- Click Create repository.

Collaborating on GitHub:

1.Forking a Repository:

- Go to the repository you want to fork.
- Click the Fork button in the top right corner.

2.Creating a Pull Request:

- Make changes to your forked repository.
- Navigate to the original repository.
- Click New pull request and follow the instructions to propose your changes.

3.Managing Issues:

- Navigate to the Issues tab in your repository.
- Click New issue to report a bug or request a feature.