Republic of the Philippines
Isabela State University
Echague, Isabela

College of Computing Studies, Information and Communication Technology

Chapter I.

Overview of Platform Technologies

1. Definition and Importance

Definition: Platform technologies refer to the foundational technologies that support and enable the development, deployment, and management of software applications and services. They provide the underlying infrastructure that applications run on and can include both hardware and software components.

Importance:

- **Foundation for Applications:** Platforms provide the base on which applications are built and executed. Without them, software development would not be possible.
- **Integration:** They offer mechanisms for integrating various software components and systems, facilitating communication and data exchange.
- **Scalability:** Platforms often include tools and services that help applications scale efficiently as user demands grow.
- **Productivity:** They streamline development processes by offering frameworks, libraries, and tools that accelerate application development and deployment.
- **Innovation:** Platforms enable new technologies and innovations by providing a stable base on which developers can build and experiment.

---

2. Examples of Popular Platforms

a. Operating Systems (OS):

Objective:

- Understand the role and functions of operating systems.
- Identify and differentiate between popular operating systems and their unique features.

Definition: Operating systems (OS) are crucial software that manage and coordinate the hardware and software resources of a computer. They provide the necessary environment for other applications to run and include essential functionalities such as memory management, process management, and user interface. The OS handles interactions between hardware components and software applications, ensuring efficient operation of the system.

Examples:

- **Windows:** A widely used OS known for its user-friendly interface and extensive support for various applications and games.
- **macOS:** An OS developed by Apple, optimized for Apple's hardware, known for its stability and integration with other Apple products.
- **Linux:** An open-source OS used extensively in servers, development, and embedded systems, appreciated for its flexibility and security.

Activity:

- Compare the user interfaces and basic features of Windows and Linux using screenshots or a live demo if possible.

b. Cloud Computing Platforms:

Objective:

- Comprehend the concept and benefits of cloud computing platforms.
- Identify major cloud services and their typical use cases.

Definition: Cloud computing platforms offer on-demand access to computing resources and services over the internet, allowing users to scale resources as needed without managing physical hardware. These platforms provide a range of services including virtual machines, storage, and databases, which can be accessed remotely.

Examples:

- **Amazon Web Services (AWS):** A comprehensive cloud platform providing services like EC2 for computing, S3 for storage, and RDS for databases.
- **Microsoft Azure:** A cloud service offering similar to AWS, with strong integration with Microsoft products and services.
- **Google Cloud Platform (GCP):** Known for its data analytics and machine learning services, in addition to traditional cloud resources.

Activity:

- Explore the dashboards of AWS Free Tier or Google Cloud Free Tier and discuss the basic services offered.

c. Web Development Platforms:

Objective:

- Understand the purpose and functionalities of web development platforms.

- Recognize popular platforms and their primary features for building and managing websites.

Definition: Web development platforms provide the tools and frameworks necessary to create, manage, and deploy websites and web applications. They offer features such as content management, user administration, and customization options to facilitate the development process.

Examples:

- **WordPress:** A popular CMS that allows users to build and manage websites with a user-friendly interface and a vast library of themes and plugins.
- **Drupal:** An open-source CMS known for its flexibility and ability to handle complex websites and applications.
- **Joomla:** A CMS that balances ease of use with advanced functionality, making it suitable for various types of websites.

Activity:

- Demonstrate how to create a simple page using WordPress, highlighting its key features and user interface.

d. Development Platforms:

Objective:

- Identify and understand the tools provided by development platforms for creating software applications.
- Compare different development environments based on their features and supported programming languages.

Definition: Development platforms are environments and tools designed to support the creation, testing, and deployment of software applications. They include integrated development environments (IDEs), code editors, and frameworks that streamline the development process and enhance productivity.

Examples:

- **Visual Studio Code:** A free, open-source code editor with support for multiple programming languages, extensions, and debugging tools.
- **Eclipse:** An IDE primarily used for Java development, with support for other languages through plugins.
- **Android Studio:** The official IDE for Android development, providing comprehensive tools for building and debugging Android applications.

Activity:

- Show how to write and run a simple code snippet in Visual Studio Code, demonstrating its features and capabilities.

e. Virtualization Platforms:

Objective:

- Understand the concept of virtualization and its benefits.
- Learn how to create and manage virtual machines using virtualization platforms.

Definition: Virtualization platforms enable the creation and management of multiple virtual machines (VMs) on a single physical machine. This technology allows users to run different operating systems or applications in isolated environments, optimizing hardware utilization and simplifying management.

Examples:

- **VirtualBox:** A free, open-source virtualization tool that supports creating and managing VMs on various operating systems.
- **VMware Workstation:** A robust virtualization platform offering advanced features for running multiple virtual machines on Windows and Linux.

Activity:

- Demonstrate how to create a new virtual machine using VirtualBox, highlighting key features such as snapshots and resource allocation.

f. Mobile Platforms

Objective:

- Understand the role of mobile platforms in the development and deployment of mobile applications.
- Identify key mobile platforms and their features, and compare their ecosystems.

Definition: Mobile platforms are operating systems designed for mobile devices such as smartphones and tablets. They provide the environment for mobile applications to run and include necessary APIs and services for app development. These platforms handle user interactions, resource management, and connectivity, ensuring apps function properly on mobile devices.

Examples:

- **Android:** Developed by Google, Android is an open-source mobile OS used on a wide range of devices from various manufacturers. It supports a vast ecosystem of apps available

through the Google Play Store and offers extensive customization options for users and developers.
- **iOS:** Developed by Apple, iOS is the mobile OS used exclusively on Apple devices like the iPhone and iPad. Known for its seamless integration with Apple's hardware and services, iOS offers a highly controlled and secure environment for apps, available through the Apple App Store.
- **Windows Mobile:** Although less common now, Windows Mobile was a Microsoft OS designed for mobile devices. It has been largely replaced by Windows 10 Mobile, which is now also phased out in favor of other platforms.

Activity:

- **Comparison of Mobile Platforms:** Use a table or chart to compare Android and iOS based on criteria like user interface, app ecosystems, and development tools.
- **Hands-on Demo:** If possible, demonstrate the basic features of Android Studio or Xcode (for iOS development) and show how to create a simple mobile app.

Visuals and Diagrams

1. **Diagram of Platform Technologies**: Illustrate how different platforms interact with each other, showing the relationship between operating systems, cloud computing, web development platforms, development tools, and virtualization.
2. **Table of Platform Examples**: Create a table listing each type of platform with examples, features, and use cases.

Summary

Platform technologies are essential for supporting and enabling the creation, deployment, and management of software applications. By understanding different types of platforms and their importance, students can appreciate how foundational technologies drive modern computing and development practices.

Activity

1. **Discussion:** Have students discuss the role of different platforms in their daily use of technology and how they interact with these platforms.
2. **Research Assignment:** Assign students to research a platform not covered in class and present its key features and use cases.

Laboratory Activity 1: Lab Orientation and Tools Setup

Objective:

- Familiarize students with the development environment.
- Install and configure Visual Studio Code (VS Code).

Duration:

- 3 hours

Materials Needed:

- Computers with Microsoft Windows installed
- Internet connection
- Visual Studio Code installer
- Basic text editor (e.g., Notepad)

Activity Outline:

1. Introduction to the Development Environment (30 minutes)

a. **Overview of the Development Environment:**

What is an Integrated Development Environment (IDE)?

An **Integrated Development Environment (IDE)** is a software application that provides comprehensive facilities to computer programmers for software development. An IDE typically includes:

1. **Code Editor**: A text editor specifically designed for writing and editing code. It often includes syntax highlighting, code completion, and code folding features.
2. **Compiler/Interpreter**: Tools to translate the code written into a programming language into machine code or bytecode that the computer can execute.
3. **Debugger**: A tool to test and debug code by allowing developers to run their code step-by-step, set breakpoints, and inspect variables.
4. **Build Automation Tools**: Features to automate the process of compiling and linking code, often including tools for managing dependencies.
5. **Version Control Integration**: Tools for integrating with version control systems like Git to manage changes to the source code over time.
6. **Project Management**: Tools for organizing and managing project files, resources, and configurations.

Importance of an IDE in Software Development

1. **Increased Productivity**: IDEs provide a suite of tools in one place, reducing the need to switch between different applications and streamlining the development process.
2. **Error Reduction**: Features like syntax highlighting and code suggestions help catch errors early, improving code quality and reducing debugging time.
3. **Efficiency**: Built-in tools for compiling, debugging, and version control make development tasks more efficient and less error-prone.
4. **Code Management**: IDEs often include features for managing large codebases, including code navigation tools, refactoring capabilities, and project organization.
5. **Learning and Support**: Many IDEs offer extensive documentation, tutorials, and community support, which can be invaluable for both beginners and experienced developers.

Visual Studio Code: A Popular and Versatile IDE

Visual Studio Code (VS Code) is a free, open-source IDE developed by Microsoft. It is known for its versatility and wide range of features:

- **Multi-Language Support**: VS Code supports various programming languages out of the box and can be extended to support additional languages through extensions.
- **Customization**: The IDE is highly customizable, allowing users to modify the editor's appearance and functionality through themes, extensions, and settings.
- **Intelligent Code Assistance**: Features like IntelliSense provide smart code completions based on variable types, function definitions, and imported modules.
- **Integrated Terminal**: Allows developers to run command-line tools and scripts directly within the IDE.
- **Version Control Integration**: Built-in support for Git and other version control systems helps manage source code changes and collaborate with other developers.
- **Debugging**: Powerful debugging tools make it easy to set breakpoints, inspect variables, and step through code to identify and fix issues.

b. **Exploring Visual Studio Code:**

Main Features of Visual Studio Code (VS Code)

1. **Editor Window**:
   - The central area where you write and edit code. You can open multiple files in tabs, allowing for easy navigation and comparison between different pieces of code.
2. **Sidebar**:
   - Located on the left side of the interface, the sidebar includes different views such as the Explorer (for file navigation), Source Control, Extensions, and Debugger. It provides quick access to project files, version control, and other tools.
3. **Status Bar**:
   - Positioned at the bottom of the window, the status bar displays information about the current file and environment, such as the programming language, Git branch,

and the status of any running tasks. It also includes quick access to settings and debugging controls.

4. **Command Palette**:
    - Accessed via the `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS) shortcut, the Command Palette provides a quick way to execute commands and access various features by typing in commands or searching for them.

Extensions and Their Impact on VS Code

Extensions are additional packages that can be installed to enhance or extend the functionality of VS Code. They can significantly improve the development experience by adding new features or supporting new technologies. Here's how they work:

1. **Language Support**: Extensions can add support for additional programming languages, including syntax highlighting, code snippets, and language-specific features. For example, the Python extension adds support for Python development, including linting, debugging, and Jupyter notebook integration.
2. **Code Formatting and Linting**: Extensions provide tools for code formatting and linting, ensuring that code adheres to specific style guidelines and is free from common errors.
3. **Themes and Customization**: Extensions offer new themes and customizations for the editor, allowing users to change the appearance of VS Code to better suit their preferences.
4. **Version Control Integration**: Extensions enhance version control capabilities by integrating with systems like Git, providing features such as advanced diff tools, pull request management, and commit history visualization.
5. **Debugging Tools**: Extensions can add debugging support for different languages or frameworks, including advanced features like conditional breakpoints, remote debugging, and enhanced debugging interfaces.
6. **Productivity Enhancements**: Extensions offer various productivity tools, such as integrated terminal emulators, code snippets for repetitive tasks, and project management utilities.

How to Install Extensions:

- You can install extensions from the Extensions view in the sidebar by searching for them and clicking the "Install" button.
- Once installed, extensions can be configured through settings and preferences to tailor their functionality to your specific needs.

2. Installing Visual Studio Code (30 minutes)

a. **Downloading VS Code:**

1. **Open a Web Browser**:
    - Start by opening your preferred web browser (such as Chrome, Firefox, or Edge).
2. **Go to the Visual Studio Code Download Page**:
    - In the address bar, type https://code.visualstudio.com/ and press `Enter`.

3. **Locate the Download Button**:
   - On the Visual Studio Code homepage, you'll see a prominent "Download" button. Click on it.
4. **Choose the Correct Version for Windows**:
   - You will be directed to the download page where different versions for various operating systems are listed.
   - Look for the section labeled **"Stable Build"** and select the version for **Windows**. It should be automatically detected and offer a direct download link for Windows.
5. **Download the Installer**:
   - Click on the download button for Windows (typically labeled as "64-bit" or "System Installer"). This will download the `.exe` installer file to your computer.
6. **Run the Installer**:
   - Once the download is complete, locate the installer file in your downloads folder or the location where your browser saved it.
   - Double-click the `.exe` file to start the installation process.
7. **Follow the Installation Prompts**:
   - Follow the on-screen instructions to complete the installation of Visual Studio Code. You can choose default settings or customize them based on your preferences.
8. **Launch Visual Studio Code**:
   - After installation, you can find Visual Studio Code in your Start Menu. Click on it to open and start using the IDE.

b. **Running the Installer:**

Running the Installer

1. **Locate the Installer File**:
   - After downloading, find the `.exe` file for Visual Studio Code in your downloads folder or the location where your browser saved it.
2. **Start the Installation**:
   - Double-click the `.exe` file to begin the installation process. You may see a User Account Control (UAC) prompt asking for permission to make changes to your device. Click **Yes** to proceed.
3. **Welcome Screen**:
   - The Visual Studio Code Setup Wizard will open. Click **Next** to continue.

Installation Options

4. **License Agreement**:
   - Read the license agreement and select **I accept the agreement** if you agree to the terms. Click **Next**.
5. **Select Destination Folder**:
   - Choose the folder where you want to install Visual Studio Code. The default location is usually fine for most users. Click **Next**.
6. **Choose Start Menu Folder**:

   - Select a folder for shortcuts in the Start Menu. You can use the default folder or create a new one. Click **Next**.
7. **Select Additional Tasks**:
   - You will be presented with additional tasks to customize your installation. Here are some key options:
     - **Add "Open with Code" Action to Windows Explorer File Context Menu**: This option adds an option to open files and folders directly with VS Code from the Windows File Explorer context menu.
     - **Add "Open with Code" Action to Windows Explorer Directory Context Menu**: Adds an option to open directories with VS Code.
     - **Add VS Code to PATH**: Adds VS Code to your system PATH environment variable, allowing you to run `code` from the command line.
     - **Register Code as an Editor for Supported File Types**: Sets VS Code as the default editor for certain file types.
     - **Create a Desktop Icon**: Adds a shortcut to VS Code on your desktop for easy access.
   - Select the options you want by checking the corresponding boxes. It's generally a good idea to add VS Code to the PATH and create a desktop icon for convenience. Click **Next**.
8. **Ready to Install**:
   - Review your selections. If everything looks good, click **Install** to begin the installation process.
9. **Complete Installation**:
   - The installer will copy files and set up VS Code according to your selections. Once the installation is complete, you’ll see a completion screen. Check the box **"Launch Visual Studio Code"** if you want to start VS Code immediately, then click **Finish**.

Post-Installation

- **Launch VS Code**: You can now launch Visual Studio Code either from the Start Menu or by using the desktop shortcut if you created one.
- **First-Time Setup**: When you first open VS Code, you might be prompted to install additional components or extensions. You can choose to install these based on your needs.

c. **Completing the Installation**

Opening and Exploring the Welcome Screen

1. **Launch Visual Studio Code**:
   - Double-click the VS Code icon on your desktop or find it in the Start Menu to open it.
2. **Welcome Screen Overview**:
   - When you first open VS Code, you should see the **Welcome** screen. This screen provides a helpful introduction and quick access to key features.
   - **Key Sections of the Welcome Screen**:

        - **Get Started**: Links to basic tutorials and documentation to help you get started with VS Code.
        - **Recent Projects**: A list of recently opened folders and files.
        - **Welcome Tour**: An interactive tour that introduces you to the main features of VS Code.
3. **Close the Welcome Screen**:
    - Once you are familiar with the welcome screen, you can close it by clicking the **X** in the upper-right corner or using the command `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS) and typing "Welcome: Close Welcome Page" to close it.

Accessing Basic Settings and Preferences

1. **Open Settings**:
    - Click on the **gear icon** in the lower-left corner of the VS Code window to open the **Settings** menu. Alternatively, you can open the Settings by pressing `Ctrl+,` (Windows/Linux) or `Cmd+,` (macOS).
2. **Explore Settings**:
    - The **Settings** menu will open in a new tab. Here you can see and adjust various settings for VS Code.
    - **Categories**: Settings are organized into categories such as **User**, **Workspace**, and **Folder**. User settings apply globally, while workspace settings are specific to the current project.
    - **Search Bar**: Use the search bar at the top to quickly find specific settings.
3. **Modify Settings**:
    - **Basic Settings**: You can adjust settings like font size, theme, and editor behavior.
        - For example, to change the theme, search for "Color Theme" and choose a different one from the dropdown list.
        - To change the font size, search for "Font Size" and adjust the value.
4. **Access Settings JSON**:
    - For more advanced settings, you can directly edit the `settings.json` file.
    - Click the **Open Settings (JSON)** icon in the top right of the Settings tab or use the command `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS), type "Preferences: Open Settings (JSON)", and press `Enter`.
    - This file allows you to manually edit settings in JSON format.
5. **Keyboard Shortcuts**:
    - To view or customize keyboard shortcuts, click on the **gear icon** in the lower-left corner and select **Keyboard Shortcuts**. Alternatively, you can use the command `Ctrl+K Ctrl+S` (Windows/Linux) or `Cmd+K Cmd+S` (macOS).
    - You can search for specific commands and assign custom shortcuts if needed.
6. **Extensions**:
    - To enhance your development environment, you can install extensions. Click on the **Extensions view icon** in the sidebar (or press `Ctrl+Shift+X` on Windows/Linux or `Cmd+Shift+X` on macOS).
    - Search for and install extensions that fit your needs, such as language support, debuggers, or productivity tools.

3. Configuring Visual Studio Code (1 hour)

a. Initial Configuration:

Configuring Basic Settings in Visual Studio Code

1. Change the Theme

1. **Open Settings**:
   - Click the **gear icon** in the lower-left corner of VS Code and select **Settings**. Alternatively, press `Ctrl+,` (Windows/Linux) or `Cmd+,` (macOS).
2. **Search for Themes**:
   - In the search bar at the top of the Settings tab, type “Color Theme” to find theme-related settings.
3. **Choose a Theme**:
   - Click on the dropdown menu under **Color Theme** and select a theme from the list. VS Code will apply the theme immediately.

2. Adjust Font Size

1. **Open Settings**:
   - Follow the same steps as above to open the **Settings** menu.
2. **Search for Font Size**:
   - Type “Font Size” in the search bar.
3. **Change Font Size**:
   - You will see a setting called **Editor: Font Size**. Adjust the value to change the font size used in the editor. For example, set it to `14` or `16` to increase the size.
4. **Save Changes**:
   - Changes are saved automatically. Close the settings tab once you're done.

Setting Up a Workspace

1. Create a New Folder

1. **Open File Explorer**:
   - Use Windows File Explorer (or Finder on macOS) to navigate to the location where you want to create a new project folder.
2. **Create a New Folder**:
   - Right-click in the desired location and select **New > Folder**. Name the folder according to your project or workspace needs (e.g., `MyProject`).

2. Open VS Code and Create a Workspace

1. **Open VS Code**:
   - Launch Visual Studio Code.
2. **Open the Folder**:

   - Click on **File** in the top menu and select **Open Folder**.
   - Navigate to the folder you created earlier, select it, and click **Select Folder** (or **Open** on macOS).
3. **Set Up Workspace**:
   - Once the folder is opened in VS Code, you'll see it listed in the **Explorer** sidebar on the left. This folder is now your workspace, and you can start adding files and organizing your project here.
4. **Save Workspace Configuration**:
   - To save the current workspace setup, go to **File** > **Save Workspace As...**. Choose a location and name for the workspace file, which will save the current folder and any specific settings or configurations you have.

Summary

1. **Configure Basic Settings**:
   - Change the theme and adjust the font size to suit your preferences through the Settings menu.
2. **Set Up a Workspace**:
   - Create a new folder using File Explorer and then open it in VS Code to start working on your project. This folder becomes your workspace where you can manage all related files.

b. Installing Essential Extensions:

Installing Essential Extensions in Visual Studio Code

1. Open the Extensions View

1. **Launch VS Code**:
   - Open Visual Studio Code if it's not already running.
2. **Access the Extensions View**:
   - Click on the **Extensions view icon** in the sidebar on the left side of the window (it looks like four squares with one slightly detached). Alternatively, press `Ctrl+Shift+X` (Windows/Linux) or `Cmd+Shift+X` (macOS).

2. Search for and Install Extensions

1. **Search for an Extension**:
   - In the Extensions view, type the name of the extension you want to install in the search bar at the top.
2. **Install the Extension**:
   - Find the desired extension in the search results. For example, to install the Python extension:
     - Type "Python" in the search bar.
     - Click on the **Python** extension by Microsoft.
     - Click the **Install** button.

3. **Repeat for Other Extensions**:
    - Similarly, you can search for and install other essential extensions:
        - **C++ Extension**:
            - Search for "C++" or "C/C++".
            - Install the **C/C++** extension by Microsoft.
        - **HTML/CSS Extension**:
            - Search for "HTML" or "CSS".
            - Install extensions like **HTML Snippets** or **CSS Peek** to enhance HTML and CSS development.

Recommended Extensions Based on Course Requirements

1. Python Development

- **Python**: Provides support for Python programming, including IntelliSense, linting, and debugging.
- **Pylint**: A linter for Python code to help find and fix errors.
- **Jupyter**: For working with Jupyter Notebooks within VS Code.

2. C++ Development

- **C/C++**: Offers support for C and C++ development, including IntelliSense, debugging, and code navigation.
- **CMake Tools**: For working with CMake-based projects and build systems.

3. Web Development

- **HTML Snippets**: Provides useful HTML code snippets to speed up development.
- **CSS Peek**: Allows you to quickly view CSS definitions from your HTML files.
- **Prettier - Code Formatter**: An extension to format code consistently across different languages and files.

How to Install Recommended Extensions

1. **Open Extensions View**:
    - Click on the Extensions view icon in the sidebar or use `Ctrl+Shift+X` / `Cmd+Shift+X`.
2. **Search and Install**:
    - Type the name of the recommended extension in the search bar.
    - Click on the extension from the search results.
    - Click the **Install** button.
3. **Verify Installation**:
    - After installation, you should see the extension listed in the Extensions view under **Installed**.
    -

Summary

1. **Access Extensions View**:
   - Click the Extensions view icon or use the shortcut `Ctrl+Shift+X` / `Cmd+Shift+X`.
2. **Install Extensions**:
   - Search for extensions like **Python**, **C/C++**, **HTML Snippets**, and **Prettier**, and click **Install**.
3. **Recommended Extensions**:
   - Install extensions relevant to your course requirements to enhance your development environment and productivity.

c. Exploring VS Code Features:

Exploring VS Code Features

1. Opening, Editing, and Saving Files

1. **Open a File**:
   - **Using the File Menu**: Click on **File** in the top menu and select **Open File**. Navigate to the file you want to open and click **Open**.
   - **Using the Explorer Sidebar**: In the **Explorer** sidebar, navigate to the file you want to open. Click on it to open it in the editor window.
2. **Edit a File**:
   - Click inside the editor window where your file is open. You can now make changes to the file. VS Code provides features like syntax highlighting, code suggestions, and autocomplete to assist with editing.
3. **Save a File**:
   - **Using the File Menu**: Click **File** and select **Save** or **Save As** to save the current file or save a copy with a new name.
   - **Using the Shortcut**: Press `Ctrl+S` (Windows/Linux) or `Cmd+S` (macOS) to quickly save changes to the current file.

2. Using the Integrated Terminal

1. **Open the Integrated Terminal**:
   - Click on the **Terminal** menu in the top menu bar and select **New Terminal**. Alternatively, you can use the shortcut `Ctrl+` (Windows/Linux) or `Cmd+` (macOS) to open the terminal.
2. **Run Commands**:
   - The terminal will open at the bottom of the VS Code window. You can type and run commands just like you would in a regular command-line interface.
   - For example, type `python --version` to check the installed Python version or `npm install` to install Node.js packages.
3. **Use Multiple Terminals**:
   - You can open multiple terminals by clicking the **+** icon in the terminal panel. Each terminal tab can run different commands simultaneously.

4. **Terminal Customization**:
   - Right-click inside the terminal to access options like **Clear** or **Copy**. You can also adjust terminal settings from the Settings menu (search for "terminal").

3. Basic Debugging Features

1. **Set Breakpoints**:
   - Open the file containing the code you want to debug. Click in the gutter (left margin) next to the line number where you want to set a breakpoint. A red dot will appear indicating a breakpoint.
2. **Start Debugging**:
   - Click on the **Run and Debug** icon in the sidebar (or use the shortcut `Ctrl+Shift+D` on Windows/Linux or `Cmd+Shift+D` on macOS).
   - Click the **Run** button (green triangle) to start debugging. VS Code will execute your code and pause at any breakpoints you've set.
3. **Debug Console**:
   - Use the **Debug Console** at the bottom to view output, inspect variables, and execute debugging commands. You can type expressions and evaluate them in real-time.
4. **Inspect Variables**:
   - When the debugger pauses execution at a breakpoint, you can hover over variables to see their current values or use the **Variables** pane in the debug view to inspect and modify variable values.
5. **Step Through Code**:
   - Use the debugging controls to step through your code:
     - **Continue (F5)**: Resume execution until the next breakpoint.
     - **Step Over (F10)**: Execute the current line of code and move to the next line.
     - **Step Into (F11)**: Enter into the function called at the current line.
     - **Step Out (Shift+F11)**: Exit the current function and return to the caller.

Summary

1. **Open, Edit, and Save Files**:
   - Use the File menu or Explorer sidebar to open files. Edit directly in the editor and save with `Ctrl+S` / `Cmd+S`.
2. **Use the Integrated Terminal**:
   - Open the terminal via the Terminal menu or shortcut. Run commands and manage multiple terminals if needed.
3. **Basic Debugging**:
   - Set breakpoints, start debugging, and use the Debug Console to inspect variables and control execution.

4. Practice Exercise (1 hour)

a. **Creating a Simple Project:**

Creating a Simple Project in Visual Studio Code

1. Creating a Simple HTML File

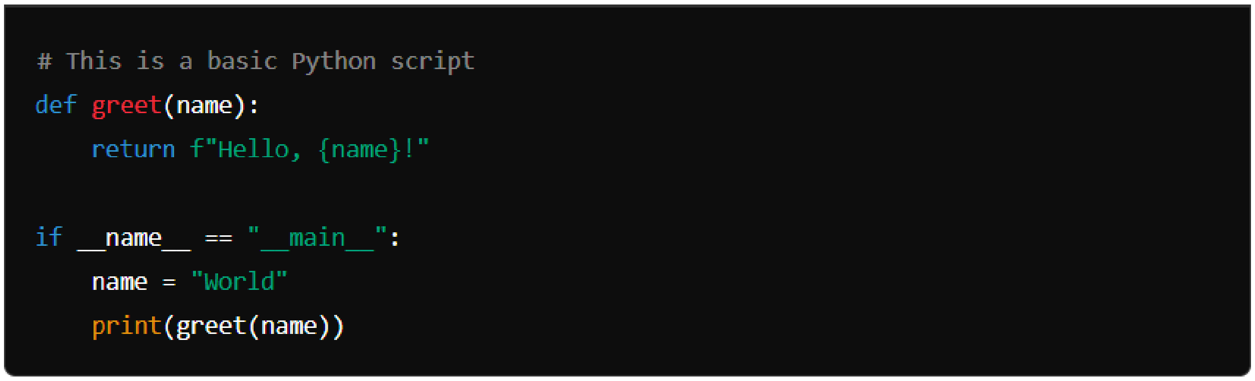
1. **Open VS Code**:
   - Launch Visual Studio Code if it's not already open.
2. **Create a New Folder for Your Project**:
   - Go to **File** > **Open Folder** and select the folder where you want to create your project. If you haven't created a folder yet, create one in File Explorer and then open it in VS Code.
3. **Create a New HTML File**:
   - In the Explorer sidebar, right-click on the folder and select **New File**.
   - Name the file `index.html` and press `Enter`.
4. **Add Basic HTML Content**:
   - Open `index.html` and add the following basic HTML content:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My First HTML Page</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a simple HTML file.</p>
</body>
</html>
```

1. **Practice Using VS Code Features**:
   - **Syntax Highlighting**: Notice that VS Code automatically highlights HTML tags and attributes.
   - **Code Formatting**: Press `Shift+Alt+F` (Windows/Linux) or `Shift+Option+F` (macOS) to format your HTML code.
   - **Live Preview**: If you have the Live Server extension installed, right-click in the editor and select **Open with Live Server** to view your HTML page in a browser.
   -

2. Creating a Basic Python Script

1. **Create a New Python File**:
   - In the Explorer sidebar, right-click on the folder and select **New File**.
   - Name the file `script.py` and press `Enter`.
2. **Add Basic Python Content**:
   - Open `script.py` and add the following Python code:



```
# This is a basic Python script
def greet(name):
    return f"Hello, {name}!"

if __name__ == "__main__":
    name = "World"
    print(greet(name))
```

1. **Practice Using VS Code Features**:
   - **Syntax Highlighting**: Observe how Python keywords and syntax are highlighted.
   - **Code Formatting**: Press `Shift+Alt+F` (Windows/Linux) or `Shift+Option+F` (macOS) to format your Python code. If this doesn't work, ensure you have the Python extension installed and configured.
   - **Running Code**:
     - **Open Integrated Terminal**: Press `Ctrl+` (Windows/Linux) or `Cmd+` (macOS) to open the integrated terminal.
     - **Run the Python Script**: In the terminal, type `python script.py` and press `Enter` to run your Python script. Ensure Python is installed and added to your system PATH.

Summary

1. **Create an HTML File**:
   - Open or create a folder in VS Code, then create a file named `index.html`. Add basic HTML content, practice syntax highlighting, code formatting, and use Live Server if available.
2. **Create a Python Script**:
   - Create a file named `script.py` in the same folder. Add simple Python code, use syntax highlighting, format the code, and run the script from the integrated terminal.

b. **Review and Troubleshooting:** - Time for students to explore the IDE on their own and address any issues they encounter. - Answer any questions and assist with troubleshooting installation or configuration problems.

5. Wrap-Up and Reflection (30 minutes)

a. **Review Key Points:**

Recap of Key Features of Visual Studio Code (VS Code)

1. Editor Window

- **Purpose**: The main area where you write and edit code.
- **Features**: Syntax highlighting, code suggestions, and auto-completion help improve coding efficiency and accuracy.

2. Sidebar

- **Purpose**: Provides access to various features such as the file explorer, search, source control, and extensions.
- **Features**:
  - **Explorer**: Navigate and manage files and folders.
  - **Search**: Find and replace text across files.
  - **Source Control**: Integrate with version control systems like Git.
  - **Extensions**: Install and manage extensions to enhance functionality.

3. Status Bar

- **Purpose**: Displays important information about the current file and project, such as the file type, line number, and errors.
- **Features**: Shows language mode, git branch, and notifications.

4. Command Palette

- **Purpose**: Provides quick access to VS Code commands and settings.
- **Features**: Open with `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS) and search for commands.

5. Integrated Terminal

- **Purpose**: Allows you to run command-line tasks directly within VS Code.
- **Features**: Run scripts, manage version control, and execute other commands without leaving the editor.

6. Debugging Features

- **Purpose**: Helps in identifying and fixing issues in your code.
- **Features**: Set breakpoints, step through code, inspect variables, and view the call stack.

7. Extensions

- **Purpose**: Extend VS Code's capabilities to support additional programming languages, tools, and features.
- **Features**: Install extensions for language support (e.g., Python, C++), code formatting, linting, and more.

Application in Future Projects and Coursework

1. Efficient Coding

- **Use the Editor Window**: Write, edit, and format code efficiently with features like syntax highlighting and code suggestions.
- **Apply Code Formatting**: Keep your code clean and readable using automatic formatting.

2. Project Management

- **Utilize the Sidebar**: Organize and navigate project files easily. Use the Explorer to manage project structure and access files quickly.
- **Version Control**: Leverage the Source Control feature to manage and track changes in your codebase using Git.

3. Command Execution

- **Leverage the Integrated Terminal**: Run commands, execute scripts, and manage dependencies directly from the terminal.

4. Debugging

- **Implement Debugging Tools**: Set breakpoints, step through code, and inspect variables to troubleshoot and resolve issues in your code.

5. Customization and Extensions

- **Enhance with Extensions**: Install extensions to support different programming languages, tools, and frameworks relevant to your coursework and projects.
- **Adjust Settings**: Customize VS Code to fit your workflow and preferences.

6. Practice and Application

- **Create and Test Code**: Use VS Code to develop, test, and debug code for assignments and projects.
- **Organize Projects**: Create workspaces to manage and maintain project files and settings effectively.

Summary

- **Main Features**: VS Code’s editor window, sidebar, status bar, command palette, integrated terminal, debugging features, and extensions.
- **Application**: These tools will help streamline coding, manage projects, execute commands, debug code, and customize your development environment.

b. **Feedback:** - Collect feedback to improve future laboratory sessions.