**Download and Install Visual Studio Code.** [**https://code.visualstudio.com/Download**](https://code.visualstudio.com/Download)

**To me I have already installed the visual studio code, and I have followed the following steps:**

### For Windows:

**Download the Installer:**

* 1. Visit the [official VS Code website](https://code.visualstudio.com/).
  2. Click on the download button for Windows.

**Run the Installer:**

* 1. Once the download is complete, run the installer (VSCodeUserSetup-{version}.exe).

**Setup Wizard:**

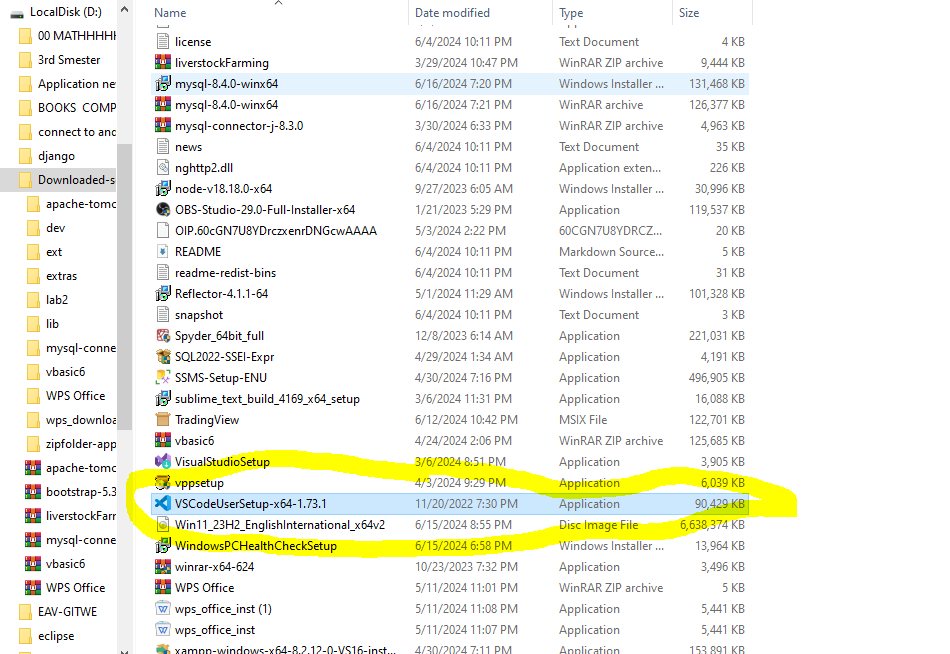
* 1. Follow the prompts in the setup wizard.
  2. Accept the license agreement.
  3. Choose the destination folder.
  4. Select additional tasks (like creating a desktop icon, adding to PATH, and registering code as an editor for supported file types).

**Install:**

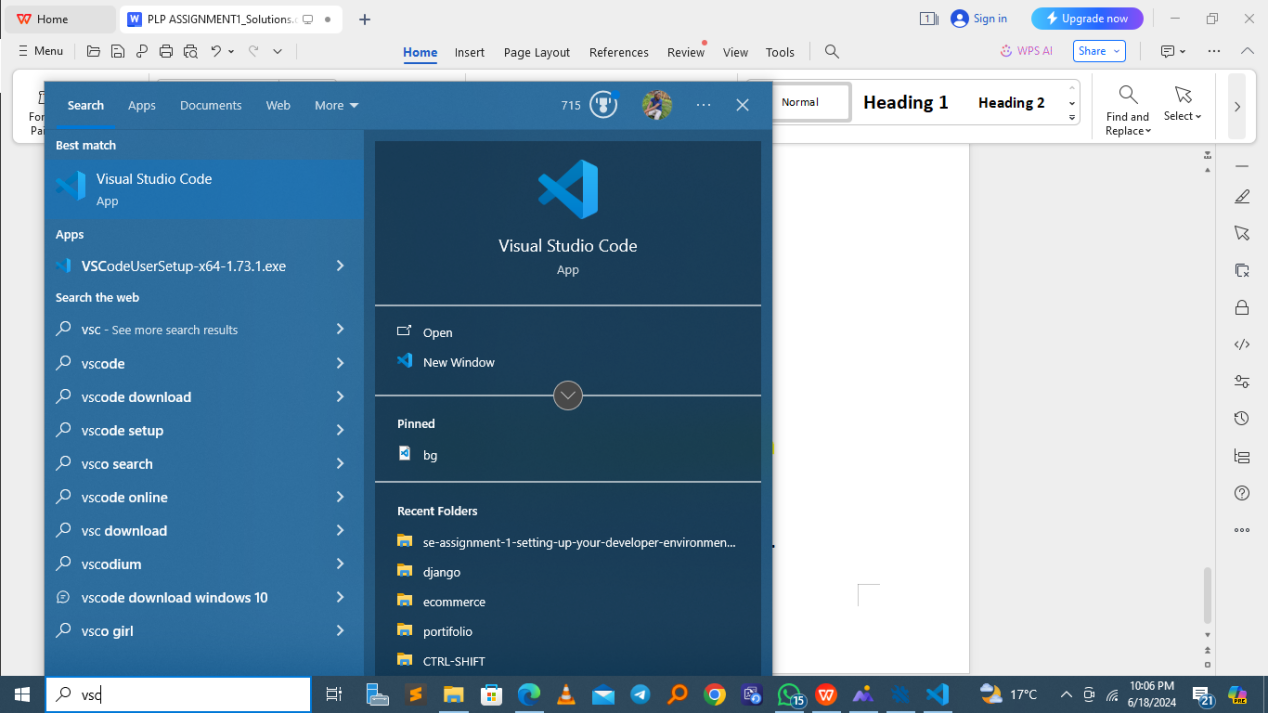
* 1. Click on the Install button.
  2. Once the installation is complete, you can launch VS Code.

**Therefore from my pc,**

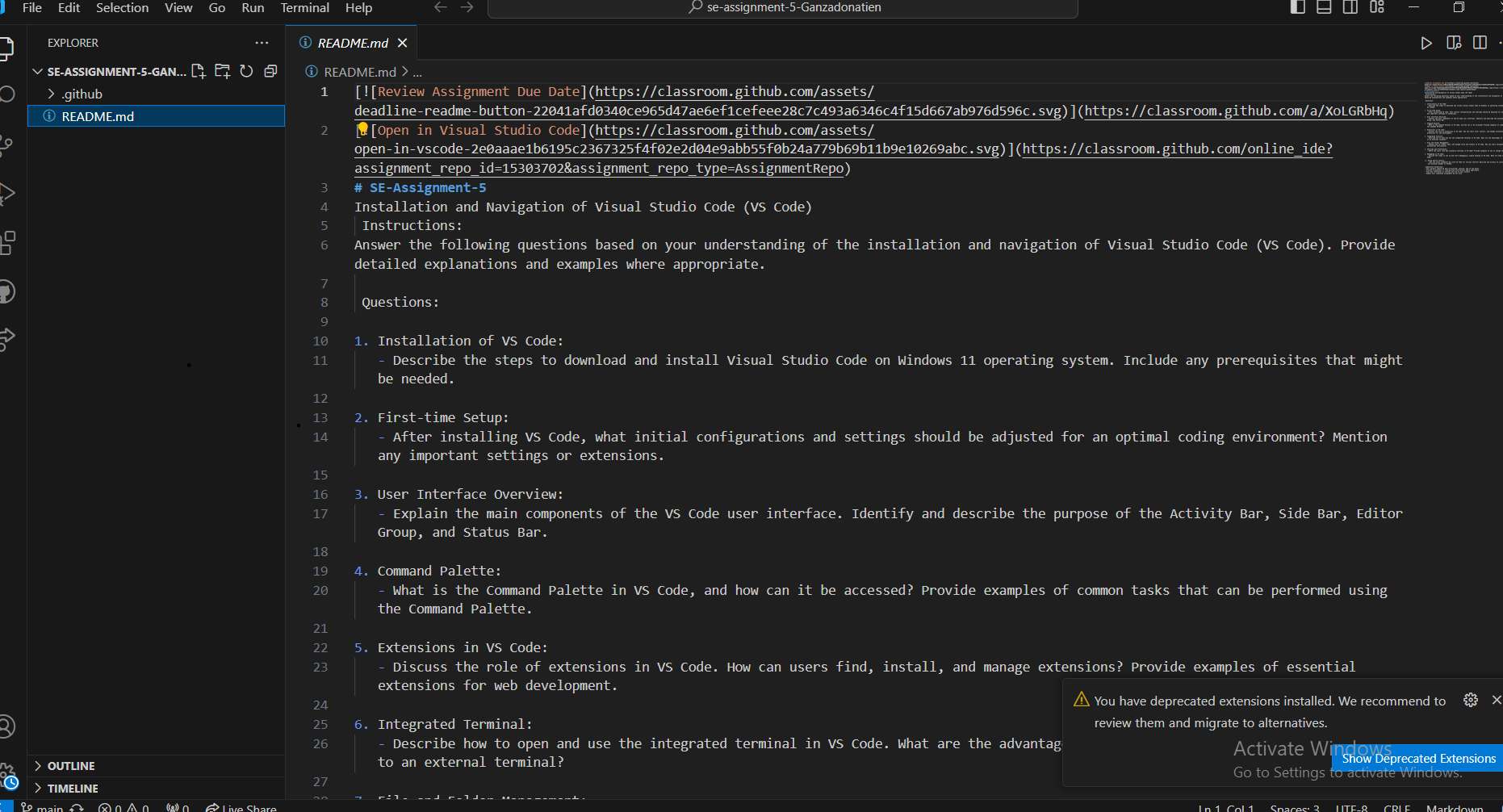
**-Downloaded vscode is:**



**After installation I can search it from the search bar.**

****

**The I opened it and I am answering the plp assignments using the vscode.**



The Command Palette in Visual Studio Code (VS Code) is a powerful feature that allows users to access and execute various commands and actions within the editor. It serves as a central hub for accessing functionality without needing to memorize specific keyboard shortcuts or navigate through menus. Here’s how you can access and use the Command Palette:

### Accessing the Command Palette:

To open the Command Palette in VS Code, you can use the following methods:

* **Keyboard Shortcut**: Press Ctrl + Shift + P on Windows/Linux or Cmd + Shift + P on macOS.
* **Menu Option**: Click on View in the top menu, then select Command Palette....
* **Right-click Menu**: Right-click anywhere in the editor or file explorer area and select Command Palette....

### Examples of Common Tasks Using the Command Palette:

1. **Opening Files**: You can quickly open a file by typing "Open File" in the Command Palette and then entering the file name.
2. **Switching Between Tabs**: Use "View: Show All Editors" to see all open tabs and switch between them directly.
3. **Searching for Symbols**: Typing "Go to Symbol in File" allows you to quickly navigate to a specific function or variable within the current file.
4. **Running Tasks**: VS Code tasks (configured in tasks.json or via extensions) can be executed using commands like "Run Task".
5. **Version Control Actions**: Commands like "Git: Pull", "Git: Push", "Git: Commit", etc., allow you to perform version control operations directly.
6. **Extensions Management**: Install, uninstall, enable, or disable extensions via commands like "Extensions: Install Extensions".
7. **Settings Management**: Quickly access and modify VS Code settings using commands such as "Preferences: Open Settings".
8. **Debugging**: Commands like "Debug: Start Debugging" or "Debug: Stop Debugging" help manage your debugging sessions.
9. **Workspace Management**: Commands like "Add Folder to Workspace" or "Save Workspace As..." help manage your workspace configuration.
10. **Customization and Configuration**: You can customize VS Code's behavior and appearance using commands like "Color Theme", "Change Language Mode", etc.

Extensions in Visual Studio Code (VS Code) play a crucial role in extending its functionality beyond basic text editing. They allow users to customize their development environment, integrate with external tools, and enhance productivity by providing features tailored to specific programming languages, frameworks, or workflows.

### Finding and Installing Extensions:

1. **Accessing the Marketplace**:
   * Extensions can be found and installed from the VS Code Marketplace (accessible via the Extensions view in VS Code or online at [marketplace.visualstudio.com](https://marketplace.visualstudio.com/)).
2. **Searching for Extensions**:
   * Users can search for extensions by name, category (e.g., Programming Languages, Snippets, Themes), or functionality (e.g., Git integration, debugging tools).
3. **Installing Extensions**:
   * Once a desired extension is found, click on "Install" to add it to VS Code. Some extensions may require additional permissions or dependencies.

### Managing Extensions:

1. **Viewing Installed Extensions**:
   * The Extensions view in VS Code lists all installed extensions and allows users to enable, disable, uninstall, or update them as needed.
2. **Configuring Extensions**:
   * Each extension may provide its own settings and configuration options, accessible through the VS Code settings (Ctrl + ,) or via the extension's details in the Extensions view.
3. **Updating Extensions**:
   * VS Code notifies users when updates are available for installed extensions. Updates can be applied manually or automatically based on user preferences.

### Examples of Essential Extensions for Web Development:

1. **ESLint**: Provides real-time linting for JavaScript and TypeScript to ensure code quality and adherence to coding standards.
2. **Prettier - Code formatter**: Automatically formats code (HTML, CSS, JavaScript, TypeScript, etc.) to maintain consistent style across the project.
3. **Debugger for Chrome**: Allows debugging JavaScript code running in the Chrome browser directly from VS Code.
4. **Live Server**: Launches a local development server with live reload capability for web pages, enhancing the development workflow.
5. **HTML CSS Support**: Provides autocompletion, linting, and syntax highlighting for HTML and CSS files.
6. **Auto Rename Tag**: Automatically renames paired HTML/XML tags, ensuring consistency and reducing manual effort.
7. **GitLens - Git supercharged**: Enhances Git integration within VS Code, providing advanced features like blame annotations, code lens, and commit history exploration.
8. **Path Intellisense**: Autocompletes filenames and paths in your code, reducing errors and improving efficiency when referencing files.
9. **Bracket Pair Colorizer**: Colorizes matching brackets in your code for easier visual identification of code blocks.
10. **REST Client**: Allows sending HTTP requests and viewing responses directly from within VS Code, useful for testing APIs.

### Advantages of Using Extensions in VS Code:

* **Customization**: Extensions allow users to tailor VS Code to their specific needs and preferences, enhancing both productivity and comfort.
* **Enhanced Functionality**: They extend the core capabilities of VS Code, integrating additional tools and features directly into the editor.
* **Community Support**: Many extensions are open-source and benefit from active community contributions and updates, ensuring ongoing improvement and compatibility with new technologies.
* **Workflow Optimization**: By automating common tasks, providing advanced debugging tools, and improving code quality through linting and formatting, extensions streamline development workflows.

To open and use the integrated terminal in Visual Studio Code (VS Code), follow these steps:

1. **Open VS Code**: Launch Visual Studio Code on your computer.
2. **Open a Project**: Open the folder or workspace where your project files are located.
3. **Open Integrated Terminal**:
   * There are several ways to open the integrated terminal:
     + Use the keyboard shortcut: Ctrl + (backtick key) on Windows/Linux or Cmd + on macOS.
     + Alternatively, you can navigate through the menu: View -> Terminal.
     + You can also right-click anywhere in the editor or file explorer area and select Open in Terminal.
4. **Using the Integrated Terminal**:
   * Once the terminal is open, you can interact with it just like any other terminal.
   * You can run commands, navigate directories (cd, ls), install packages (npm, pip, etc.), and execute scripts directly from within VS Code.
5. **Customization**:
   * VS Code's integrated terminal supports customization through settings. You can configure various aspects such as the shell path, font size, and color scheme to match your preferences.

### Advantages of Using the Integrated Terminal:

1. **Convenience**: The integrated terminal allows you to stay within the same window as your code editor, eliminating the need to switch between different applications or windows.
2. **Context Awareness**: Since the terminal is integrated with VS Code, it automatically opens at the root of your project. This makes it easier to run project-specific commands without manually navigating to the project directory.
3. **Efficiency**: Tasks such as debugging, running scripts, or managing version control (e.g., Git commands) can be seamlessly integrated into your workflow. You can quickly execute commands related to your coding tasks without disrupting your focus.

Managing files and folders in Visual Studio Code (VS Code) is essential for organizing and editing your projects effectively. Here’s a comprehensive guide on how to create, open, and manage files and folders, along with tips for efficient navigation between different files and directories:

### Creating Files and Folders:

1. **Creating a New File**:
   * To create a new file, you can use several methods:
     + Use the keyboard shortcut Ctrl + N (Windows/Linux) or Cmd + N (macOS) to create a new file in the current workspace.
     + Right-click in the Explorer sidebar or in the editor area and choose New File.
     + Use the Command Palette (Ctrl + Shift + P or Cmd + Shift + P) and type "New File" to create a new file.
2. **Creating a New Folder**:
   * Similarly, create a new folder using:
     + Ctrl + Shift + N (Windows/Linux) or Cmd + Shift + N (macOS) keyboard shortcut.
     + Right-click in the Explorer sidebar or in the editor area and select New Folder.
     + Use the Command Palette (Ctrl + Shift + P or Cmd + Shift + P) and type "New Folder".

### Opening Files and Folders:

1. **Opening Files**:
   * Double-click on a file in the Explorer sidebar to open it.
   * Use the File -> Open File... menu option or the keyboard shortcut Ctrl + O (Windows/Linux) or Cmd + O (macOS) to open a file.
2. **Opening Folders**:
   * Use File -> Open Folder... or Ctrl + K Ctrl + O (Windows/Linux) or Cmd + K Cmd + O (macOS) to open an entire folder as a workspace in VS Code.

### Managing Files and Folders:

1. **Renaming Files and Folders**:
   * Right-click on a file or folder in the Explorer sidebar and select Rename, or press F2 to rename it directly.
2. **Moving Files and Folders**:
   * Drag and drop files or folders within the Explorer sidebar to move them to a different location within your project directory.
3. **Deleting Files and Folders**:
   * Right-click on a file or folder in the Explorer sidebar and select Delete, or press Delete key (Windows/Linux) or Cmd + Delete (macOS) after selecting it.

### Navigating Between Files and Directories Efficiently:

1. **Using the Explorer Sidebar**:
   * The Explorer sidebar (accessible via View -> Explorer or Ctrl + Shift + E) displays a tree view of your project's files and folders. You can navigate by clicking on items in the sidebar.
2. **Switching Between Open Files**:
   * Use Ctrl + Tab (Windows/Linux) or Cmd + Tab (macOS) to switch between recently opened files.
   * Use Ctrl + P (Windows/Linux) or Cmd + P (macOS) to open the Quick Open dialog, where you can start typing the file name to quickly navigate to it.
3. **Using Tabs**:
   * Each open file in VS Code is represented as a tab at the top of the editor. Click on a tab to switch between open files.
4. **Navigating Directories in the Integrated Terminal**:
   * Use the integrated terminal (Ctrl + or Cmd + ) to navigate directories using command-line commands (cd, ls, etc.).
5. **Using Keyboard Shortcuts and Commands**:
   * Familiarize yourself with keyboard shortcuts for common tasks like opening files (Ctrl + O), opening the Command Palette (Ctrl + Shift + P), and navigating between tabs (Ctrl + Tab).

### Tips for Efficient File and Folder Management:

* **Use Workspace**: Organize related files and folders into VS Code workspaces for better project management.
* **Search**: Utilize VS Code's powerful search functionality (Ctrl + Shift + F for global search, Ctrl + F for search within a file) to quickly locate files and content within files.
* **Extensions**: Install extensions like "File Utils" or "Path Intellisense" to enhance file and folder management capabilities.

1. **Direct Interaction**: You can easily interact with files and folders in your project using terminal commands alongside your coding activities. This is particularly useful for tasks that involve file manipulation or batch operations.
2. **Output Handling**: Output from terminal commands appears directly within VS Code. This means you can easily review command output, errors, or warnings while working on your code.
3. **Integration with VS Code Features**: Some VS Code extensions and features integrate tightly with the integrated terminal, providing enhanced functionality such as running tasks, debugging configurations, and more.

n Visual Studio Code (VS Code), users can find and customize their settings through the Settings UI or by editing the settings.json file directly. Here’s a step-by-step guide on how to change the theme, font size, and keybindings in VS Code:

### Accessing the Settings

#### Settings UI

1. **Open Settings**:
   * Click on the gear icon in the bottom left corner and select "Settings."
   * Alternatively, you can open the Command Palette (Ctrl+Shift+P or Cmd+Shift+P on macOS) and type "Preferences: Open Settings (UI)" and select it.

#### settings.json File

1. **Open settings.json**:
   * Click on the gear icon in the bottom left corner, then click on "Settings."
   * In the top right corner of the Settings UI, click on the icon with the file and pencil to open the settings.json file directly.
   * Or, open the Command Palette and type "Preferences: Open Settings (JSON)" and select it.

### Changing the Theme

1. **Using the Settings UI**:
   * Open the Command Palette (Ctrl+Shift+P or Cmd+Shift+P on macOS).
   * Type and select "Preferences: Color Theme."
   * Choose a theme from the list.
2. **Using settings.json**:
   * Add or modify the following line in the settings.json file:

json

"workbench.colorTheme": "Theme Name"

* + Replace "Theme Name" with the name of the theme you want to use, for example, "Dark+ (default dark)".

### Changing the Font Size

1. **Using the Settings UI**:
   * Open the Settings UI.
   * Search for Font Size in the search bar.
   * Adjust the "Editor: Font Size" to your preferred size.
2. **Using settings.json**:
   * Add or modify the following line in the settings.json file:

json

"editor.fontSize": 14

* + Replace 14 with your desired font size.

### Changing Keybindings

1. **Using the Keybindings UI**:
   * Open the Command Palette (Ctrl+Shift+P or Cmd+Shift+P on macOS).
   * Type and select "Preferences: Open Keyboard Shortcuts."
   * Use the search bar to find the command you want to rebind.
   * Click on the pencil icon next to the command and press the new key combination you want to use.
2. **Using keybindings.json**:
   * Open the Command Palette and type "Preferences: Open Keyboard Shortcuts (JSON)" and select it.
   * Add or modify an entry in the keybindings.json file. Here’s an example of how to change the keybinding for saving a file:

json

{

"key": "ctrl+s",

"command": "workbench.action.files.save"

}

* + Replace "ctrl+s" with your desired key combination and "workbench.action.files.save" with the command you want to bind it to.

### Debugging in VS Code

#### Setting Up and Starting Debugging

1. **Open Your Project**:
   * Open the folder containing your project files in VS Code.
2. **Open the Debug View**:
   * Click on the Run icon in the Activity Bar on the side of the window (or press Ctrl+Shift+D / Cmd+Shift+D on macOS).
3. **Create a Debug Configuration**:
   * Click on the gear icon at the top of the Debug panel and select "Add Configuration..."
   * Choose the appropriate environment for your project (e.g., Node.js, Python, C++, etc.).
   * This will create a launch.json file in the .vscode folder with default configurations.
4. **Set Breakpoints**:
   * Open the file you want to debug.
   * Click in the gutter next to the line number where you want to set a breakpoint. A red dot will appear indicating the breakpoint.
5. **Start Debugging**:
   * In the Debug panel, select the configuration you want to use from the dropdown menu.
   * Click the green play button to start debugging.

#### Key Debugging Features

* **Breakpoints**: Pause the execution of your code at specific lines.
* **Step Over, Step Into, Step Out**: Navigate through your code line by line to understand the flow and state.
* **Watch Expressions**: Monitor variables and expressions to see how their values change during execution.
* **Call Stack**: View the call stack to see the sequence of function calls that led to the current point of execution.
* **Variables**: Inspect the current state of variables in scope.
* **Debug Console**: Evaluate expressions and interact with the program's runtime.

### Using Source Control

#### Integrating Git with VS Code

1. **Initialize a Repository**:
   * Open the folder containing your project in VS Code.
   * Click on the Source Control icon in the Activity Bar on the side of the window (or press Ctrl+Shift+G / Cmd+Shift+G on macOS).
   * Click on "Initialize Repository" if you haven't already set up a Git repository in the project folder.
2. **Making Commits**:
   * **Stage Changes**: In the Source Control panel, you will see a list of changes. Click the "+" icon next to each file to stage changes, or click on the "+" icon at the top to stage all changes.
   * **Commit Changes**: Enter a commit message in the text box at the top and click the checkmark icon to commit the changes.
3. **Pushing Changes to GitHub**:
   * **Link to a Remote Repository**: If you haven’t already, link your local repository to a remote repository on GitHub.
     + Open the Command Palette (Ctrl+Shift+P / Cmd+Shift+P) and type "Git: Add Remote."
     + Enter the URL of your GitHub repository.
   * **Push Changes**: Click on the "..." icon in the Source Control panel and select "Push" to push your commits to the remote repository.
     + If prompted, enter your GitHub credentials or use an authentication token.

#### Additional Git Features in VS Code

* **Branches**: Create, switch, and manage branches directly from VS Code.
* **Merge Conflicts**: Visualize and resolve merge conflicts with an intuitive interface.
* **History and Diff**: View commit history and diffs for changes, allowing you to understand modifications over time.
* **Pull Requests**: Integrate with GitHub to create and manage pull requests within VS Code.