Laurent kiiru

SE-Assignment-5

PLP Academy

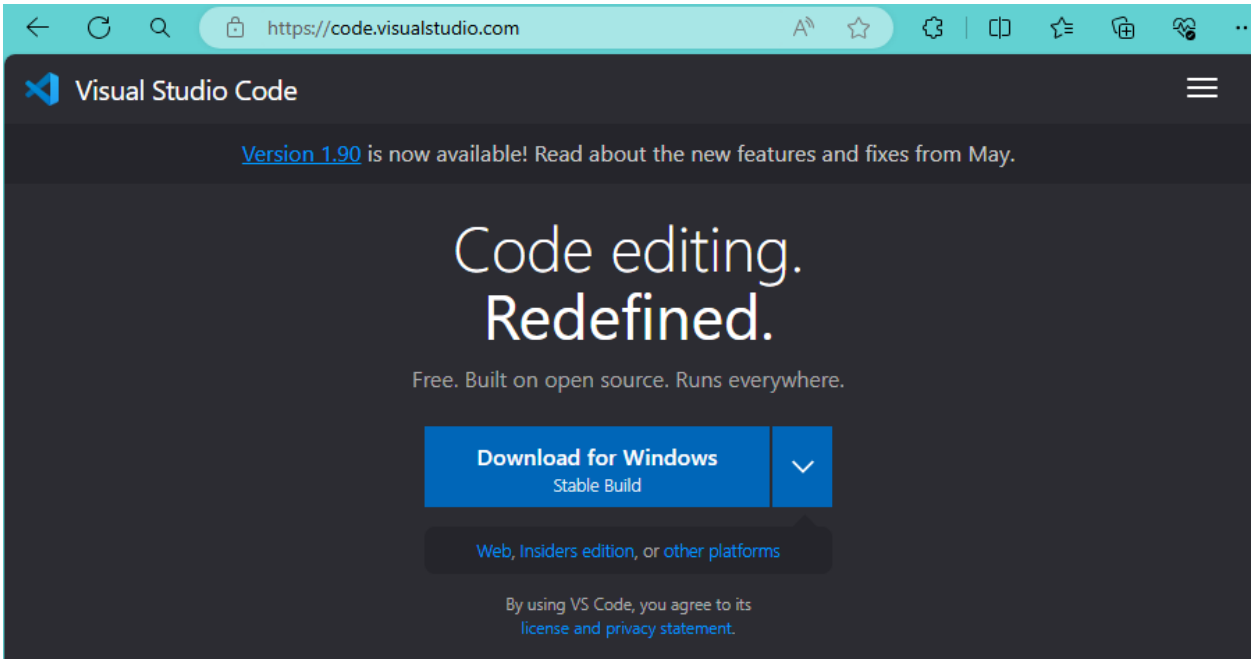
Installation and navigation of visual studio.

1. Installation of VS Code:
    - Describe the steps to download and install Visual Studio Code on Windows 11 operating system. Include any prerequisites that might be needed.

- Ensure you have windows 11 installed on your computer.
- Ensure you have admin privileges to install vs code software.

Steps

- Open a web browser.
- Navigate to the visual studio website https://code.visualstudio.com/.



- Click on the download button so that you install the downloader that is if you are using windows, i used this one since in using windows.
- **Run the Installer -** Once the download is complete, locate the `VSCodeSetup.exe` file in your Downloads folder or the location where your browser saves downloaded files. Double-click the `VSCodeSetup.exe` file to start the installation process.

- If prompted by the User Account Control (UAC), click "Yes" to allow the installer to make changes to your device.
- The Visual Studio Code Setup Wizard will open. Click "Next" to proceed through the setup process.
- Read through the license agreement. If you accept the terms, click the "I accept the agreement" checkbox, and then click "Next".
- Choose the destination folder where you want Visual Studio Code to be installed. The default location is usually fine. Click "Next".
    - You can choose additional tasks to be performed during the installation. Common options include:
    - Create a desktop icon.
    - Add "Open with Code" actions to the context menu (right-click menu) for folders and files.
    - Register Visual Studio Code as an editor for supported file types.
    - Select the options you prefer and click "Next".
- Review your settings and click "Install" to begin the installation.
- The setup wizard will install Visual Studio Code. This may take a few minutes, and installation will be complete.
- You can now open vs code from the start menu and choose your configurations such as installing extensions.

2. First-time Setup:

- After installing VS Code, what initial configurations and settings should be adjusted for an optimal coding environment? Mention any important settings or extensions

Important Extensions

- **Language Support**:
    - **Python**: Provides rich support for Python, including features like IntelliSense, linting, and debugging.
    - **JavaScript and TypeScript**: Comes pre-installed with VS Code.
    - **C/C++**: Essential for C and C++ development, providing IntelliSense, debugging, and code browsing.
- **Code Quality and Formatting**:
    - **ESLint**: Integrates ESLint into VS Code to find and fix problems in JavaScript code.

    - **Prettier**: An opinionated code formatter that supports many languages.
- **Productivity Tools**:
    - **Live Server**: Launch a local development server with a live reload feature for static and dynamic pages.
    - **Path Intellisense**: Autocompletes filenames.
    - **Bracket Pair Colorizer**: Adds color to matching brackets to help with readability.
- **Version Control**:
    - **GitLens**: Enhances the built-in Git capabilities. It helps visualize code authorship and timeline with annotations and blame information.
- **Debugging and Testing**:
    - **Debugger for Chrome**: Debug your JavaScript code in the Google Chrome browser or any other target that supports the Chrome Debugger protocol.
    - **Jest**: Provides rich features for Jest testing in JavaScript and TypeScript projects.
- **Docker and DevOps**:
    - **Docker**: Provides a set of features to integrate Docker into VS Code, enabling container management.
    - **Azure Account**: If you use Azure, this extension helps you manage your Azure subscriptions and resources.

Custom Key bindings

Customize keybindings to improve efficiency:

- Go to `File > Preferences > Keyboard Shortcuts`.
- Modify shortcuts as per your workflow requirements. For instance, you can set a shortcut for opening the terminal or formatting the code.

Snippets

Create or install code snippets to streamline repetitive coding tasks:

- Go to `File > Preferences > User Snippets`.
- Create a new snippet file for your language or install pre-existing snippet extensions from the marketplace.

Terminal Configuration

Set up the integrated terminal to use your preferred shell:

- Go to `File > Preferences > Settings` and search for "Terminal Integrated Shell".
- Set the path for the shell you want to use (e.g., Git Bash, PowerShell).

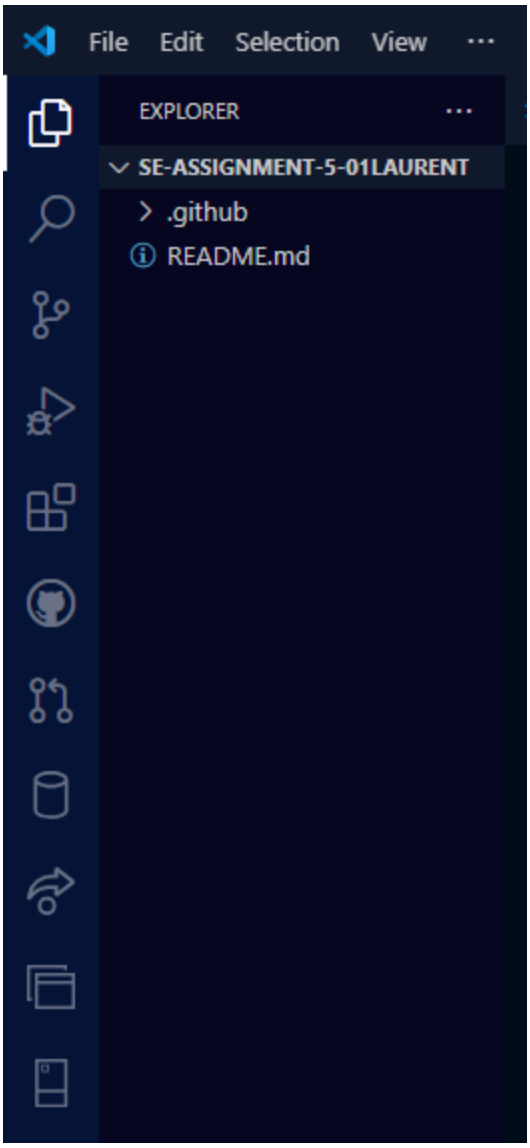
Workspace Configuration

Configure workspace-specific settings to tailor VS Code for different projects:

- Open a project folder and navigate to `File > Preferences > Settings (Workspace)` to adjust settings specific to that project.

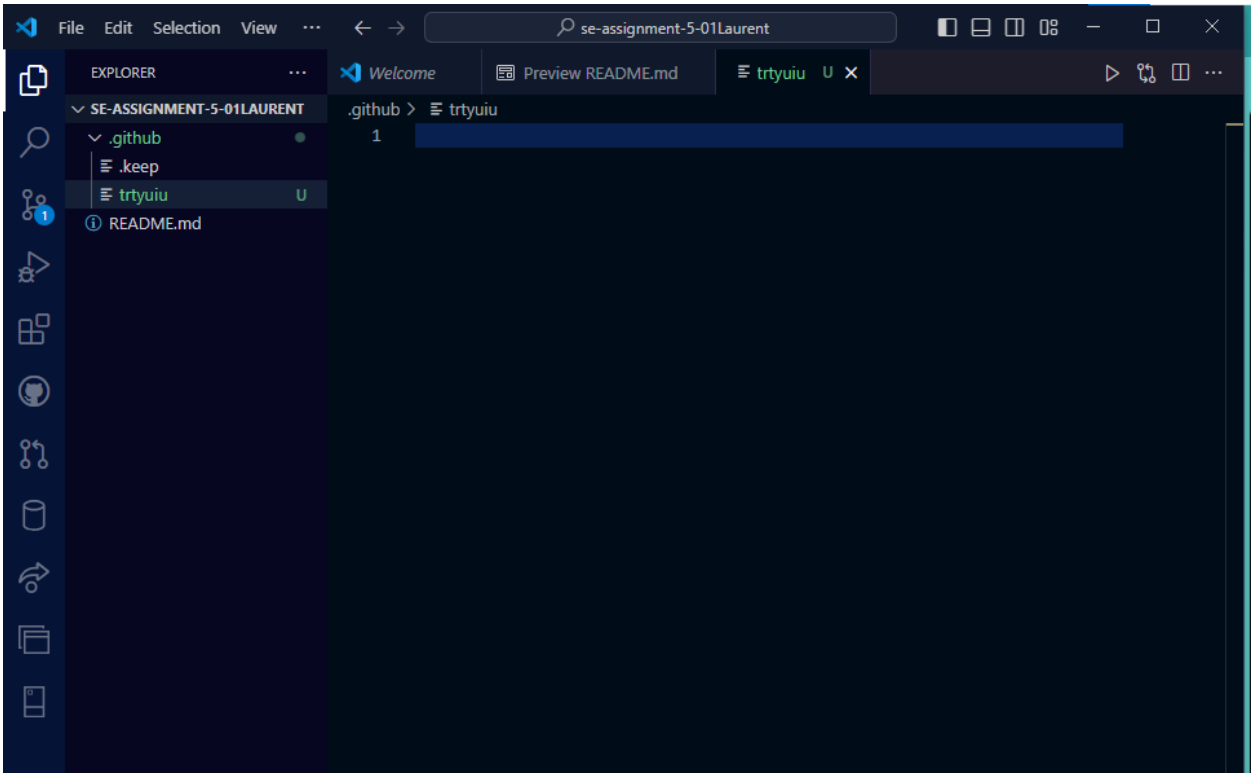
3. User Interface Overview:
    - Explain the main components of the VS Code user interface. Identify and describe the purpose of the Activity Bar, Side Bar, Editor Group, and Status Bar.

- Side bar



- **Purpose**: The Side Bar displays the contents related to the view selected in the Activity Bar. For instance:
- **Explorer View**: Shows the file and folder structure of your workspace.
- **Search View**: Displays search results within your project files.
- **Source Control View**: Shows the status of your version-controlled files.
- **Run and Debug View**: Lists your debugging configurations, breakpoints, and variables.
- **Extensions View**: Lists installed extensions and allows you to browse and install new ones.

- Editor Group.



Purpose: The Editor Group is where you open and edit files. It supports:

- **Multiple Tabs**: Open multiple files in tabs within a single group.
- **Multiple Groups**: Split the editor into multiple groups (columns or rows) to view and edit files side-by-side.
- **Diff View**: Compare two files side-by-side to see differences (commonly used in version control).

- Status Bar

Purpose: The Status Bar provides information about the current state of the editor and workspace. Key features include:

- **Language Mode**: Indicates the programming language of the currently open file. Clicking it allows you to change the language mode.
- **Encoding**: Shows the file's character encoding (e.g., UTF-8).
- **EOL (End of Line)**: Displays the type of line ending used in the file (e.g., LF, CRLF).

- **Indentation**: Shows the current indentation settings (spaces or tabs, and the size).
- **Git Branch**: Displays the active Git branch and any changes to be committed.
- **Problems**: Shows the number of errors and warnings in the code.
- **Notifications**: Displays various notifications and messages from extensions and the system.

4. Command Palette:
    - What is the Command Palette in VS Code, and how can it be accessed? Provide examples of common tasks that can be performed using the Command Palette.

Command Palette in Visual Studio Code (VS Code) is a powerful feature that provides quick access to a wide range of commands and functionalities. It is essentially a searchable interface that allows you to execute commands without having to navigate through menus or remember keyboard shortcuts.

How to access the command palette.

- **Keyboard Shortcut**: Press `Ctrl+Shift+P` on Windows or `Cmd+Shift+P` on macOS.
- **Menu**: Go to `View > Command Palette` in the menu bar.

Common Tasks Performed Using the Command Palette

- **File Operations**:
- **Open File**: Type `> Open File` to quickly open a file by name.
- **Save All**: Type `> Save All` to save all open files.
- **Close Editor**: Type `> Close Editor` to close the current file.
- **View and Navigation**:
- **Toggle Sidebar Visibility**: Type `> View: Toggle Sidebar Visibility` to show or hide the sidebar.
- **Go to Symbol in File**: Type `@` to list and navigate to symbols (e.g., functions, variables) within the current file.
- **Go to Line**: Type `:` followed by a line number to navigate directly to a specific line in the file.
- **Editor Configuration**:

- **Change Language Mode**: Type `> Change Language Mode` to switch the syntax highlighting and other features to a different programming language.
- **Toggle Word Wrap**: Type `> Toggle Word Wrap` to enable or disable word wrapping in the editor.
    - **Extensions**:
- **Install Extensions**: Type `> Extensions: Install Extensions` to open the extensions view and install new extensions.
- **Disable Extension**: Type `> Extensions: Disable` to disable a specific extension.
    - **Version Control**:
- **Git: Clone**: Type `> Git: Clone` to clone a repository from a remote source.
- **Git: Commit**: Type `> Git: Commit` to commit changes with a message.
    - **Debugging**:
- **Start Debugging**: Type `> Debug: Start Debugging` to start a debugging session.
- **Add Configuration**: Type `> Debug: Add Configuration` to add a new debugging configuration.
    - **Terminal**:
- **Create New Integrated Terminal**: Type `> Terminal: Create New Integrated Terminal` to open a new terminal instance.
- **Run Task**: Type `> Tasks: Run Task` to run a pre-configured task (e.g., build, test).
    - **Custom Commands**:
- **Run Shell Command**: Type `> Tasks: Run Task` to execute a shell command configured in your tasks.json file.
- **Open Settings**: Type `> Preferences: Open Settings` to access the settings editor

5. Extensions in VS Code:
    - Discuss the role of extensions in VS Code. How can users find, install, and manage extensions? Provide examples of essential extensions for web development.

- Finding Extensions
    - Click on the Extensions icon in the Activity Bar on the left side of the window.
    - Alternatively, use the Command Palette (`Ctrl+Shift+P` on Windows or `Cmd+Shift+P` on macOS) and type `Extensions: Install Extensions`.
    - **Search -** Use the search bar within the Extensions view to find specific extensions by name or functionality.

- Installing Extensions
  - Search for the desired extension in the Extensions view.
  - Click the `Install` button next to the extension you want to add.
  - Some extensions may require a reload of VS Code to activate.
- **Command Palette.**
  - Open the Command Palette (`Ctrl+Shift+P` or `Cmd+Shift+P`).
  - Type `Extensions: Install Extensions` and press Enter.
  - Search for the extension and click `Install`.

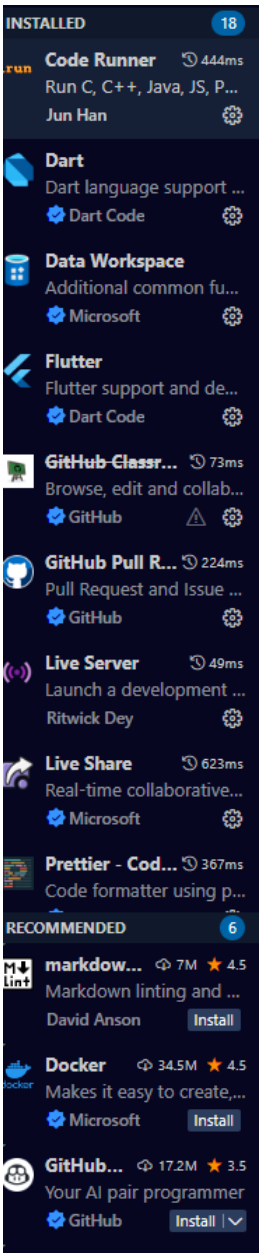
Managing Extensions

- Enable/Disable - In the Extensions view, you can enable or disable extensions by clicking the gear icon next to the extension and selecting `Disable` or `Enable`.
- Update - Extensions will automatically check for updates. When an update is available, an update button will appear next to the extension in the Extensions view.
- Uninstall - To uninstall an extension, click the gear icon next to the extension and select `Uninstall`.
- **Settings -** Many extensions have configurable settings. Access these settings by clicking the gear icon and selecting `Extension Settings`.

Examples of Essential Extensions for Web Development.

- **Language Support**:
  - **ESLint**: Provides JavaScript and TypeScript linting using ESLint.
  - **Prettier - Code Formatter**: An opinionated code formatter that supports multiple languages.
- **Code Productivity**:
  - **Live Server**: Launches a local development server with live reload capability for static and dynamic pages.

    - **Path IntelliSense**: Autocompletes filenames for faster development.
- **Version Control**:
    - **GitLens**: Enhances Git capabilities within VS Code, providing insights into code authorship and history.
- **Debugging and Testing**:
    - **Debugger for Chrome**: Enables debugging JavaScript code running in the Google Chrome browser.
    - **Jest**: Integrates Jest testing framework into VS Code for running and debugging tests.
- **HTML and CSS**:
    - **HTML CSS Support**: Enhances HTML and CSS language support.
    - **Auto Rename Tag**: Automatically renames paired HTML/XML tags.
- **Snippet Tools**:
    - **JavaScript (ES6) code snippets**: Provides a collection of useful JavaScript snippets.
    - **CSS Peek**: Allows peeking to CSS definitions directly from HTML files.
- **Frameworks and Libraries**:
    - **Vue.js Extension Pack**: Provides support for Vue.js development.
    - **React Extension Pack**: A set of extensions for React development, including snippets and debugging tools.
- **Containerization and DevOps**:
    - **Docker**: Adds support for Docker, including syntax highlighting, commands, and integration with Docker Hub.



6. Integrated Terminal:
    - Describe how to open and use the integrated terminal in VS Code. What are the advantages of using the integrated terminal compared to an external terminal?

Opening the Integrated Terminal

Using the Menu:

- Go to `View > Terminal` from the menu bar.

Using Keyboard Shortcuts:

- Press `Ctrl+` (backtick) on Windows or `Cmd+` on macOS.

Command Palette:

- Open the Command Palette using `Ctrl+Shift+P` (Windows) or `Cmd+Shift+P` (macOS).
- Type `Toggle Terminal` and select it.

Using the Integrated Terminal

Basic Operations:

- **Create New Terminal**: Click the + icon in the terminal panel or use the `Ctrl+Shift+` (Windows) or `Cmd+Shift+` (macOS) shortcut.
- **Switch Between Terminals**: If you have multiple terminals open, you can switch between them using the dropdown menu at the top of the terminal panel.
- **Split Terminal**: Click the split terminal icon to divide the terminal into multiple sections within the same panel.
- **Close Terminal**: Click the trash can icon or use the `Ctrl+W` (Windows) or `Cmd+W` (macOS) shortcut to close the active terminal.

Customizing the Terminal:

- **Change Shell**: To change the default shell (e.g., from PowerShell to Git Bash or Command Prompt), go to `File > Preferences > Settings` and search for `Terminal Integrated Shell`.
- **Resize**: You can resize the terminal by dragging the top edge of the terminal panel.

Running Commands:

- The integrated terminal allows you to run any command that you would normally run in an external terminal. This includes navigating the file system, running scripts, executing build commands, and using version control tools like Git.

Advantages of Using the Integrated Terminal

Convenience and Context:

- **Single Window**: By using the integrated terminal, you can avoid switching between different applications, keeping all your coding and terminal activities within a single window. This reduces context switching and streamlines your workflow.
- **Context Awareness**: The integrated terminal opens in the context of your current workspace or project directory, eliminating the need to navigate to the project directory manually every time you open a new terminal.

Enhanced Integration:

- **Project Integration**: The integrated terminal is aware of the VS Code workspace and can directly interact with files and configurations specific to your project. For example, running `npm start` or `yarn build` directly within the project folder.
- **Editor Interaction**: You can copy and paste commands and outputs between the terminal and the editor easily. For instance, copying error messages from the terminal and searching for them within your code.

Customization and Control:

- **Shell Choice**: You can customize which shell you want to use (e.g., Bash, PowerShell, Command Prompt, Zsh) directly within VS Code.
- **Split and Multiple Terminals**: The ability to split the terminal into multiple panels allows for running and monitoring several processes simultaneously within the same interface.

Productivity Features:

- **Keyboard Shortcuts**: Integrated keyboard shortcuts make it faster to open, close, and switch between terminals.
- **Theme Consistency**: The terminal inherits the same color scheme and font settings as your VS Code theme, providing a consistent visual experience.

Environment Management:

- **Terminal Profiles**: You can define terminal profiles to quickly open terminals with specific configurations, such as different environments or shell configurations.

- **Task Integration**: Integrating tasks with the terminal allows you to automate common workflows, such as running tests or building projects with a single command.

7. File and Folder Management:
    - Explain how to create, open, and manage files and folders in VS Code. How can users navigate between different files and directories efficiently?

Using the Explorer View:

- **Create a File**:
    - Open the Explorer view by clicking the file icon in the Activity Bar or using the keyboard shortcut `Ctrl+Shift+E`.
    - Right-click in the Explorer pane where you want to create the file and select `New File`.
    - Enter the name of the new file and press Enter.
- **Create a Folder**:
    - Right-click in the Explorer pane and select `New Folder`.
    - Enter the name of the new folder and press Enter.

Using the Command Palette:

- Press `Ctrl+Shift+P` (Windows) or `Cmd+Shift+P` (macOS) to open the Command Palette.
- Type `File: New File` or `File: New Folder` and press Enter.
- Enter the name of the new file or folder.

Using Keyboard Shortcuts:

- To create a new file, use the keyboard shortcut `Ctrl+N` (Windows) or `Cmd+N` (macOS).
- Save the new file by pressing `Ctrl+S` (Windows) or `Cmd+S` (macOS) and providing a filename.

Opening Files and Folders

Using the Explorer View:

- Click on any file in the Explorer pane to open it in the editor.
- Double-click a file to keep it open in a tab.

Using the Command Palette:

- Press `Ctrl+Shift+P` (Windows) or `Cmd+Shift+P` (macOS).
- Type `File: Open File` or `File: Open Folder` and press Enter.
- Navigate to and select the file or folder you want to open.

Using the Menu:

- Go to `File > Open File` or `File > Open Folder` from the menu bar.
- Browse and select the file or folder.

Using Keyboard Shortcuts:

- To quickly open a file, use `Ctrl+O` (Windows) or `Cmd+O` (macOS).
- To open a folder, use `Ctrl+K` followed by `Ctrl+O` (Windows) or `Cmd+K` followed by `Cmd+O` (macOS).

Managing Files and Folders

Rename:

- Right-click on the file or folder in the Explorer view and select `Rename`.
- Enter the new name and press Enter.

Move:

- Drag and drop the file or folder to the desired location within the Explorer pane.

Delete:

- Right-click on the file or folder and select `Delete`.
- Confirm the deletion when prompted.

Copy and Paste:

- Right-click on the file or folder and select `Copy`.
- Right-click in the destination folder and select `Paste`.

Navigating Between Files and Directories Efficiently

Quick Open:

- Press `Ctrl+P` (Windows) or `Cmd+P` (macOS) to open the Quick Open dialog.
- Start typing the name of the file you want to open, and select it from the list.

File Explorer:

- Use the Explorer view (shortcut `Ctrl+Shift+E` or `Cmd+Shift+E`) to navigate the directory structure and open files with a click.

Breadcrumbs:

- Enable Breadcrumbs by selecting `View > Show Breadcrumbs`.
- Use the Breadcrumbs at the top of the editor to navigate through the file path and switch between files and directories.

Go to Symbol:

- Press `Ctrl+Shift+O` (Windows) or `Cmd+Shift+O` (macOS) to open the "Go to Symbol" dialog.
- Type the name of the symbol (e.g., function, variable) to navigate directly to it within the file.

Go to Definition:

- Right-click on a symbol in your code and select `Go to Definition`, or press `F12`.

Navigate Back and Forward:

- Use `Alt+Left Arrow` and `Alt+Right Arrow` (Windows) or `Ctrl+Minus` and `Ctrl+Shift+Minus` (macOS) to navigate back and forward through your editing history.

Integrated Terminal Navigation:

- Open the integrated terminal with `Ctrl+` (backtick) or `Cmd+` (macOS).
- Use standard command-line navigation commands (e.g., `cd`, `ls`, `dir`) to navigate directories and manage files.

8. Settings and Preferences:
    - Where can users find and customize settings in VS Code? Provide examples of how to change the theme, font size, and keybindings.

Accessing and Customizing Settings

Accessing Settings

Using the Menu:

- Go to `File > Preferences > Settings` (Windows) or `Code > Preferences > Settings` (macOS).

Using the Command Palette:

- Press `Ctrl+Shift+P` (Windows) or `Cmd+Shift+P` (macOS) to open the Command Palette.
- Type `Preferences: Open Settings` and press Enter.

Using Keyboard Shortcuts:

- Press `Ctrl+,` (Windows) or `Cmd+,` (macOS).

Changing the Theme

Access Settings as described above.
In the Settings tab, search for `Color Theme`.
Alternatively, use the Command Palette:

- Press `Ctrl+Shift+P` (Windows) or `Cmd+Shift+P` (macOS).
- Type `Preferences: Color Theme` and press Enter.
- Select a theme from the list that appears.

Changing the Font Size

Access Settings as described above.
In the Settings tab, search for `Font Size`.

Adjust the `Editor: Font Size` setting by entering the desired font size.
Alternatively, you can directly modify the `settings.json` file:

- Click on the {} icon in the top right corner of the Settings tab to open the settings in JSON format.
- Add or modify the following line:

```
json
Copy code
"editor.fontSize": 16
```

Changing Key Bindings

Access Keyboard Shortcuts:

- Go to `File > Preferences > Keyboard Shortcuts` (Windows) or `Code > Preferences > Keyboard Shortcuts` (macOS).
- Alternatively, press `Ctrl+K Ctrl+S` (Windows) or `Cmd+K Cmd+S` (macOS).

Modify Key Bindings:

- In the Keyboard Shortcuts editor, you can search for commands and modify their keybindings.
- To change a keybinding, click on the command you want to modify and then press the desired key combination.
- Example: To change the keybinding for opening a new terminal:
  - Search for `Terminal: Create New Integrated Terminal`.
  - Click on the existing keybinding or the empty space next to it.
  - Press the desired key combination (e.g., `Ctrl+Alt+T`).

Keybindings JSON:

- For advanced users, you can edit the keybindings directly in JSON format by clicking the {} icon in the Keyboard Shortcuts editor.
- Example JSON entry to change the terminal keybinding:

```
json
[
  {
    "key": "ctrl+alt+t",
    "command": "workbench.action.terminal.new"
```

```
  }
]
```

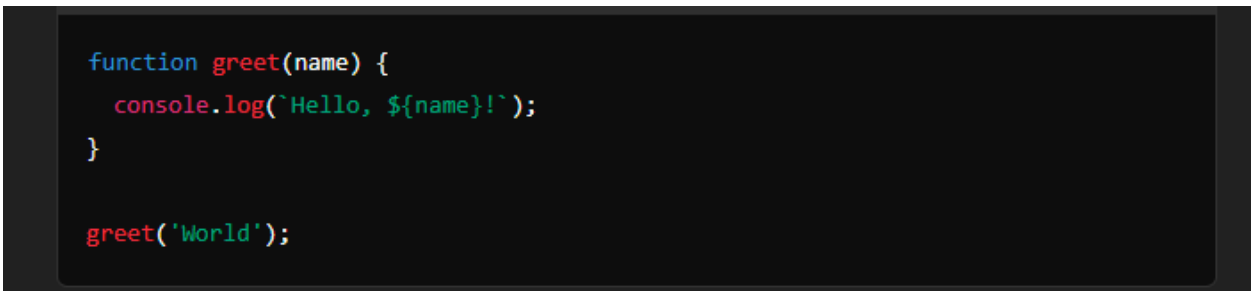
9. Debugging in VS Code:
    - Outline the steps to set up and start debugging a simple program in VS Code. What are some key debugging features available in VS Code?

Open Your Project:

- Launch VS Code and open your project folder by selecting `File > Open Folder` or using the shortcut `Ctrl+K Ctrl+O`.

Create a Simple Program:

- For example, create a new file named `app.js` and write a simple JavaScript program:



```
function greet(name) {
  console.log(`Hello, ${name}!`);
}

greet('World');
```

Configure the Debugger:

- Open the Debug view by clicking the debug icon in the Activity Bar or using the shortcut `Ctrl+Shift+D`.
- Click the `create a launch.json file` link in the Debug view. This will open a selection of environments. Choose `Node.js` (since we are using a JavaScript example).
- VS Code will create a `launch.json` file in a `.vscode` folder with default configurations:

`Json`

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Launch Program",
      "skipFiles": ["<node_internals>/**"],
      "program": "${workspaceFolder}/app.js"
    }
  ]
}
```

Set Breakpoints:

- Open `app.js` in the editor.
- Click in the gutter next to the line numbers where you want to set a breakpoint. For example, set a breakpoint on the `console.log` line:

```
javascript
console.log(`Hello, ${name}!`);
```

Start Debugging:

- In the Debug view, ensure `Launch Program` is selected in the dropdown at the top.
- Click the green play button or press `F5` to start debugging.

Key Debugging Features in VS Code

Breakpoints:

- **Set/Clear Breakpoints**: Click in the gutter next to the line number to toggle breakpoints.
- **Conditional Breakpoints**: Right-click on a breakpoint and select `Edit Breakpoint` to set conditions for stopping execution.

Watch Expressions:

- Add expressions to the Watch panel to evaluate and monitor variables or expressions during execution.
- Open the Watch panel in the Debug view, click the + icon, and enter the expression you want to watch.

Call Stack:

- View the call stack to see the function call sequence leading to the current point of execution.
- The Call Stack panel is located in the Debug view and shows all active stack frames.

Variable Inspection:

- Inspect local and global variables, as well as their values, in the Variables panel.
- Hover over variables in the editor to see their current values.

Step Controls:

- Use the controls in the Debug toolbar to navigate through your code:
  - **Continue (`F5`)**: Resume execution until the next breakpoint.
  - **Step Over (`F10`)**: Execute the next line of code, stepping over function calls.
  - **Step Into (`F11`)**: Step into functions to debug them.
  - **Step Out (`Shift+F11`)**: Step out of the current function.
  - **Restart (`Ctrl+Shift+F5`)**: Restart the debugging session.
  - **Stop (`Shift+F5`)**: Stop the debugging session.

Debug Console:

- Use the Debug Console to evaluate expressions and execute commands during a debugging session.
- Access the Debug Console from the Debug view.

Exception Handling:

- Configure VS Code to break on exceptions by modifying the `launch.json` file:

```
json
```

```
"configurations": [
  {
    "type": "node",
    "request": "launch",
    "name": "Launch Program",
    "skipFiles": ["<node_internals>/**"],
    "program": "${workspaceFolder}/app.js",
    "runtimeArgs": ["--nolazy"],
    "env": { "NODE_ENV": "development" },
    "outFiles": ["${workspaceFolder}/out/**/*.js"],
    "stopOnEntry": false,
    "console": "integratedTerminal",
    "internalConsoleOptions": "neverOpen"
  }
]
```

10. Using Source Control:
    - How can users integrate Git with VS Code for version control? Describe the process of initializing a repository, making commits, and pushing changes to GitHub.

Initializing a Repository

Open VS Code:

- Launch Visual Studio Code and open your project folder (`File > Open Folder`).

Open the Source Control View:

- Click on the Source Control icon in the Activity Bar on the left side of the window (or use the shortcut `Ctrl+Shift+G`).

Initialize Git Repository:

- Click on the `Initialize Repository` button (it looks like a + icon) in the Source Control view header.
- Alternatively, open the Command Palette (`Ctrl+Shift+P`), type `Git: Initialize Repository`, and press Enter.

Select Repository Location:

Choose the root folder of your project to initialize Git. VS Code will create a hidden `.git` folder in this directory to store Git-related metadata.

Making Commits

Stage Changes:

- In the Source Control view, you'll see a list of changed files. Click the + icon next to each file you want to stage for commit. Staged files appear with a + icon beside them.

Enter Commit Message:

- Below the list of staged files, enter a descriptive commit message in the text box that says "Message (press Ctrl+Enter to commit)".

Commit Changes:

- Press `Ctrl+Enter` (Windows) or `Cmd+Enter` (macOS) to commit the changes. Alternatively, you can click the checkmark icon (`√ Commit`) next to the message box.

Pushing Changes to GitHub

Linking to a GitHub Repository:

- If your project is not yet linked to a GitHub repository, create a new repository on GitHub and copy the repository URL.

Add Remote Repository:

- Open the Command Palette (`Ctrl+Shift+P`), type `Git: Add Remote`, and press Enter.
- Paste the GitHub repository URL and press Enter.

Push Commits to GitHub:

- After committing your changes locally:
  - Open the Command Palette (`Ctrl+Shift+P`), type `Git: Push`, and press Enter.
  - VS Code will prompt you to select the remote repository (GitHub repository) you want to push your changes to. Select it and press Enter.
  - If you haven't authenticated with GitHub in VS Code, it will prompt you to sign in using your GitHub credentials.

Monitor Push Progress:

- VS Code will display the progress of the push operation in the Output view (Git section).

- **Branching and Merging**: Use VS Code's Git integration to create and switch branches (`Git: Checkout to...` in Command Palette) and merge branches (`Git: Merge Branch...` in Command Palette).

- **Viewing History**: Use the Source Control view to view commit history (`...` next to `Changes`), compare changes (`Compare Changes` option), and navigate through commit history.