Questions:

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

Answer 1
To download and install Visual Studio Code (VS Code) on Windows 11, follow these steps:
 Prerequisites
1. Administrator Access: Ensure you have administrator rights on your Windows 11 machine to install software.
2. .NET Framework: While VS Code itself does not require the .NET Framework, some extensions might. It’s generally a good idea to ensure that you have the latest version of the .NET Framework installed. You can check this through Windows Update or by downloading it directly from the [Microsoft website](https://dotnet.microsoft.com/download/dotnet-framework).

Steps to Download and Install Visual Studio Code

1. Open Your Web Browser:
   - Open your preferred web browser (e.g., Edge, Chrome, Firefox).

2. Go to the Visual Studio Code Website:
   - Navigate to the official Visual Studio Code download page: https://code.visualstudio.com/.

3. Download Visual Studio Code:
   - On the homepage, you’ll see a prominent download button that automatically detects your operating system.
   - Click the “Download for Windows” button. This will download the VS Code installer (`.exe` file).

4. Run the Installer:
   - Locate the downloaded `VSCodeSetup.exe` file (usually in your Downloads folder) and double-click it to start the installation.

5. Installation Setup:
   - A setup wizard will open. Follow these steps in the wizard:
     - Step 1: Read and accept the license agreement, then click “Next”.
     - Step 2: Choose the installation location or leave it as default, then click “Next”.
     - Step 3: Select additional tasks:
       - Add "Open with Code" action to the Windows Explorer file context menu (recommended).
       - Add "Open with Code" action to the Windows Explorer directory context menu (recommended).
       - Register Code as an editor for supported file types (recommended).

    - Add to PATH (this allows you to open VS Code from the command line).
  - Click “Next”, then “Install” to begin the installation.

6. Complete Installation:
  - Once the installation is complete, you can choose to launch Visual Studio Code immediately by checking the appropriate box and clicking “Finish”.

First-time Setup
1. Launch Visual Studio Code:
  - If you didn’t choose to launch VS Code at the end of the installation, you can open it from the Start menu or desktop shortcut.

2. Configure Settings and Extensions:
  - Welcome Page: On first launch, VS Code will show a Welcome page with useful tips and shortcuts.
  - Theme and Interface: Customize your theme and interface settings from the settings icon in the lower left corner or by pressing `Ctrl + ,` (comma).
  - Install Extensions: Enhance functionality by installing extensions. Click on the Extensions icon on the left sidebar (or press `Ctrl + Shift + X`) and search for extensions you need (e.g., Python, C#, JavaScript).

3. Set Up Your Workspace:
  - Open a folder or create a new file to start coding.
  - Customize your workspace settings and layout as needed.

Additional Configuration (Optional)
- Version Control Integration: If you use Git, you can integrate it with VS Code. Ensure Git is installed on your system and then configure it in the VS Code settings.
- Settings Sync: If you use VS Code on multiple devices, you can enable Settings Sync to synchronize your settings, extensions, and configurations across devices.

Question 2
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.
Command Palette:

Answer 2
Visual Studio Code (VS Code) has a user-friendly and customizable interface designed to streamline coding tasks. Here are the main components of the VS Code user interface and their purposes:

Main Components of the VS Code User Interface

1. Activity Bar:
  - Location: The vertical bar on the far left side of the window.
  - Purpose: Provides quick access to different views and functionalities. By default, it includes icons for:
    - Explorer: Manages files and folders in your workspace.

  - Search: Performs searches across files in your workspace.
  - Source Control: Manages version control using Git or other source control providers.
  - Run and Debug: Starts debugging configurations and sessions.
  - Extensions: Manages and installs VS Code extensions.
- Customization: Users can customize which icons appear here and can access more views from this bar.

2. Side Bar:
   - Location: To the right of the Activity Bar.
   - Purpose: Displays the content of the selected view from the Activity Bar. For example:
     - Explorer View: Shows the file and folder structure of the workspace.
     - Search View: Displays search results.
     - Source Control View: Shows version control status and operations.
     - Extensions View: Lists installed and recommended extensions.
   - Functionality: Allows detailed interaction with the selected view, such as opening files, committing changes, or managing extensions.

3. Editor Group:
   - Location: Central area of the interface.
   - Purpose: The main workspace where you write and edit code.
   - Features:
     - Supports multiple editors side-by-side (split view).
     - Tabs at the top allow switching between open files.
     - Code editing features like syntax highlighting, IntelliSense, and code folding.
   - Customization: Users can open, close, and arrange editors to fit their workflow.

4. Status Bar:
   - Location: The horizontal bar at the bottom of the window.
   - Purpose: Provides information about the current workspace and open file. This includes:
     - Current programming language mode.
     - Line and column numbers of the cursor position.
     - Git branch and status.
     - Errors and warnings in the file.
     - Notifications for extensions.
   - **Interactive Elements**: Some items are interactive, allowing quick access to relevant settings or commands.

5. Command Palette:
   - Activation: Accessed via `Ctrl + Shift + P` (Windows/Linux) or `Cmd + Shift + P` (Mac).
   - Purpose: Provides a quick way to execute commands and access functionality without navigating the menu or interface.
   - Features:
     - Search for and execute any command within VS Code.
     - Access recently used commands.
     - Directly open files or settings.
   - **Usage**: Particularly useful for keyboard-centric workflows and finding commands that are not bound to specific keys or easily accessible from the interface.

Question 3
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.
Extensions in VS Code:

Answer 3

Command Palette in VS Code

The Command Palette in Visual Studio Code is a powerful tool that provides quick access to all available commands, settings, and functionality within the editor. It allows users to perform a wide range of tasks without navigating through menus or remembering keyboard shortcuts.

Accessing the Command Palette

- Windows/Linux: Press `Ctrl + Shift + P`.
- Mac: Press `Cmd + Shift + P`.

Alternatively, you can access it from the menu: `View > Command Palette`.

Examples of Common Tasks Using the Command Palette

1. Opening Files:
   - Type `>Open File` and select the file you want to open.

2. Changing Language Mode:
   - Type `>Change Language Mode` to set the syntax highlighting and features for a specific programming language.

3. Running Code:
   - Type `>Run Task` to execute predefined tasks (e.g., build scripts, tests).

4. Installing Extensions:
   - Type `>Extensions: Install Extensions` to open the Extensions view and search for new extensions to install.

5. Git Commands:
   - Type `>Git: Commit` to commit changes.
   - Type `>Git: Pull` to pull the latest changes from a remote repository.

6. Opening Settings:
   - Type `>Preferences: Open Settings (UI)` to open the settings in a graphical interface.
   - Type `>Preferences: Open Settings (JSON)` to open the settings file in JSON format.

7. Formatting Code:
   - Type `>Format Document` to format the currently open file according to the selected code style.

8. Starting Debugging:
   - Type `>Debug: Start Debugging` to begin a debugging session with the current configuration.

Extensions in VS Code

Extensions are a key feature of Visual Studio Code, allowing users to enhance and customize the editor's functionality. They can add support for new programming languages, debuggers, tools, and other utilities.

Managing Extensions

- Accessing Extensions View: Click the Extensions icon in the Activity Bar on the side of the window or press `Ctrl + Shift + X` (Windows/Linux) or `Cmd + Shift + X` (Mac).
- Installing Extensions:
  - Search for extensions in the Extensions view and click the "Install" button.
  - Use the Command Palette: Type `>Extensions: Install Extensions` and search for the desired extension.

Popular Extension Categories

1. Programming Language Support:
   - Python, JavaScript, TypeScript, C++, Java, and many more.

2. Linters and Formatters:
   - ESLint, Prettier, Pylint, Black.

3. Version Control:
   - GitLens (enhances Git integration), Git Graph (visualize Git branches).

4. Debugging:
   - Debugger for Chrome, Python, Node.js.

5. Snippets and Code Completion:
   - JavaScript (ES6) code snippets, Python snippets, and many other language-specific snippets.

6. Themes and Icons:
   - Material Theme, One Dark Pro, VSCode Icons.

7. Productivity Tools:
   - Live Server (for web development), Path Intellisense (auto-completes filenames), Bracket Pair Colorizer.

Question 4
Discuss the role of extensions in VS Code. How can users find, install, and manage extensions? Provide examples of essential extensions for web development.

Integrated Terminal:

Answer 4
M Role of Extensions in VS Code

Extensions play a crucial role in Visual Studio Code (VS Code) by allowing users to enhance and customize the functionality of the editor. They enable support for additional programming languages, frameworks, debuggers, and other tools, making VS Code a versatile and powerful development environment.

Finding, Installing, and Managing Extensions

Finding Extensions
- Extensions View: Open the Extensions view by clicking the Extensions icon in the Activity Bar on the side of the window or by pressing `Ctrl + Shift + X` (Windows/Linux) or `Cmd + Shift + X` (Mac).
- Marketplace: Browse the [Visual Studio Code Marketplace](https://marketplace.visualstudio.com/VSCode) online to find extensions.

Installing Extensions
1. Using the Extensions View:
   - In the Extensions view, type the name or category of the extension in the search bar.
   - Browse the results, and click the "Install" button next to the desired extension.

2. Using the Command Palette:
   - Press `Ctrl + Shift + P` (Windows/Linux) or `Cmd + Shift + P` (Mac) to open the Command Palette.
   - 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 installed extensions as needed.
- Update: Check for updates in the Extensions view, and click "Update" to get the latest version.
- Uninstall: Remove an extension by clicking the "Uninstall" button in the Extensions view.

Essential Extensions for Web Development

1. Language Support:
   - ESLint: Integrates ESLint into VS Code to provide linting support for JavaScript and TypeScript.
   - Prettier - Code Formatter: Automatically formats your code to ensure consistency.
   - HTML CSS Support: Provides IntelliSense and validation for HTML class names with CSS.

2. Frameworks and Libraries:
   - Vetur: Adds Vue.js support, including IntelliSense, debugging, and snippets.

- React Native Tools: Adds support for React Native development, including IntelliSense and debugging.

3. Version Control:
   - GitLens: Enhances Git capabilities in VS Code, providing rich information about commits, authors, and files.
   - **GitHub Pull Requests and Issues**: Manages GitHub pull requests and issues directly within VS Code.

4. Productivity:
   - Live Server: Launches a local development server with live reload capability for static and dynamic pages.
   - Path Intellisense: Auto-completes filenames as you type paths in your code.
   - Bracket Pair Colorizer: Highlights matching brackets to improve code readability.

Integrated Terminal

The Integrated Terminal in VS Code provides a convenient way to access a command-line interface directly within the editor.

Accessing the Integrated Terminal
- Shortcut: Press `Ctrl + ` (backtick) on Windows/Linux or `Cmd + ` (backtick) on Mac.
- Menu: Go to `View > Terminal`.

Features and Usage
- Multiple Terminals: Open multiple terminal sessions and switch between them using tabs.
- Shell Customization: Choose your preferred shell (e.g., PowerShell, Command Prompt, Bash) in the terminal settings.
- Split Terminal: Split the terminal window to run multiple commands side by side.
- Task Running: Execute build scripts, tests, and other tasks directly from the terminal.

Question 5
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?
File and Folder Management:

Answer 5
Opening and Using the Integrated Terminal in VS Code

Opening the Integrated Terminal

1. Shortcut:
   - Windows/Linux: Press `Ctrl + ` (backtick).
   - Mac: Press `Cmd + ` (backtick).

2. Menu:
   - Go to `View > Terminal` from the top menu.

3. Command Palette:
   - Press `Ctrl + Shift + P` (Windows/Linux) or `Cmd + Shift + P` (Mac).
   - Type `>Terminal: Create New Integrated Terminal` and press Enter.

Using the Integrated Terminal

1. Basic Commands:
   - You can run any command you would typically run in your default shell (e.g., Command Prompt, PowerShell, Bash) directly in the integrated terminal.

2. Multiple Terminals:
   - Open multiple terminals by clicking the "+" icon in the terminal tab bar.
   - Switch between terminals using the dropdown menu in the terminal tab bar or by clicking on the terminal tabs.

3. Split Terminals:
   - Click the split terminal icon (two vertical bars) to split the terminal into multiple panes.
   - Each pane operates independently, allowing you to run different commands side by side.

4. Customizing the Terminal:
   - You can change the default shell by going to `File > Preferences > Settings` and searching for `terminal.integrated.shell.windows`, `terminal.integrated.shell.linux`, or `terminal.integrated.shell.osx` based on your operating system.
   - Adjust other settings like font size, cursor style, and integrated terminal behavior in the settings.

Advantages of Using the Integrated Terminal Compared to an External Terminal

1. Convenience:
   - Accessing the terminal within the same window eliminates the need to switch between different applications, streamlining your workflow.

2. Context Awareness:
   - The integrated terminal opens in the context of the currently active workspace, ensuring that commands are run in the correct directory without needing additional navigation.

3. Project Management:
   - You can have multiple terminals open in different directories of your project, making it easy to manage various parts of a larger project simultaneously.

4. Integration with VS Code Features:
   - The terminal is tightly integrated with other VS Code features like debugging, task running, and version control, providing a seamless development experience.

5. Customization and Extensions:
   - Easily customize the terminal settings to suit your preferences and extend its functionality using VS Code extensions.

File and Folder Management in VS Code

1. Explorer View:
   - Access the Explorer view by clicking the file icon in the Activity Bar or by pressing `Ctrl + Shift + E`.

2. Opening Files and Folders:
   - File: Use `File > Open File` or press `Ctrl + O` to open individual files.
   - Folder: Use `File > Open Folder` or press `Ctrl + K, Ctrl + O` to open an entire folder as a workspace.

3. Creating Files and Folders:
   - Right-click in the Explorer view and select "New File" or "New Folder".
   - Alternatively, press `Ctrl + N` for a new file and save it with `Ctrl + S`.

4. Renaming and Deleting:
   - Right-click a file or folder in the Explorer view and select "Rename" or "Delete".
   - Press `F2` to rename a selected file or folder, and press `Delete` to remove it.

5. Moving and Copying:
   - Drag and drop files and folders within the Explorer view to move them.
   - Use `Ctrl + C` to copy and `Ctrl + V` to paste files and folders within the Explorer view.

6. Search and Replace:
   - Use `Ctrl + F` to search within the currently open file.
   - Use `Ctrl + Shift + F` to search across all files in the workspace.
   - Use `Ctrl + H` to open the search and replace feature within the current file.

7. Integrated Source Control:
   - Manage version control directly within VS Code, view changes, stage, commit, and push/pull changes to remote repositories using the Source Control view.

Question 6
Explain how to create, open, and manage files and folders in VS Code. How can users navigate between different files and directories efficiently?
Settings and Preferences:

Answer 6
Creating, Opening, and Managing Files and Folders in VS Code

Creating Files and Folders

1. Using the Explorer View:
   - Open the Explorer view by clicking the file icon in the Activity Bar or by pressing `Ctrl + Shift + E`.
   - Right-click on a folder (or the root of the workspace) and select "New File" to create a new file or "New Folder" to create a new folder.
   - Alternatively, press `Ctrl + N` to create a new file and save it using `Ctrl + S`.

2. Using the Command Palette:
   - Press `Ctrl + Shift + P` to open the Command Palette.
   - Type `>File: New File` or `>File: New Folder` and press Enter.

3. Using Keyboard Shortcuts:
   - Create a new file: `Ctrl + N`.
   - Save the new file: `Ctrl + S`.
   - Create a new folder: Right-click in the Explorer and select "New Folder".

Opening Files and Folders

1. Using the Explorer View:
   - Double-click on a file to open it.
   - Click the folder icon next to a folder to expand it and see its contents.

2. Using the Menu:
   - Open a file: `File > Open File` or press `Ctrl + O`.
   - Open a folder: `File > Open Folder` or press `Ctrl + K, Ctrl + O`.

3. Using the Command Palette:
   - Press `Ctrl + Shift + P`.
   - Type `>File: Open File` or `>File: Open Folder` and press Enter.

4. Quick Open:
   - Press `Ctrl + P` to quickly open files by typing the file name.

Managing Files and Folders

1. Renaming:
   - Right-click on the file or folder and select "Rename".
   - Alternatively, select the file or folder and press `F2`.

2. Deleting:
   - Right-click on the file or folder and select "Delete".
   - Alternatively, select the file or folder and press `Delete`.

3. Moving and Copying:
   - Drag and drop files and folders within the Explorer view to move them.
   - Use `Ctrl + C` to copy and `Ctrl + V` to paste files and folders within the Explorer view.

4. Search and Replace:
   - Use `Ctrl + F` to search within the currently open file.
   - Use `Ctrl + Shift + F` to search across all files in the workspace.
   - Use `Ctrl + H` to open the search and replace feature within the current file.

Navigating Between Different Files and Directories Efficiently

1. Quick Open:
   - Press `Ctrl + P` to quickly open files by typing the file name.

2. Go to Symbol:
   - Press `Ctrl + Shift + O` to navigate to a symbol in the currently open file.

3. Go to Definition:
   - Press `F12` or right-click and select "Go to Definition" to navigate to the definition of a symbol.

4. File Explorer:
   - Use the Explorer view to browse and navigate through directories and files.

5. Breadcrumb Navigation:
   - Use the breadcrumb navigation at the top of the editor to quickly navigate to parent directories and files.

6. Tabs and Split View:
   - Open multiple files in tabs and use split view (right-click on a tab and select "Split Right" or "Split Down") to view and edit files side by side.

Settings and Preferences

Accessing Settings

1. Settings UI:
   - Go to `File > Preferences > Settings` or press `Ctrl + ,` (comma).
   - This opens the settings in a graphical interface.

2. Settings JSON:
   - Open the Command Palette with `Ctrl + Shift + P` and type `>Preferences: Open Settings (JSON)`.
   - This opens the `settings.json` file for direct editing.

Common Settings

1. Editor Settings:
   - Font size, theme, word wrap, line numbers, etc.
   - Example: `"editor.fontSize": 14`, `"editor.wordWrap": "on"`.

2. Extensions:
   - Configure settings specific to installed extensions.
   - Example: `"python.pythonPath": "C:\\Python39\\python.exe"` for Python extension.

3. Keybindings:
   - Customize keyboard shortcuts via `File > Preferences > Keyboard Shortcuts` or `Ctrl + K, Ctrl + S`.

4. Workspace Settings:
   - Specific to the current workspace, saved in `.vscode/settings.json`.

Sync Settings

1. Settings Sync:
   - Sync your settings, keybindings, extensions, and snippets across multiple devices.
   - Enable Settings Sync from the gear icon in the lower-left corner and follow the prompts.

Question 7
Where can users find and customise settings in VS Code? Provide examples of how to change the theme, font size, and keybindings.
Debugging in VS Code:

Answer 7
Finding and Customizing Settings in VS Code

Accessing Settings

1. Settings UI:
   - Windows/Linux: Go to `File > Preferences > Settings` or press `Ctrl + ,` (comma).

2. Settings JSON:
   - Open the Command Palette with `Ctrl + Shift + P`.
   - Type `>Preferences: Open Settings (JSON)` and press Enter.
   - This opens the `settings.json` file where you can directly edit settings in JSON format.

Examples of Customizing Settings

1. Changing the Theme:
   - Settings UI:
     1. Open the Settings UI.
     2. In the search bar, type "Color Theme".
     3. Click on "Color Theme" under "Preferences" and choose a theme from the list.
   - Command Palette:
     1. Press `Ctrl + Shift + P`.
     2. Type `>Preferences: Color Theme` and press Enter.
     3. Select a theme from the list.

2. Changing the Font Size:
   - Settings UI:
     1. Open the Settings UI.
     2. In the search bar, type "Font Size".
     3. Under "Text Editor", adjust the `Editor: Font Size` setting.
   - Settings JSON:
     1. Open the `settings.json` file.
     2. Add or modify the following line:

```json
"editor.fontSize": 14
```
3. Change the value to your preferred font size.

3. Customizing Keybindings:
- Keybindings UI:
  1. Go to `File > Preferences > Keyboard Shortcuts` or press `Ctrl + K, Ctrl + S`.
  2. In the search bar, type the command you want to rebind.
  3. Click the pencil icon next to the command, then press the desired key combination.
- Keybindings JSON:
  1. Open the Command Palette with `Ctrl + Shift + P`.
  2. Type `>Preferences: Open Keyboard Shortcuts (JSON)` and press Enter.
  3. Add or modify keybindings in the `keybindings.json` file. Example:
```json
[
  {
    "key": "ctrl+alt+n",
    "command": "workbench.action.files.newUntitledFile"
  }
]
```

Debugging in VS Code

Setting Up Debugging

1. Install Debugger Extension:
- Depending on the language you are using, you might need to install a specific debugger extension. For example:
  - Python: Install the "Python" extension.
  - JavaScript/TypeScript: The built-in Node.js debugger can be used.

2. Open the Debug View:
- Click on the Debug icon in the Activity Bar on the side of the window or press `Ctrl + Shift + D`.

3. Configure Launch Configuration:
- In the Debug view, click on the gear icon to open the `launch.json` file.
- VS Code will generate a basic `launch.json` file if it doesn’t exist. Customize this file to match your debugging needs.
- Example for Node.js:
```json
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Launch Program",
      "program": "${workspaceFolder}/app.js"
    }
  ]
}
```

Starting a Debugging Session

1. Set Breakpoints:
   - Click in the gutter to the left of the line numbers in the code editor to set breakpoints.

2. Start Debugging:
   - Press `F5` to start the debugging session.
   - Alternatively, use the Debug view to select and start a debug configuration.

3. Using Debugging Tools:
   - Debug Toolbar: Use the toolbar to control the debugging session (continue, step over, step into, step out, restart, stop).
   - **Variables Pane**: View and evaluate variables in the current scope.
   - Watch Pane: Add expressions to watch and evaluate their values during debugging.
   - Call Stack Pane: View the call stack and navigate between stack frames.
   - Breakpoints Pane: Manage breakpoints (enable/disable, remove).

Question 8
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?
Using Source Control:

Answer 8
Setting Up and Starting Debugging a Simple Program in VS Code

Example: Debugging a Simple Node.js Program

1. Install VS Code and Node.js:
   - Ensure Visual Studio Code is installed.
   - Install Node.js from nodejs.org.

2. Install the Node.js Extension (if not already installed):
   - Open VS Code.
   - Go to the Extensions view by clicking the Extensions icon in the Activity Bar or pressing `Ctrl + Shift + X`.
   - Search for "Node.js Extension Pack" and install it.

3. Create a Simple Node.js Program:
   - Open VS Code and create a new folder for your project (`File > Open Folder`).
   - Inside the folder, create a new file named `app.js` and add the following code:

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

const userName = 'World';
greet(userName);
```

4. Open the Debug View:
   - Click on the Debug icon in the Activity Bar or press `Ctrl + Shift + D`.

5. Configure the Debugger:
   - Click the gear icon in the Debug view to create a `launch.json` file.
   - Select "Node.js" as the environment.
   - VS Code will generate a `launch.json` file. It should look like this:

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

6. Set Breakpoints:
   - Open `app.js` and click in the gutter to the left of the line numbers to set a breakpoint. Set a breakpoint on the line `console.log(`Hello, ${name}!`);`.

7. Start Debugging:
   - Press `F5` to start the debugging session. Alternatively, click the green play button in the Debug view.
   - The program will run and pause at the breakpoint you set.

8. Using the Debugging Tools:
   - Debug Toolbar: Use the toolbar to control the session (Continue, Step Over, Step Into, Step Out, Restart, Stop).
   - Variables Pane: View and evaluate variables in the current scope.
   - Watch Pane: Add expressions to watch and evaluate their values during debugging.
   - Call Stack Pane: View the call stack and navigate between stack frames.
   - **Breakpoints Pane**: Manage breakpoints (enable/disable, remove).

Key Debugging Features in VS Code

1. Breakpoints:
   - Set breakpoints to pause execution at specific lines.
   - Conditional breakpoints to pause execution when certain conditions are met.

2. Step Controls:
   - Continue: Resume program execution until the next breakpoint.
   - Step Over: Execute the current line and move to the next line.
   - Step Into: Step into the function call on the current line.
   - Step Out: Step out of the current function and return to the calling line.

3. Variables:
   - Inspect and modify variables in the current scope.
   - Hover over variables in the code to see their values.

4. Watch Expressions:
   - Add expressions to watch and evaluate their values during the debugging session.

5. Call Stack:
   - View the call stack to understand the sequence of function calls.

6. Debug Console:
   - Execute commands and evaluate expressions in the context of the running program.

7. Exception Handling:
   - Configure the debugger to break on exceptions (all or uncaught exceptions).

Using Source Control in VS Code

1. Git Integration:
   - VS Code has built-in Git support. Open the Source Control view by clicking the Source Control icon in the Activity Bar or pressing `Ctrl + Shift + G`.

2. Initializing a Repository:
   - If your project folder is not yet a Git repository, click "Initialize Repository" in the Source Control view.

3. Cloning a Repository:
   - Open the Command Palette (`Ctrl + Shift + P`).
   - Type `>Git: Clone` and enter the repository URL.
   - Choose a folder to clone the repository into.

4. Staging and Committing Changes:
   - Stage Changes: Click the "+" icon next to the file in the Source Control view to stage it.
   - Commit Changes: Enter a commit message in the input box at the top of the Source Control view and click the checkmark icon to commit the changes.

5. Branching and Merging:
   - Click on the branch name in the bottom left corner to switch branches or create a new branch.
   - Use the Command Palette (`Ctrl + Shift + P`) and type `>Git: Merge Branch` to merge branches.

6. Pushing and Pulling Changes:
   - Use the icons at the top of the Source Control view to push and pull changes to/from the remote repository.

7. Resolving Conflicts:
   - When conflicts occur, VS Code highlights them in the editor. Use the inline buttons to accept incoming or current changes.

Question 9
How can users integrate Git with VS Code for version control? Describe the process of initialising a repository, making commits, and pushing changes to GitHub.
Submission Guidelines:

Answer 9
Integrating Git with VS Code for Version Control

1. Setting Up Git

- Install Git: Download and install Git from git-scm.com.
- Configure Git: Open a terminal and run the following commands to set up your Git username and email:
  ```sh
  git config --global user.name "Your Name"
  git config --global user.email "your.email@example.com"
  ```

2. Initializing a Repository

1. Open Your Project Folder in VS Code:
   - Go to `File > Open Folder` and select your project folder.

2. Initialize Git Repository:
   - Open the Source Control view by clicking the Source Control icon in the Activity Bar or by pressing `Ctrl + Shift + G`.
   - Click "Initialize Repository" in the Source Control view. This creates a `.git` directory in your project folder, which stores the version control data.

3. Create a `.gitignore` File:
   - Create a file named `.gitignore` in the root of your project folder to specify which files and directories to ignore.
   - Example content for a Node.js project:
     ```
     ```

```
node_modules/
.env
```

3. Making Commits

1. Staging Changes:
   - In the Source Control view, you will see a list of changes. Click the "+" icon next to each file to stage it, or click the "+" icon at the top to stage all changes.

2. Writing Commit Messages:
   - After staging your changes, type a commit message in the input box at the top of the Source Control view.

3. Committing Changes:
   - Click the checkmark icon at the top of the Source Control view to commit your staged changes.

4. Pushing Changes to GitHub

1. Creating a GitHub Repository:
   - Go to [GitHub](https://github.com/) and create a new repository.

2. Adding Remote Repository:
   - Open a terminal in VS Code.
   - Add the GitHub repository as a remote by running the following command, replacing `<URL>` with your repository's URL:
```sh
git remote add origin <URL>
```

3. Pushing Changes:
   - Push your changes to GitHub by running the following command:
```sh
git push -u origin master
```
   - For subsequent pushes, you can simply run:
```sh
git push
```

Submission Guidelines

When submitting code or changes to a repository, follow these guidelines:

1. Write Clear Commit Messages:
   - Use concise, descriptive commit messages that explain the purpose of the changes. For example:

```
Add user authentication
```

2. Commit Frequently:
   - Make small, incremental commits frequently to ensure changes are easy to track and manage.

3. Use Branches:
   - For new features or bug fixes, create a new branch. This keeps the `master` or `main` branch stable and clean.
   - Example:
     ```sh
     git checkout -b new-feature
     ```

4. Pull Requests:
   - When ready to merge changes into the main branch, create a pull request on GitHub. This allows for code review and discussion before merging.

5. Follow Project Conventions:
   - Adhere to any specific conventions or guidelines set by the project, such as code style, commit message format, and branch naming.