Installation and Navigation of Visual Studio Code (VS Code) Instructions: Answer the following questions based on your understanding of the installation and navigation of Visual Studio Code (VS Code). Provide detailed explanations and examples where appropriate.

Questions:

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.

Downloading and installing Visual Studio Code (VS Code) on a Windows 11 operating system:

Prerequisites:

1. Windows 11 Operating System: Ensure that your system is running Windows 11.

2. Administrator Access: You need to have administrator privileges to install software.

Steps to Download and Install Visual Studio Code:

1. Open Your Web Browser:

- Launch any web browser like Microsoft Edge, Google Chrome, or Firefox.

2. Navigate to the VS Code Website:

- Go to the official Visual Studio Code website: https://code.visualstudio.com/.

3. Download the Installer:

- On the VS Code homepage, click on the "Download for Windows" button. This will download the installer for Windows.

4. Run the Installer:

- Once the download is complete, navigate to your Downloads folder (or the location where you saved the installer).
- Double-click on the downloaded installer file (usually named “VSCodeUserSetup-x64-x.x.x.exe”).

5. Start the Installation Process:

- A setup window will appear. Click "Next" to start the installation process.

6. Accept the License Agreement:

- Read the license agreement and click on the "I accept the agreement" option, then click "Next".

7. Select Installation Location:

- Choose the destination folder where you want to install VS Code. The default location is usually fine. Click "Next" to continue.

8. Select Additional Tasks:

- You can choose additional tasks such as:
- Creating a desktop icon.
- Adding "Open with Code" actions to the context menu.
- Adding the installation to the PATH (important for using “code” command in the terminal).
- Select the options you prefer and click "Next".

9. Install Visual Studio Code:

- Click the "Install" button to begin the installation.

10. Complete the Installation:

- Once the installation is complete, you can choose to launch VS Code immediately by checking the "Launch Visual Studio Code" box and then click "Finish".

11. Launch Visual Studio Code:

- If you didn't choose to launch VS Code during the final step of the installation, you can open it from the Start menu or by double-clicking the desktop icon (if you created one).

Post-Installation Setup (Optional):

1. Install Extensions:

- Open VS Code and go to the Extensions view by clicking on the square icon on the sidebar or pressing "Ctrl+Shift+X".
- Search for and install any extensions you need, such as Python, JavaScript, or C# extensions.

2. Customize Settings:

- You can customize the settings of VS Code by going to "File" > "Preferences" > "Settings" or by pressing "Ctrl+",.

3. Install Git:

- If you plan to use version control, install Git from https://git-scm.com/. Follow the instructions to install Git and then restart VS Code to ensure it recognizes Git.

Visual Studio Code is now installed on your Windows 11 system and ready for development.

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.

Initial Configurations and Settings. Install Essential Extensions:

1. Language Support:
Install extensions for the programming languages you will be using. Some popular ones include:

Python: Python by Microsoft

JavaScript/TypeScript: ESLint, Prettier - Code formatter

C/C++: C/C++ by Microsoft

Java: Java Extension Pack

HTML/CSS: HTML CSS Support, Live Server

2. Version Control:
If you use Git, install the GitLens extension.

Snippets and Utilities: Install helpful utilities such as IntelliCode, Path Intellisense, and Bracket Pair Colorizer.

3. Theme and Icons:
Theme: Go to File > Preferences > Color Theme or press Ctrl+K Ctrl+T to choose a theme that is comfortable for your eyes. Popular themes include Dracula Official, One Dark Pro, and Solarized Dark.

Icons: Customize your file icons by going to File > Preferences > File Icon Theme and choosing one like VSCode Icons.

4. Editor Settings:

Tab Size and Format: Set the tab size and choose whether to use spaces or tabs. You can configure this in File > Preferences > Settings, then search for editor.tabSize and editor.insertSpaces.

5. Auto Save:
Enable auto save by setting "files.autoSave": "afterDelay" in your settings.

Code Formatting: Configure code formatting settings, such as enabling format on save with "editor.formatOnSave": true.

6. Terminal Settings:
Customize the integrated terminal by going to File > Preferences > Settings and searching for terminal.integrated.shell.windows. You can set it to use Git Bash or another preferred shell.

7. Workspace Settings:
Set up workspace-specific settings by creating a .vscode folder in your project directory and adding settings in settings.json.

Important Settings:

1. Settings Sync:
Enable Settings Sync to synchronize your settings, extensions, and keyboard shortcuts across devices. Go to File > Preferences > Settings Sync and follow the prompts to set it up.

2. Linting and Intellisense:
Configure linters like ESLint for JavaScript or Flake8 for Python. Install the relevant extensions and set up configuration files (.eslintrc.json or .flake8).

Ensure IntelliSense is working optimally by configuring settings like "editor.quickSuggestions" and "editor.parameterHints".

3. Code Completion and Snippets:
Customize code completion settings and add user-defined snippets. Go to File > Preferences > User Snippets to create or edit snippets for different languages.

4. Debugger Configuration:
Set up debugging configurations by going to the Run view (Ctrl+Shift+D) and creating or editing launch.json files for your projects.

5. Version Control Integration:
Configure Git integration by setting user name and email in the terminal:
sh

Copy code
git config --global user.name "Your Name"
git config --global user.email "you@example.com"

6. Keybindings:
Customize keyboard shortcuts by going to File > Preferences > Keyboard Shortcuts or pressing Ctrl+K Ctrl+S.

Recommended Extensions:

1. Code Quality and Productivity:
Prettier - Code formatter
ESLint
Path Intellisense
IntelliCode
GitLens

2. Language-Specific Extensions:

Python: Python, Pylance

JavaScript/TypeScript: ESLint, TypeScript Hero

C/C++: C/C++, CMake Tools

Java: Java Extension Pack

Web Development: HTML CSS Support, Live Server

3. Theming and Customization:

Dracula Official

One Dark Pro

VSCode Icons

By configuring these settings and installing the recommended extensions, you will have a well-optimised environment tailored to your coding needs in Visual Studio Code.

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.

1. Activity Bar

The Activity Bar is located on the far left side of the VS Code window. It provides access to different views and features through icons, each representing a specific function or section of the IDE.

- **Purpose:** The Activity Bar serves as a quick navigation tool, allowing users to switch between various functionalities such as Explorer, Search, Source Control, Run and Debug, Extensions, and other custom views.

- **Common Icons:**
    - **Explorer:** Displays files and folders in the workspace.
    - **Search:** Provides a powerful search and replace functionality.
    - **Source Control:** Integrates with version control systems like Git.
    - **Run and Debug:** Manages debugging sessions and configurations.
    - **Extensions:** Allows users to discover and manage extensions.

2. Side Bar

The Side Bar is positioned immediately to the right of the Activity Bar and changes its content based on the selected view in the Activity Bar.

- Purpose: The Side Bar provides detailed information and tools relevant to the current view. It displays file structures, search results, source control changes, debugging information, and extensions management.

- **Usage:**
    - **In Explorer View:** Shows the directory structure, open editors, and outline of the current file.
    - **In Search View:** Displays search results and replace options.
    - **In Source Control View:** Lists changes, branches, and repositories.
    - **In Run and Debug View:** Shows debugging controls, call stacks, variables, and breakpoints.
    - In Extensions View: Lists installed extensions and available extensions from the marketplace.

3. Editor Group

The Editor Group is the central area of the VS Code window where files are opened and edited. It supports multiple editors in separate tabs or split views.

- **Purpose:** The Editor Group is the primary workspace where code is written, edited, and reviewed. It can host multiple editors for different files, allowing side-by-side comparison and editing.

- **Features:**
    - **Tabs:** Each open file is represented by a tab at the top of the editor.
    - **Split View:** Users can split the editor horizontally or vertically to view and edit multiple files simultaneously.
    - **IntelliSense:** Provides code suggestions, autocompletion, and error checking.
    - **Syntax Highlighting:** Different colors and styles for various elements of the code, enhancing readability.
    - **Code Navigation:** Features like Go to Definition, Peek Definition, and Find All References.

4. Status Bar

The Status Bar is located at the bottom of the VS Code window. It provides useful information about the current state of the editor and the workspace.

- **Purpose:** The Status Bar displays context-specific information and provides quick access to settings and actions related to the current file and workspace.

- **Common Indicators:**
    - **Line and Column Number:** Shows the current cursor position in the editor.
    - **Language Mode:** Indicates the programming language of the current file and allows switching between languages.
    - **Git Branch:** Displays the current Git branch and status.
    - **Encoding:** Shows the file encoding (e.g., UTF-8).

- **EOL (End of Line) Sequence:** Indicates the line-ending style (e.g., LF or CRLF).
- **Indentation:** Displays the current indentation settings (e.g., spaces or tabs).
- **Notifications:** Alerts and messages about the editor or extensions.

In Conclusion:

- **Activity Bar:** Left side, quick access to views and features.
- **Side Bar:** Next to the Activity Bar, shows detailed information and tools for the selected view.
- **Editor Group:** Central area, main workspace for writing and editing code.
- **Status Bar:** Bottom of the window, displays contextual information and quick actions.

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.

The Command Palette in Visual Studio Code (VS Code) is a powerful tool that allows users to access a wide range of commands and features without having to navigate through menus or remember complex keyboard shortcuts. It provides a quick way to execute commands and perform tasks efficiently.

Using the Command Palette

How to Access the Command Palette:
The Command Palette can be accessed in several ways:

1. **Using the Keyboard Shortcut:**

- Press “Ctrl+Shift+P” (Windows/Linux) or “Cmd+Shift+P” (macOS) to open the Command Palette.

2. Through the Menu:

- Click on View in the top menu and then select Command Palette.

3. Using the Command Prompt:

- Press F1 to open the Command Palette directly.

Examples of Common Tasks Using the Command Palette

The Command Palette can be used to perform a wide variety of tasks. Here are some common examples:

1. Opening a File:

- Type “> Open File” and select the desired file from the list.

2. Opening a Folder:

- Type “> Open Folder” and choose the folder you want to open.

3. Searching for a Command:

- Simply type the name of the command you are looking for. For example, type “> git” to see all git-related commands.

4. Changing the Colour Theme:

- Type “> Preferences: Colour Theme” and select a different theme from the list.

5. Installing Extensions:

- Type “> Extensions: Install Extensions” and search for the extension you want to install.

6. Running a Task:

- Type “> Tasks: Run Task” to see a list of available tasks and choose one to run.

7. Opening Settings:

- Type “> Preferences: Open Settings” to access the settings.

8. Formatting Code:

- Type “> Format Document” to format the entire document according to the code formatting rules.

9. Toggling the Integrated Terminal:

- Type “> View: Toggle Integrated Terminal” to show or hide the integrated terminal.

10. Debugging:

- Type “> Debug: Start Debugging” to start a debugging session.

Additionally:

- **Quick Access:**
  - Commands can be quickly accessed by typing the first few letters. For example, typing “> reload” will bring up the “Reload Window” command.
- **Run Saved Code Snippets:**
  - You can run saved code snippets by typing the name of the snippet.
- **Multi-cursor Editing:**

- Type “> Insert Cursor Below” or “> Insert Cursor Above” to add multiple cursors for simultaneous editing.

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.

Role of Extensions in VS Code

Extensions play a crucial role in enhancing the functionality and user experience of Visual Studio Code (VS Code). They allow users to customize and extend the capabilities of the editor to meet their specific development needs. Extensions can provide additional features, tools, and integrations that are not available in the default installation of VS Code.

Key Roles of Extensions:

1. **Language Support:** Extensions can add support for new programming languages, providing features like syntax highlighting, code completion, linting, and debugging.
2. **Productivity Tools:** Extensions can enhance productivity by adding features such as snippets, code formatting, task runners, and version control integrations.
3. **Themes and Icons:** Extensions can customize the look and feel of VS Code with different themes and icon packs.
4. **Integrations:** Extensions can integrate VS Code with external tools and services, such as Docker, Kubernetes, cloud platforms, and databases.
5. **Utilities:** Extensions can add various utilities such as live servers, REST clients, and terminal enhancements.

Finding, Installing, and Managing Extensions

Finding Extensions:

Users can find extensions in the VS Code marketplace, which can be accessed directly from within the editor or via the web.

1. Via VS Code:

    - Open the Extensions view by clicking the Extensions icon in the Activity Bar or by pressing Ctrl+Shift+X.
    - Use the search bar at the top of the Extensions view to find specific extensions or browse popular and recommended extensions.

2. Via Web:

    - Visit the Visual Studio Code Marketplace in a web browser.
    - Use the search bar and categories to explore available extensions.

Installing Extensions:

1. Via VS Code:

    - Find the desired extension in the Extensions view.
    - Click the "Install" button next to the extension name.

2. Via Web:

    - Find the desired extension in the marketplace.
    - Click the "Install" button, which will prompt you to open VS Code if not already open.

Managing Extensions:

1. Enabling/Disabling:

    - In the Extensions view, right-click on an installed extension and select "Enable" or "Disable".

2. Updating:

- Extensions will automatically check for updates, but users can manually check by clicking the gear icon on the installed extension and selecting "Check for Updates".

3. Uninstalling:

   - Right-click on the installed extension in the Extensions view and select "Uninstall".

4. Configuring:

   - Some extensions may have configurable settings. These can be accessed by clicking the gear icon on the extension and selecting "Extension Settings".

Essential Extensions for Web Development

Here are some essential extensions for web development, along with their roles:

1. Prettier - Code Formatter:

   - Role: Automatically formats code to ensure a consistent style.

2. ESLint:

   - Role: Integrates ESLint into VS Code for JavaScript/TypeScript linting.

3. Live Server:

   - Role: Launches a local development server with live reload feature.

4. Debugger for Chrome:

   - Role: Allows debugging JavaScript code in the Chrome browser from VS Code.

5. Path Intellisense:

   - Role: Provides autocompletion for file paths in the project.

6. HTML CSS Support:

   - Role: Enhances HTML and CSS support with Intellisense and validation.

7. JavaScript (ES6) Code Snippets:

   - Role: Provides code snippets for JavaScript in ES6 syntax.

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 and Using the Integrated Terminal in VS Code

The integrated terminal in Visual Studio Code (VS Code) allows you to run command-line tools from within the editor. This feature is convenient for developers as it keeps the workflow within a single window.

How to Open the Integrated Terminal

There are several ways to open the integrated terminal in VS Code:

1. **Using the Keyboard Shortcut:**
   - Press “Ctrl+” (Windows/Linux) or “Cmd+” (macOS).
2. **Through the Menu:**
   - Click on “View” in the top menu and then select “Terminal”.
3. **Command Palette:**
   - Open the Command Palette using “Ctrl+Shift+P” (Windows/Linux) or “Cmd+Shift+P” (macOS), then type “> View: Toggle Integrated Terminal” and select it.

Using the Integrated Terminal

Once the terminal is open, you can use it like any other terminal. Here are some basic tasks you can perform:

1. Running Commands:
    - You can run any command that you would in an external terminal, such as “ls”, “cd”, “npm install”, “git status”, etc.
2. Opening Multiple Terminals:
    - Click the “+” icon in the terminal tab bar to open a new terminal instance.
    - Use the dropdown menu next to the “+” icon to switch between terminal instances.
3. Splitting the Terminal:
    - Click the split icon in the terminal tab bar to split the terminal into multiple panes.
    - You can also right-click on a terminal tab and select “Split”.
4. Configuring the Terminal:
    - You can customize the terminal shell by going to “File” > “Preferences” > “Settings”, and then searching for "terminal.integrated.shell".

Advantages of Using the Integrated Terminal

Using the integrated terminal in VS Code offers several advantages over an external terminal:

1. Convenience:

- The integrated terminal allows you to stay within the VS Code environment, eliminating the need to switch between different applications.

2. Context Awareness:

   - The terminal is contextually aware of the workspace, making it easier to run commands relative to the project's root directory.

3. Synchronization:

   - When you open a new terminal, it opens in the current project's root directory by default, reducing the need for navigation.

4. Ease of Use:

   - Integrated terminal instances can be easily created, closed, and managed directly from within the editor.

5. Productivity:

   - Having the terminal integrated means you can quickly switch between coding and running commands, debugging, or running tests, streamlining your workflow.

6. Customizability:

   - The integrated terminal can be customized to use different shells (e.g., Bash, PowerShell, Zsh) and configured to match your development needs.

7. Visual Integration:

   - The terminal is part of the editor layout, allowing you to view output alongside your code without overlapping or obscuring windows.

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?

Creating, opening, and managing files and folders in Visual Studio Code (VS Code) is straightforward and essential for efficient development. Below is a detailed guide on how to perform these tasks and navigate between different files and directories effectively.

Creating Files and Folders

Creating Files:

1. **Using the Explorer:**
    - Open the Explorer view by clicking the Explorer icon in the Activity Bar or pressing “Ctrl+Shift+E”.
    - Right-click in the Explorer pane and select “New File” or click the new file icon (a file with a plus sign).
    - Enter the file name and press “Enter”.
2. **Using the Command Palette:**
    - Open the Command Palette with “Ctrl+Shift+P”.
    - Type “>New File” and press “Enter”.
    - Enter the file name and press “Enter”.
3. **Using Keyboard Shortcuts:**
    - Press “Ctrl+N” to create a new untitled file.
    - To save the file, press “Ctrl+S", enter the file name, and choose the directory.

Creating Folders:

1. **Using the Explorer:**

- Open the Explorer view (“Ctrl+Shift+E”).
- Right-click in the Explorer pane and select “New Folder” or click the new folder icon (a folder with a plus sign).
- Enter the folder name and press “Enter”.

Opening Files and Folders

Opening Files:

1. **Using the Explorer:**

   - Navigate through the directory structure in the Explorer pane.
   - Click on the file you wish to open.

2. **Using the Command Palette:**

   - Open the Command Palette with “Ctrl+Shift+P”.
   - Type “>Open File” and press “Enter”.
   - Navigate to the file and select it.

3. **Using Keyboard Shortcuts:**

   - Press “Ctrl+O” to open the file dialog.
   - Navigate to the desired file and open it.

4. Drag and Drop:

   - Drag a file from your file explorer (Windows Explorer, Finder, etc.) and drop it into the VS Code window.

Opening Folders:

1. **Using the File Menu:**

   - Go to “File > Open Folder”.
   - Navigate to and select the folder you want to open.

2. **Using the Command Palette:**

- Open the Command Palette with “Ctrl+Shift+P”.
- Type >Open Folder and press “Enter”.
- Navigate to and select the folder.

3. **Using Keyboard Shortcuts:**

- Press “Ctrl+K Ctrl+O” to open the folder dialog.
- Navigate to and select the folder.

Managing Files and Folders

1. **Renaming:**

- Right-click on the file or folder in the Explorer pane and select Rename.
- Alternatively, select the file or folder and press “F2”.

2. **Deleting:**

- Right-click on the file or folder in the Explorer pane and select “Delete”.
- Alternatively, select the file or folder and press “Delete”.

3. **Moving:**

- Drag and drop files or folders within the Explorer pane to move them.
- Alternatively, cut (“Ctrl+X”) and paste (“Ctrl+V”) files or folders to move them.

Navigating Between Files and Directories

1. **Explorer:**

- Use the Explorer pane to quickly navigate between files and directories by expanding and collapsing folders and clicking on files.

2. **File Tabs:**

    - Open files appear as tabs at the top of the Editor Group. Click on tabs to switch between open files.

3. **Quick Open:**

    - Press “Ctrl+P” to open the Quick Open dialog.
    - Start typing the file name, and VS Code will display a list of matching files. Select the desired file from the list.

4. **Go to Definition:**

    - Place the cursor on a symbol (e.g., a function or variable) and press “F12” to navigate to its definition.
    - Alternatively, right-click on the symbol and select Go to Definition.

5. **Go to Symbol:**

    - Press “Ctrl+Shift+O” to open the Go to Symbol in File dialog. This allows you to quickly navigate to symbols within the current file.

6. **Breadcrumb Navigation:**

    - The breadcrumb trail at the top of the editor shows the file path and symbols. Click on any part of the breadcrumb to navigate directly to it.

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.

One can find and modify settings through the Settings UI, settings.json file, or the Command Palette.

Accessing Settings

1. **Using the Settings UI:**
    - Keyboard Shortcut: Press Ctrl+, (Windows/Linux) or Cmd+, (macOS).
    - Through the Menu: Click on File > Preferences > Settings (Windows/Linux) or Code > Preferences > Settings (macOS).
2. **Using the Command Palette:**
    - Open the Command Palette using Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS), then type > Preferences: Open Settings (UI) and select it.
3. **Editing settings.json Directly:**
    - In the Settings UI, click on the {} icon in the top right corner to open and edit the settings.json file directly.

Examples of Customizations

Changing the Theme

1. **Using the Settings UI:**
    - Go to the Settings UI as described above.
    - In the search bar, type Color Theme.
    - Click on Color Theme and select the desired theme from the dropdown list.
2. **Using the Command Palette:**
    - Open the Command Palette with Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS).

- Type > Preferences: Color Theme and select the desired theme from the list.

Changing the Font Size

1. **Using the Settings UI:**

    - Access the Settings UI.
    - In the search bar, type Font Size.
    - Adjust the Editor: Font Size setting to the desired value.

2. **Using the settings.json File:**

    - Open the settings.json file.
    - Add or modify the following line: "editor.fontSize": 16
    - Replace “16” with your desired font size.

Changing Keybindings

1. **Using the Keybindings UI:**

    - Keyboard Shortcut: Press “Ctrl+K Ctrl+S” (Windows/Linux) or “Cmd+K Cmd+S” (macOS).
    - Through the Menu: Click on “File” > “Preferences” > “Keyboard Shortcuts” (Windows/Linux) or “Code” > “Preferences” > “Keyboard Shortcuts” (macOS).

2. **Using the Command Palette:**

    - Open the Command Palette with “Ctrl+Shift+P” (Windows/Linux) or “Cmd+Shift+P” (macOS).
    - Type “> Preferences: Open Keyboard Shortcuts” and select it.

3. **Customizing a Keybinding:**

- In the Keybindings UI, search for the command you want to change.
- Click the pencil icon next to the command to edit the keybinding.
- Press the new key combination you want to assign.

4. **Using the keybindings.json File:**

   - In the Keybindings UI, click on the “{}” icon in the top right corner to open and edit the “keybindings.json” file directly.
   - Add or modify the keybinding entry. For example:

     ```
     {
         "key": "ctrl+alt+n",
         "command": "workbench.action.files.newUntitledFile"
     }
     ```
   - Replace “ctrl+alt+n” with your desired key combination and “workbench.action.files.newUntitledFile” with the command you want to bind.

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?

Setting up and starting debugging a simple program in Visual Studio Code (VS Code) involves several steps, including writing the program, configuring the debugger, and running the debugger. Here's a step-by-step guide along with key debugging features available in VS Code.

Steps to Set Up and Start Debugging

1. Write Your Program

First, create a simple program in the language of your choice. For example, let's write a simple Python program:

```python
# sample.py
def greet(name):
    return f"Hello, {name}!"

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

2. Install the Necessary Extension

Ensure you have the necessary extension installed for your programming language. For Python, install the Python extension.

- Open the Extensions view (“Ctrl+Shift+X”).
- Search for "Python" and install the extension by Microsoft.

3. Open the Debug View

Open the Debug view by clicking the Debug icon in the Activity Bar on the side of the window or pressing “Ctrl+Shift+D”.

4. Create a Debug Configuration

To debug your program, you need to create a debug configuration.

- In the Debug view, click on the gear icon (“Run and Debug”).
- Select “Add Configuration” and choose the relevant configuration template for your programming language. For “Python, select Python: Current File”.

This will create a “.vscode/launch.json” file with a default configuration. Here is an example configuration for Python:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Current File",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal"
        }
    ]
}
```

5. Set Breakpoints

Set breakpoints in your code where you want the execution to pause. Click to the left of the line number in the editor or press “F9” with the cursor on the desired line.

6. Start Debugging

- Ensure your program file is open and active in the editor.
- In the Debug view, select your configuration from the dropdown menu.
- Click the green play button (“Start Debugging”) or press “F5”.

Key Debugging Features in VS Code

1. **Breakpoints:**
    - Set breakpoints by clicking in the gutter next to the line numbers or pressing “F9”. Breakpoints pause the execution of your code at specific lines, allowing you to inspect the state of the program.
2. **Watch:**
    - The Watch pane allows you to monitor specific expressions and variables. Add expressions to the Watch pane to evaluate them at each breakpoint.
3. **Call Stack:**

- The Call Stack pane shows the stack trace of your program at the current point of execution, allowing you to see the sequence of function calls that led to the current point.

4. **Variables:**

   - The Variables pane displays the current values of local and global variables. Expand objects to inspect their properties.

5. **Debug Console:**

   - The Debug Console allows you to execute arbitrary expressions in the context of the currently paused execution, helping you inspect and modify the state of the program.

6. **Step Controls:**

   - Use the step controls 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 the function call on the current line.
     - **Step Out (Shift+F11):** Step out of the current function back to the caller.
     - **Restart (Ctrl+Shift+F5):** Restart the debugging session.
     - **Stop (Shift+F5):** Stop the debugging session.

7. **Conditional Breakpoints:**

   - Set conditional breakpoints that only pause execution when a specific condition is met. Right-click on a breakpoint and select "Edit Breakpoint" to add a condition.

8. **Logpoints:**

   - Logpoints allow you to log messages to the console without pausing execution. Right-click in the gutter and select "Add Logpoint".

References

Official VS Code Debugging Documentation

Python Debugging in VS Code

JavaScript Debugging in VS Code

By following these steps, you can set up and start debugging a simple program in VS Code. Utilizing the key debugging features will help you efficiently identify and resolve issues in your code.

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.

Visual Studio Code (VS Code) has built-in support for Git, making it easy to manage version control within the editor. Here's how you can initialize a repository, make commits, and push changes to GitHub.

Initializing a Git Repository

1. **Open a Project Folder:**

    - Open the folder containing your project in VS Code by clicking on "File"> "Open Folder".

2. **Initialize Git:**

    - Open the Source Control view by clicking the Source Control icon in the Activity Bar on the side of the window or by using the shortcut "Ctrl+Shift+G" (Windows/Linux) or "Cmd+Shift+G" (macOS).
    - Click on "Initialize Repository" in the Source Control view.

- This will create a ".git" folder in your project directory, initializing it as a Git repository.

Making Commits

1. Stage Changes:

   - In the Source Control view, you will see a list of files that have been changed.
   - Click the "+" icon next to each file to stage it. Alternatively, you can click the "+" icon next to the "Changes" header to stage all changes at once.

2. Commit Changes:

   - Once the changes are staged, type a commit message in the input box at the top of the Source Control view.
   - Click the checkmark icon ("✓") or press "Ctrl+Enter" (Windows/Linux) or "Cmd+Enter" (macOS) to commit the staged changes.

Pushing Changes to GitHub

1. Connect to a Remote Repository:

   - If you haven't already, create a repository on GitHub.
   - Copy the repository URL (e.g., "https://github.com/username/repository.git").
   - In VS Code, open the Command Palette using "Ctrl+Shift+P" (Windows/Linux) or "Cmd+Shift+P" (macOS), then type "> Git: Add Remote" and select it.
   - Enter the remote name (commonly "origin") and paste the repository URL.

2. Push Changes:

   - Open the Command Palette and type "> Git: Push" and select it.
   - If prompted, choose the remote repository (e.g., "origin") and the branch to push to (commonly "main" or "master").

- You may be asked to enter your GitHub credentials or set up an SSH key for authentication.

Additionally:

Cloning a Repository:

- You can clone an existing repository by opening the Command Palette and typing “> Git: Clone”, then entering the repository URL.

Branch Management:

- Create and switch branches directly from the Source Control view or the Command Palette by using “> Git: Create Branch” and “> Git: Checkout to”.

Pulling Changes:

- Pull changes from the remote repository by opening the Command Palette and typing “> Git: Pull”.

View Git History:

- Use extensions like "GitLens" to enhance the Git experience in VS Code, providing more insights into your repository history and contributions.

By integrating Git with VS Code, one can streamline your version control workflow, keeping all the development activities within a single environment.

REFERENCES:

Prettier Extension

https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode

ESLint Extension

https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint

Live Server Extension

https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer

Debugger for Chrome Extension

https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome

Path Intellisense Extension

https://marketplace.visualstudio.com/items?itemName=christian-kohler.path-intellisense

HTML CSS Support Extension

https://marketplace.visualstudio.com/items?itemName=ecmel.vscode-html-css

JavaScript (ES6) Code Snippets Extension

https://marketplace.visualstudio.com/items?itemName=xabikos.JavaScriptSnippets

Visual Studio Code Marketplace: VS Code Extensions

https://marketplace.visualstudio.com/vscode

Official VS Code documentation: Managing Extensions

https://code.visualstudio.com/docs/editor/extension-marketplace

Integrated Terminal in Visual Studio Code

https://code.visualstudio.com/docs/terminal/basics

Command Palette in Visual Studio Code

https://code.visualstudio.com/docs/getstarted/userinterface#_command-palette

Official VS Code Documentation on Basic Editing

https://code.visualstudio.com/docs/editor/codebasics

Official VS Code Documentation on Opening and Managing Files

https://code.visualstudio.com/docs/editor/codebasics#_opening-and-managing-files

Official VS Code Documentation on Explorer

https://code.visualstudio.com/docs/getstarted/userinterface#_explorer

Official VS Code Documentation on Command Palette

https://code.visualstudio.com/docs/getstarted/userinterface#_command-palette

Settings

https://code.visualstudio.com/docs/getstarted/settings

Color Theme

https://code.visualstudio.com/docs/getstarted/themes

Font Size

https://code.visualstudio.com/docs/getstarted/settings#_editor

Keyboard Shortcuts

https://code.visualstudio.com/docs/getstarted/keybindings

Visual Studio Code - Version Control

https://code.visualstudio.com/docs/sourcecontrol/overview

GitHub - Connecting to GitHub with SSH

https://docs.github.com/en/authentication/connecting-to-github-with-ssh

Git - First-Time Git Setup

https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

End.

Submission Guidelines:

- Your answers should be well-structured, concise, and to the point.
- Provide screenshots or step-by-step instructions where applicable.
- Cite any references or sources you use in your answers.
- Submit your completed assignment by 1st July