Installation and navigation of virtual studio code (VS code)

1. Installation of vs code

I'll download VS Code from the official website at https://code.visualstudio.com/. I click on the download button for Windows and the installer (.exe) file starts downloading to my computer. Once the download finishes, I locate the file in my Downloads folder and double-click on it to run the installer.

Next, I follow the installation wizard prompts. I click "Next" to proceed through each step, choose the destination folder where I want VS Code installed, and opt to create desktop shortcuts for easier access.

After configuring everything, I click "Install" to begin the installation process. It usually takes a few moments to complete.

Once installed, I can launch VS Code from the Start menu or by using the desktop shortcut.

Before installing, I made sure my Windows 11 system meets the necessary prerequisites: it’s a 64-bit version of Windows 11, has sufficient disk space for installation and any additional files, and is connected to the internet for downloading the installer and initial setup. Depending on my needs, I have also had Git installed since I plan to use version control features within VS Code.

1. First-time Setup:

After installing the VS Code, the first thing I saw on the screen was the ways in which I can open new files and folders. From here I had to install all the extensions that will assist me on my adventure of coding and some of the thing I had to configure was the theme, I don’t know if it’s a norm or a way in which all the coders or developers like but I chose the dark theme to me , it was complement to my personality and to aid at the fact that I will be spending much time in front of the screen this will held not affect my sight. The dark theme will also be good as too much light can tend to push you to doze off.

From all the settings and extensions, one of which was important was all the python verified extensions, and also the prettier extension which will aid on making my coding screen look good.

1. User Interface Overview:

Visual Studio Code (VS Code) has a user-friendly interface designed to maximize productivity for developers. Here are the main components of the VS Code user interface:

1. **Activity Bar**:
   * **Purpose**: The Activity Bar is located on the side of the editor and contains icons for various activities such as exploring files, searching, source control management, debugging, and extensions. It provides quick access to different functionalities of VS Code.
2. **Side Bar**:
   * **Purpose**: The Side Bar is also located on the side of the editor and typically displays file explorer and source control views by default. It allows you to navigate and manage your project's files and folders, as well as interact with version control systems like Git.
3. **Editor Group**:
   * **Purpose**: VS Code supports multiple editors organized into groups. Each editor group can contain one or more editor tabs, allowing you to work on multiple files or different parts of the same file simultaneously. Editor groups are customizable and can be split or merged as needed.
4. **Status Bar**:
   * **Purpose**: The Status Bar is located at the bottom of the VS Code window and provides information about the current state of your workspace and editor. It includes features such as the notification area, language mode selector, indentation settings, and line feed/character encoding. It also displays Git branch information and allows quick access to settings and commands.

These components work together to provide a flexible and efficient environment for coding, debugging, and managing projects within VS Code. Each component can be tailored and extended through settings and extensions to suit individual preferences and workflows.

1. Command Palette:

The Command Palette in Visual Studio Code (VS Code) is a powerful tool that allows you to execute commands with a keyboard shortcut and search for commands that are not readily available through menus or the interface. It provides a quick way to access various features and functionalities within VS Code.

**Accessing the Command Palette:**

To access the Command Palette in VS Code, you can use the following keyboard shortcut:

* **Windows**: Ctrl + Shift + P

This shortcut opens the Command Palette at the top of the editor window, where you can start typing to filter and search for commands.

Examples of Common Tasks Using the Command Palette:

1. **Opening Files and Folders**:
   * Typing File: Open... allows you to quickly open files or folders in your workspace without navigating through the Explorer.
2. **Running Tasks**:
   * For example, Tasks: Run Task... lets you run predefined tasks from your workspace's task configuration file (tasks.json).
3. **Changing Language Modes**:
   * Typing Change Language Mode followed by the name of a programming language (e.g., Change Language Mode to JavaScript) changes the language mode of the current file.
4. **Searching and Replacing**:
   * Replace or Search commands allow you to find and replace text within the current file or across your entire workspace.
5. **Git Operations**:
   * Commands such as Git: Commit, Git: Pull, Git: Push, etc., provide shortcuts for common Git operations without switching to the Source Control view.
6. **Extensions Management**:
   * Extensions: Install Extensions opens the Extensions view where you can search for, install, and manage VS Code extensions.
7. **Settings and Preferences**:
   * Preferences: Open Settings (JSON) or Preferences: Open Settings (UI) allows you to modify VS Code settings either through the JSON file or the graphical interface.
8. **Debugging**:
   * Commands like Debug: Start Debugging, Debug: Run, Debug: Stop, etc., help you control the debugging process directly from the Command Palette.
9. **Terminal Operations**:
   * Terminal: New Terminal opens a new integrated terminal window within VS Code.
10. **Window Management**:
    * Commands such as View: Toggle Full Screen, View: Toggle Sidebar, View: Split Editor, etc., help you manage the layout and appearance of the VS Code window.

The Command Palette in VS Code is extremely resourceful and customizable. It's particularly useful for power users who prefer to navigate and control their development environment primarily through keyboard shortcuts and quick commands.

1. Extensions in VS Code:

**Python** by Microsoft:

* This official extension provides rich support for Python including IntelliSense (code completion), debugging, linting, code formatting, and more. It's essential for basic Python development in VS Code.

**Pylance**:

* Developed by Microsoft in collaboration with the Python community, Pylance offers fast and feature-rich Python language support with advanced type checking (powered by Pyright), IntelliSense, and more.

**Jupyter**:

* The Jupyter extension allows you to work with Jupyter Notebooks directly within VS Code, providing interactive computing and data visualization capabilities for Python.

**Python Docstring Generator**:

* Simplifies the process of writing docstrings for Python functions, methods, and classes by generating docstring templates based on function signatures.

**Django**:

* If you're working with Django, this extension provides support for Django projects, including syntax highlighting, IntelliSense, and snippets tailored for Django templates and Python code.

**Coverage Gutters**:

* Integrates with coverage.py to display code coverage information directly in the editor's gutter (next to line numbers), helping you visualize which lines of code are covered by your tests.

**autoDocstring**:

* Automatically generates docstrings for Python functions based on function arguments and return values, saving time and ensuring consistent documentation.

**Python Test Explorer for Visual Studio Code**:

* This extension allows you to discover and run your Python tests (e.g., pytest, unittest, nose) directly from within VS Code, providing a graphical test explorer interface.

**Python Extension Pack**:

* A curated collection of popular Python extensions bundled together, including the Python extension by Microsoft, Jupyter, Django, and other useful tools for Python development.

**Installing Python Extensions in VS Code:**

To install these extensions (and others) for Python development in VS Code:

1. Open VS Code.
2. Go to the Extensions view by clicking on the Extensions icon in the Activity Bar (or use Ctrl + Shift + X).
3. Search for the extension by name (e.g., "Python", "Pylance", "Jupyter").
4. Click on the extension you want to install and then click Install.

**Managing Python Extensions:**

Once installed, you can manage Python extensions in VS Code:

* Enable/Disable: Toggle extensions on or off based on your current needs.
* Update: Keep your extensions up to date to benefit from new features and bug fixes.
* Uninstall: Remove extensions that you no longer use or need.

1. Integrated Terminal:

To open a terminal on vs code, you can use two ways which both serve the purpose. The first way is to use the keyboard shortcut, thus “Ctrl + `” control plus backtick, (backtick is the key that is located below the “esc” key on the keyboard. The second option is to use the menu on the VS code. Using the menu can be tricky depending on how you have opened your VS code. If you have opened your vs code to a full screen view, the menu options are located on the top left corner of the screen. If the VS code is opened as half or not fully showing everything, the terminal is found on the three dots. From there you will find the option to open the terminal.

Advantages of using integrated terminal over external terminal

1. Seamless Integration: The integrated terminal is directly embedded within the editor interface, allowing you to work without switching between different applications. This integration promotes a smoother workflow and saves time by reducing context switching.
2. Context Awareness: The integrated terminal is aware of the current project or workspace settings within the editor. It automatically opens at the root directory of your project, making it easier to navigate and execute commands relevant to your codebase.
3. Customization: Editors like VS Code allow customization of the integrated terminal with different shell environments, fonts, colors, and other settings. This flexibility helps tailor the terminal to your preferences and workflow requirements.
4. Direct Access to Editor Features: You can interact with editor features directly from the terminal. For instance, you can run build scripts, execute debugging commands, or even manage version control operations seamlessly within the same interface.
5. Efficient Debugging: Debugging tools and commands can be integrated with the editor's terminal, providing a unified debugging experience. This can include running debugging sessions, inspecting variables, and handling breakpoints directly from the terminal window.
6. Resource Efficiency: Since the integrated terminal shares resources with the editor, it typically consumes fewer system resources compared to running a separate terminal application concurrently with the editor.
7. File and Folder Management

**Creating Files and Folders:**

1. **Creating a New File**:

* To create a new file, you can either use the file explorer sidebar or the command palette (Ctrl + Shift + P) and type "New File". Alternatively, you can use the keyboard shortcut Ctrl + N.
* Enter the desired file name and press Enter.

1. **Creating a New Folder**:

* To create a new folder, right-click in the file explorer sidebar or use the command palette and type "New Folder". Alternatively, use the keyboard shortcut Ctrl + Shift + N.
* Enter the folder name and press Enter.

**Opening Files and Folders:**

1. **Opening Files**:
   * You can open files by double-clicking on them in the file explorer sidebar.
   * Alternatively, use the command palette (Ctrl + P) and start typing the file name. VS Code will suggest matching files as you type.
2. **Opening Folders**:
   * Open a folder by selecting File > Open Folder... from the menu, or by using the Ctrl + K followed by Ctrl + O shortcut.
   * You can also drag and drop a folder into the VS Code window to open it.

**Managing Files and Folders:**

1. **Renaming and Deleting**:
   * To rename a file or folder, right-click on it in the file explorer sidebar and select "Rename", or press F2 while it's selected.
   * To delete a file or folder, right-click on it and select "Delete", or press Delete or Backspace while it's selected.
2. **Moving and Copying**:
   * You can move files or folders by dragging and dropping them within the file explorer sidebar to their new location.
   * To copy, you can duplicate a file and then rename it.

**Navigating Between Files and Directories Efficiently:**

**Switching Between Open Files**:

* Use Ctrl + Tab to cycle through recently opened files.
* Use Ctrl + P to open the file switcher and start typing the file name to quickly jump to a specific file.

**Navigating Directory Structure**:

* Use the file explorer sidebar to navigate through directories. You can expand and collapse folders by clicking on them or using the arrow keys.
* Use breadcrumbs at the top of the editor to quickly switch between parent directories or open files.

**Command Palette**:

* Utilize the command palette (Ctrl + Shift + P ) to access various file and folder management commands quickly. For example, you can open files, create new files/folders, and perform various navigation tasks.

1. Settings and Preferences:

In Visual Studio Code (VS Code), users can access and customize settings through several avenues to tailor their coding environment. Settings are found in the "Settings" tab accessible via File > Preferences > Settings, or by pressing Ctrl + , (Cmd + , on macOS). Here, users can search for specific settings or browse categories such as "Workbench", "Editor", and "Extensions" to modify various aspects of the editor's behaviour and appearance. For instance, to change the theme, users can type "colour theme" in the search bar, select a desired theme from the dropdown list, and apply it. Adjusting font size involves searching for "editor font size" and setting the preferred size in pixels. Customizing keybindings involves navigating to "Keyboard Shortcuts" and clicking on the "Edit in settings.json" link to customize keybindings using JSON syntax or directly remapping commands. These customization options empower users to personalize their coding experience in VS Code according to their preferences and workflow needs.

1. Debugging in VS Code:

Install Extensions:

* Ensure that all the necessary extensions are installed for the programming language that ill be using to code, which could be python, java and/or HTML

Open Your Project:

* Open project folder in VS Code.

Create or Open Your Program:

* Create a new file or open an existing file with your program code.

Set Breakpoints:

* Click in the gutter next to the line number where you want to pause execution.

Start Debugging:

* Press F5 or click Run > Start Debugging from the menu.

Select Debugger Configuration:

* Choosing the appropriate configuration if prompted for example python or JS

Debugging Controls:

* Use the debug toolbar (play, pause, step over, step into) to navigate through your code.

Inspect Variables:

* Hover over variables in the editor or check the "Variables" pane to inspect their values.

Continue or Stop Debugging:

* Click Continue or Shift + F5 to stop debugging when done.

Review Output:

* Check the "Debug Console" for additional output or messages

The following features aid with the debugging, and they are offered by the VS code :

**Breakpoints**: Developers can set breakpoints in their code to pause execution at specific lines or conditions, allowing them to inspect the program state at that point.

**Variable Inspection**: While debugging, developers can hover over variables in the editor to see their current values or view them in the "Variables" pane, enabling them to track how values change during execution.

**Call Stack**: The call stack provides a visual representation of the current execution path, showing the sequence of function calls that led to the current point in the program. This helps developers understand the flow of their code.

**Watch Expressions**: Developers can add expressions to the "Watch" pane to monitor specific variables or expressions continuously. VS Code evaluates these expressions as the program executes, providing real-time updates on their values.

**Debug Console**: The debug console allows developers to interact with their program during debugging sessions. They can execute commands, print values, and debug log messages directly in the console to gain insights into program behavior.

**Step Controls**: VS Code offers various step controls (e.g., step over, step into, step out) in the debug toolbar. These controls enable developers to navigate through their code line-by-line or function-by-function, helping them trace and understand the execution flow.

**Conditional Breakpoints**: Developers can set breakpoints with conditions based on expressions or variables. This feature allows breakpoints to trigger only when specific conditions are met, enhancing flexibility in debugging complex scenarios.

**Debugger Configuration**: VS Code supports configurations for different programming languages and environments. Developers can customize debugger settings, such as launch configurations and environment variables, to match their specific debugging needs.

**Exception Handling**: VS Code provides options to configure how exceptions are handled during debugging. Developers can choose to break execution when exceptions occur, ignore specific types of exceptions, or handle them based on predefined conditions.

**Debugging Task Automation**: VS Code integrates debugging with tasks and scripts, allowing developers to automate common debugging workflows. They can configure tasks to launch specific debugging configurations or execute pre-debugging actions automatically.

1. Using Source Control:

Integrating Git and VS code

The integration of Git with Visual Studio Code (VS Code) enhances the development experience by seamlessly incorporating powerful version control capabilities into the editor environment. Users can initialize Git repositories directly from within VS Code, manage branches, track changes, and collaborate efficiently with teammates. The Source Control view provides a comprehensive interface for staging changes, viewing commit history, and handling branches effortlessly. Committing changes is straightforward, with the ability to add commit messages directly within VS Code. Pushing and pulling changes to and from remote repositories like GitHub is streamlined, ensuring synchronization and collaboration across distributed teams. Visual indicators and extensions further enrich the Git experience, offering tools for code analysis, history visualization, and customization. This integrated approach not only simplifies version control tasks but also promotes productivity and code quality by enabling developers to focus more on coding and less on managing their project's source code.

STEPS OF INTERGRATING GIT WITH VS code:

**Initializing a Repository**:

* Open VS Code and navigate to your project folder.
* Open Command Palette (Ctrl + Shift + P) and select "Git: Initialize Repository".
* Stage files by clicking + next to each file in the Source Control view.

**Making Commits**:

* Enter commit message and press Ctrl + Enter to commit changes.
* View commit history by clicking on the clock icon in the Source Control view.

**Pushing Changes to GitHub**:

* Link your local repository to GitHub using your personal access token.
* Click ... next to "Changes", choose "Push", select branch, and confirm to push changes.
* Authenticate with GitHub credentials if prompted.