CMPSC 104 – Document Engineering

Prof. Hang Zhao

Visual Studio Code (VSCode)

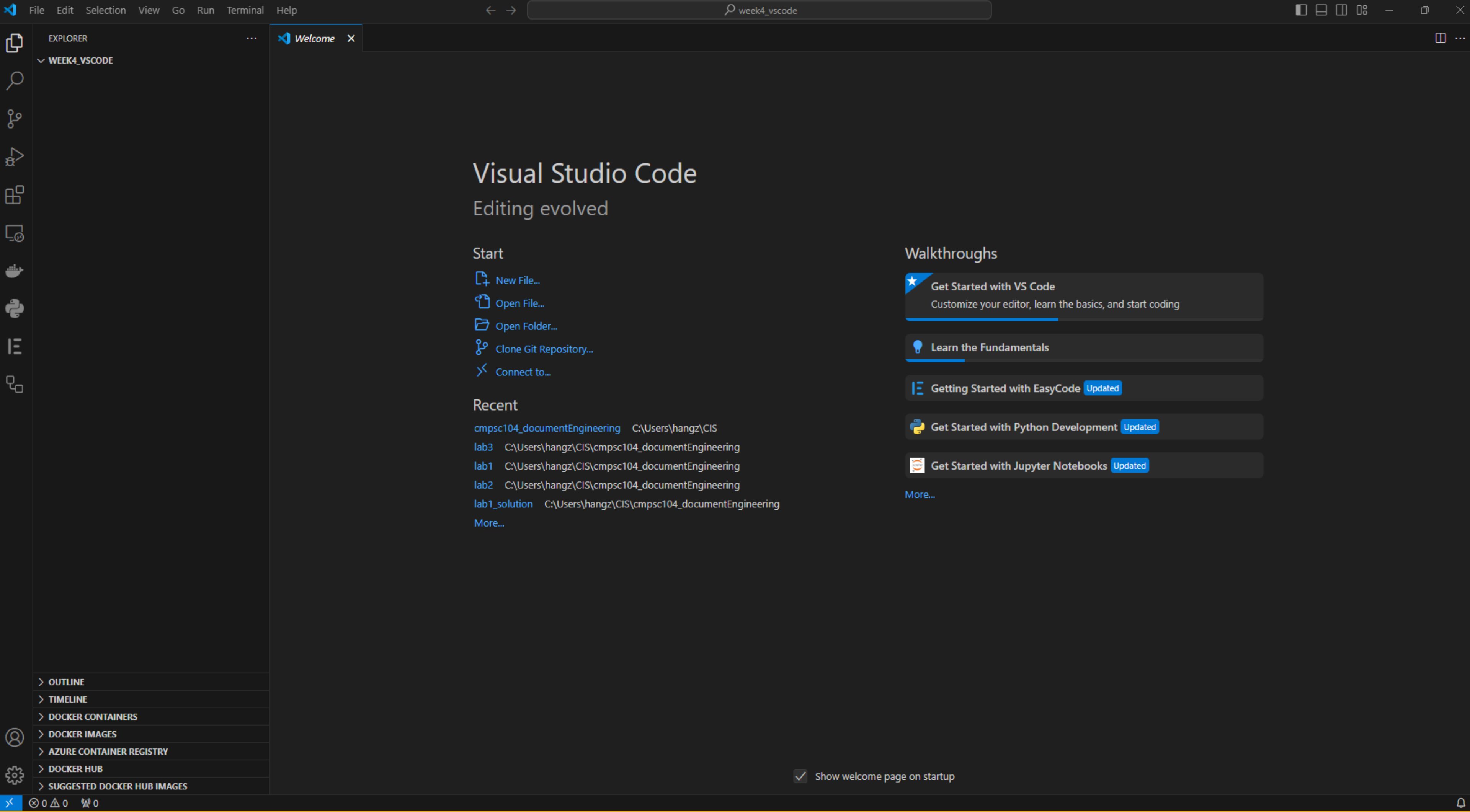
A lightweight but powerful source code editor, a free code editor

- Cross-platform compatibility: available on Windows, macOS, and Linux.
- Extensive language support: JavaScript, Python, HTML, C++, C#, Java
- Rich extension ecosystem
- Integrated Git support: Directly manage your Git repositories within the editor.
- Debugging tools
- Built-in terminal

Setting up Visual Studio Code

- Download and install VS Code: https://code.visualstudio.com/download

- Verify VSCode Installation:
- Execute `code --version` in Command Prompt (CMD)/Terminal to confirm your VSCode installation.


Visual Studio Code

Editing evolved

Start

New File...

Open File...

Open Folder...

Clone Git Repository...

Connect to...

Recent

cmpsc104_documentEngineering	C:\Users\hangz\CIS
lab3	C:\Users\hangz\CIS\cmpsc104_documentEngineering
lab1	C:\Users\hangz\CIS\cmpsc104_documentEngineering
lab2	C:\Users\hangz\CIS\cmpsc104_documentEngineering
lab1_solution	C:\Users\hangz\CIS\cmpsc104_documentEngineering

More...

Walkthroughs

Get Started with VS Code

Customize your editor, learn the basics, and start coding

Learn the Fundamentals

Getting Started with EasyCode Updated

Get Started with Python Development Updated

Get Started with Jupyter Notebooks Updated

More...

Show welcome page on startup

Introduction to Visual Studio Code

- Open a folder: File > Open Folder (cmd + o (mac), ctrl + o (win))
- Use File Explorer to view the folder's files and subfolders: View > Explorer (Ctrl+Shift+E)
- Source Control: View > Source Control (SCM) (Ctrl+Shift+G)
- Run and Debug: View > Run (Ctrl+Shift+D)
- Extensions view: View > Extensions (Ctrl+Shift+X)
- **Open the Command Palette: View > Command Palette... (Ctrl+Shift+P)**
- Integrated Terminal: View > Terminal (Ctrl+`)
- Create a new file: File > New File (Ctrl+N)
- Auto Save: File > Auto Save
- Run: Run > Start Debugging (F5)
- Zoom: Zoom out (Ctrl+-), Zoom in (Ctrl+=)

Work with .html & .md in VSCode

- Add index.html and README.md to the Explorer.
- Install Live Preview extension.

Work with Python in VSCode

- Create a new file (cmd + n (mac), ctrl + n (win)): **python_intro.py**
- Install Python extension.
- Write some python codes:

```
print("-----------------------------")
print("Hello World!")
print("This is a sample python program")
print("-----------------------------")
```

Edit and Run Code

- Install the Node.js runtime to execute JavaScript code: https://nodejs.org

  https://nodejs.org/en/download/prebuilt-installer (Windows)
  https://nodejs.org/en/download/prebuilt-installer (MacOS)
  `sudo snap install node –classic` (Linux)

- Check your Node.js installation: **node --version**

Work with JavaScript in VSCode

Create a simple "Hello world" console application called **app.js**:

```
var msg = 'Hello World';
console.log(msg);
```

Automatically format the source code: Shift+Alt+F

Turn on Auto Save: File > Auto Save

Display the Integrated Terminal: View > Terminal (Ctrl+`)

Create new terminal: Create New Terminal (Ctrl+Shift+`)

Run the application: **`node app.js`**

Productivity Tips

- View > Command Palette… (Ctrl+Shift+P)
- Ctrl+P to show the Quick Open dropdown

Multi-cursor selection:

- To add a new cursor: **Alt+Click** on Windows and Linux, **Option+Click** on macOS
- To add a new cursor above or below the current position: **Ctrl+Alt+Up, Ctrl+Alt+Down**
- To add cursors to all matches of the current selection: **Ctrl+Shift+L**

Debugging

- Run a sample **python_debug.py** app.

```
a = 33
b = 300
if b > a:
    print("b is greater than a")

c = b / 0
```

- Add a breakpoint at the line if b > a:.

Version Control

- Source Control: View > Source Control (SCM) (Ctrl+Shift+G)
- Initialize repository: **main** is the default branch
- Open the Command Palette: View > Command Palette (Ctrl+Shift+P)
- File version control status: U - Untracked file, A - Added file, M - Modified file

Linting in Visual Studio Code

Linting: highlights syntactical and stylistic problems in your code.

Linter:
- Tools or software programs used to analyze code.
- They perform the linting process by scanning your code.
- Linters are usually specific to a programming language. For example, Pylint is a linter for Python, and ESLint is commonly used for JavaScript.

Pylint: a static code analysis tool for the Python programming language according to PEP8.

Running Linters with GitHub Actions

GitHub Actions is a CI/CD platform that runs your linter every time you push code to your repository. It ensures your codebase is clean, and catches errors before they make it into production.

Set up a linter with GitHub Actions:

Create a workflow file (.yml) in your repository under the .github/workflows/ directory. You can name it py_lint.yml.

Configure the workflow:
1. set up the environment
2. install dependencies
3. run the linter.

Running Linters with GitHub Actions

Add the following content to the python-lint.yml file:

```yaml
name: Python Linting

on: push

jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: '3'
    - name: Install Pylint
      run: pip install pylint
    - name: Run Pylint
      run: pylint src/sample.py
    # Run GatorGrader: see config/gatorgrade.yml
    - name: Run GatorGrader with GatorGrade
      if: always()
      run: |
        pip install gatorgrade
        gatorgrade --config config/gatorgrade.yml
```

Running Linters with GitHub Actions

Updated sample.py file:

```python
""“The project calculates Mean."""

def calculate_mean(numbers):
    """Calculate and return the mean of a list of numbers.

    Args:
        numbers (list): A list of numbers (ints or floats).

    Returns:
        float: The mean of the numbers.
    """
    total = sum(numbers)
    count = len(numbers)
    mean = total / count
    return mean

if __name__ == "__main__":
    nums = [1, 2, 3, 4, 5]
    print("The mean is", calculate_mean(nums))
```