CMPSC 104 – Document Engineering

Prof. Hang Zhao

ALLEGHENY COLLEGE

Docunment Engineering

Overview

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.

File Edit Selection View Go Run Terminal Help

week4_vscode

EXPLORER

WEEK4_VSCODE

Welcome

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

OUTLINE

TIMELINE

DOCKER CONTAINERS

DOCKER IMAGES

AZURE CONTAINER REGISTRY

DOCKER HUB

SUGGESTED DOCKER HUB IMAGES

0 0 0

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)
- Save a file: File > Save (Ctrl+S)
- Auto Save: File > Auto Save
- Run: Run > Start Debugging (F5)
- Zoom: Zoom out (Ctrl+-), Zoom in (Ctrl+=)

Edit and Run Code

- Install the Node.js runtime to execute JavaScript code: Find Node.js for your platform at https://nodejs.org
- Check your Node.js installation: type node –version in a terminal or command prompt.
- Create new file: File > New File (Ctrl+N)
- Create a simple "Hello world" console application called app.js: IntelliSense provides suggestions as you type.
- Automatically format the source code.
- Format Document command (Shift+Alt+F)
- Turn on Auto Save: File > Auto Save
- Display the Integrated Terminal: View > Terminal (Ctrl+`)
- Split the terminal: Split Terminal (Ctrl+Shift+5)
- Create new terminal: Create New Terminal (Ctrl+Shift+`)
- Run the application: From the Integrated Terminal, type node app.js

Productivity Tips

- View > Command Palette... (Ctrl+Shift+P)
- Quick Open recent files or search by filename
- Ctrl+P to show the Quick Open dropdown
- Go to Line in a file: (type filename: line number)
- type filename@symbol name

Multi-cursor selection:

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

Personalize Visual Studio Code

- Change your Color Theme.
- Install a new Color Theme from the VS Code Extension Marketplace.
- Change your File Icon Theme.

Extensions

- Find extensions to install using the Extensions view.
- Install an extension from the VS Code Extension Marketplace.
- See what features are added via the Features Contributions tab or Command Palette (Ctrl+Shift+P).
- See recommendations for other extensions.

Debugging

- Run a sample example.py app.
- Use a launch.json configuration file.
- Single file debugging.
- Set a breakpoint.

Version Control

- Install Git https://git-scm.com
- 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)
- Rename a branch: Git: Rename Branch
- File version control status: U - Untracked file, A - Added file, M - Modified file
- Commit file: Commit ☑️ (check mark) button
- Create a branch: Git: Create Branch
- Diff editor: Inline View button
- Stage changes: Stage Changes ➕ button
- Switch branches: Status bar branch item (lower left)
- Merge branch: Views and More Actions (...) > Branch > Merge Branch
- Publish branch to GitHub
- Clone repository: Git: Clone > Clone from URL

Linting in Visual Studio Code

Linting: highlights syntactical and stylistic problems in your code.

Linter:

- The actual 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/pylint_sol.py
    # Run GatorGrader: see config/gatorgrade.yml
    - name: Run GatorGrader with GatorGrade
      if: always()
      run: |
        pip install gatorgrade
        gatorgrade --config config/gatorgrade.yml
```