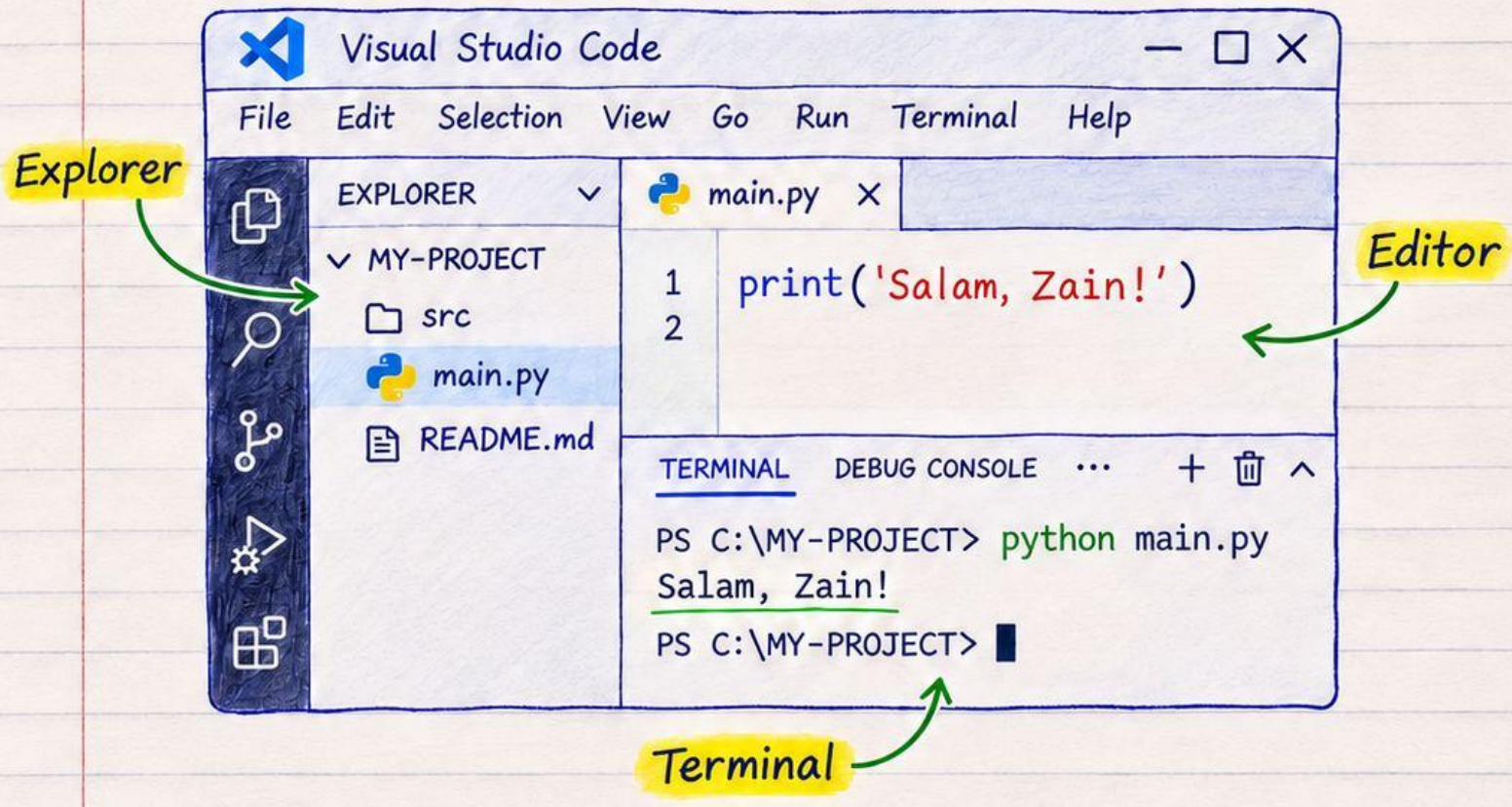
VS CODE ESSENTIALS

Windows edition • 12-page visual handbook

Write • Run • Debug • Track



Smart shortcuts

Zoom + Word Wrap

Terminal + Debugging

Git basics

Learn the tools you use every day.

YOUR PROJECT & WORKSPACE

1 • OPEN THE RIGHT FOLDER

- File > Open Folder opens the whole project.
- Example folder: zain-project, containing , app.py, README.md and assets/.
- Opening one file alone hides useful project context.



2 • FIND YOUR WAY

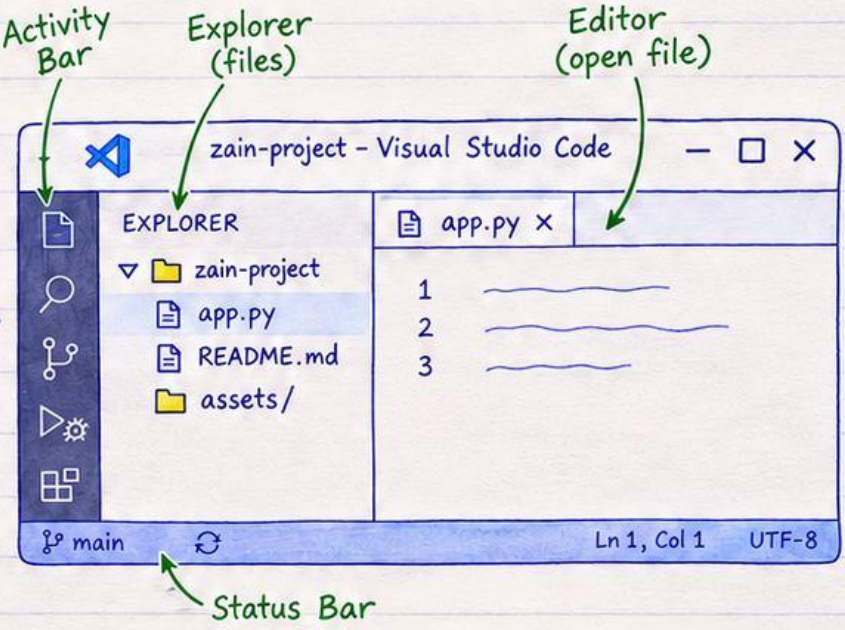
Explorer = files.

Search = project text.

Source Control = Git changes.

Run and Debug = debugging.

Extensions = add features.



3 • DAILY SHORTCUTS

Ctrl + Shift + P	: Command Palette
Ctrl + P	: find a file
Ctrl + Shift + E	: Explorer
Ctrl + B	: sidebar
Ctrl + J	: bottom panel

4 • FILES & WORKSPACES

Ctrl + N	: new file
Ctrl + S	: save
Ctrl + Shift + S	: Save As
Ctrl + W	: close editor

- Double-click a tab to keep it open.
- A workspace can contain one or several folders.
- File > Save Workspace As saves a multi-folder setup.

5 • TRUST CHECK

- Trust a folder only when you trust its source.
- Restricted Mode limits code-running features.

⚠ Key: Win shortcuts; customized bindings may differ.

EDIT & MOVE FASTER

1 • EVERYDAY EDITING

- Ctrl + C / X / V: copy / cut / paste
- Ctrl + Z / Y: undo / redo
- Ctrl + A: select all; Ctrl + S: save
- Ctrl + /: toggle line comment
- Shift + Alt + A: toggle block comment where supported

2 • LINE TOOLS

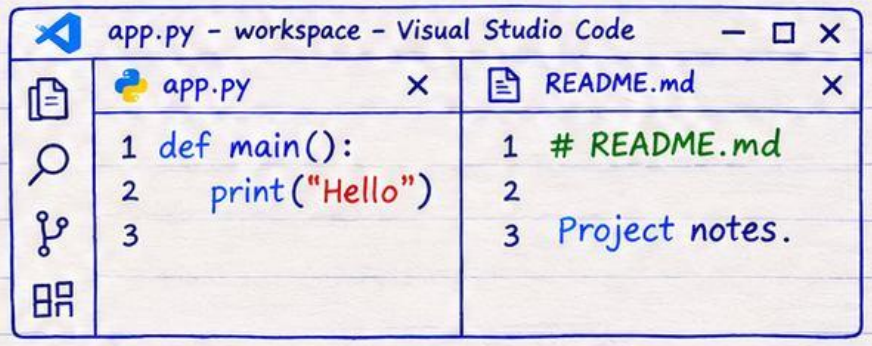
- Alt + Up / Down: move line
- Shift + Alt + Up / Down: copy line
- Ctrl + Shift + K: delete line
- Ctrl + Enter: insert line below
- Ctrl + Shift + Enter: insert line above

3 • NAVIGATION

- Ctrl + G: go to line
- Ctrl + Home / End: start / end of file
- Ctrl + Left / Right: move one word
- Hold Shift with movement keys to select
- Ctrl + Tab: switch recent editors

4 • SIDE BY SIDE

- Ctrl + Backslash: split editor.
- Ctrl + 1/2: focus editor group.



5 • CUSTOMIZE KEYS

- Ctrl + K, then Ctrl + S opens Keyboard Shortcuts.

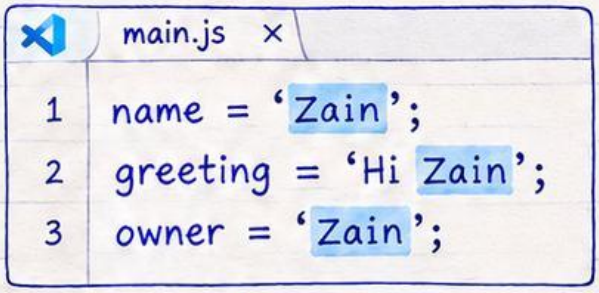
Ctrl + K , Ctrl + S

↳ A comma means press the two shortcuts in sequence.

SMART EDITING & MULTI-CURSOR

1 • EDIT MULTIPLE PLACES

- `Alt + Click` : add a cursor
- `Ctrl + D` : select next matching occurrence
- `Ctrl + Shift + L` : select all occurrences of selection
- `Esc` : return to one cursor

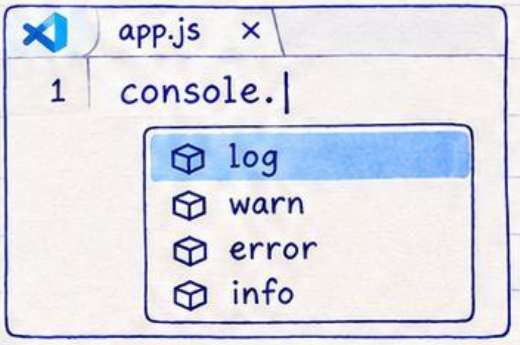


Example

⚠ Use multi-cursor carefully; it edits every selection.

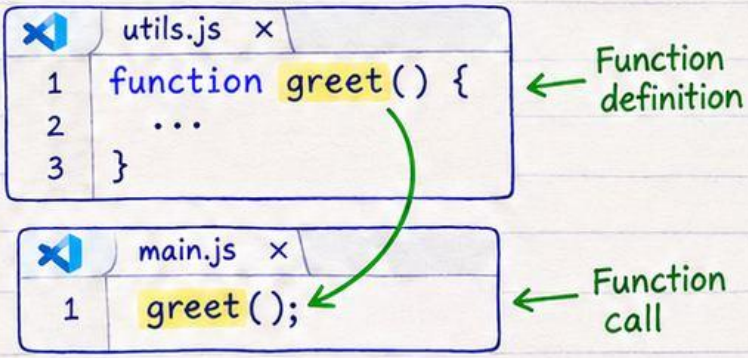
2 • CODE HELP

- `Ctrl + Space` : suggestions
- `Ctrl + Shift + Space` : parameter hints
- Hover over a symbol : documentation / type info
- `Ctrl + .` : Quick Fix or available code actions



3 • UNDERSTAND CODE

- `F12` : go to definition
- `Alt + F12` : peek definition
- `Shift + F12` : find references
- `F2` : rename a symbol across supported code



4 • RENAME SAFELY

- Use `F2` for a code symbol; plain text replacement can change unrelated text. Preview changes when available.

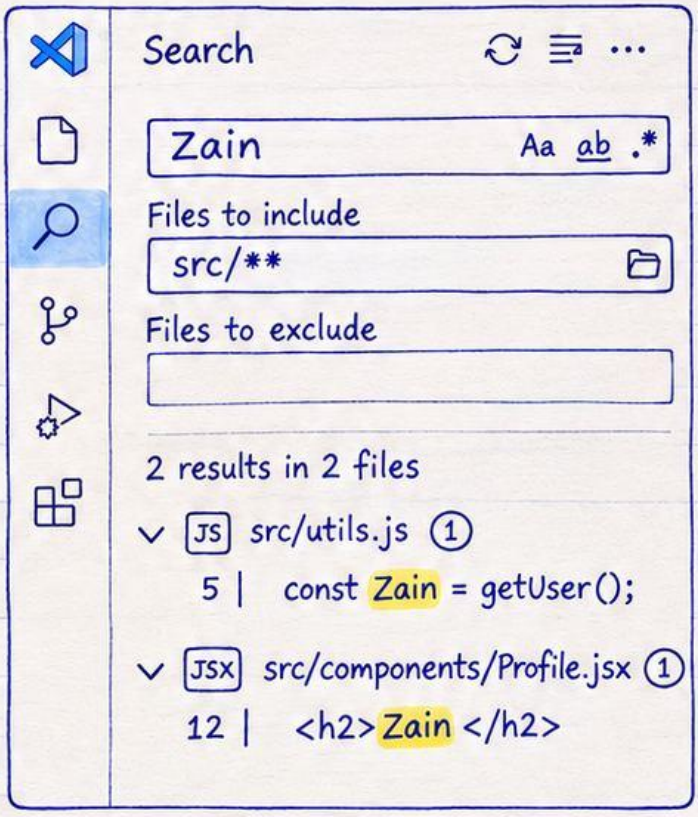
5 • LANGUAGE SUPPORT

- Suggestions and refactoring depend on the language extension and project setup. If missing, check language mode at the bottom-right.

SEARCH & REPLACE SAFELY

1 • CURRENT FILE

- Ctrl + F: find
- Ctrl + H: replace
- Enter / Shift + Enter: next / previous match



2 • PROJECT SEARCH

- Ctrl + Shift + F: find across files
- Ctrl + Shift + H: replace across files
- Click a result to see its location and context.

3 • FILTER RESULTS

- Files to include: src/**
- Files to exclude: **/node_modules/**
- Match Case: distinguish Zain from zain
- Match Whole Word: find cat without matching category
- Regular Expression: enable patterns only when needed

4 • SAFE REPLACEMENT FLOW

1. Search first.
2. Check scope and matches.
3. Replace one to verify.
4. Replace all only after review.
5. Inspect changed files.

→ Follow this flow to avoid unwanted changes.

5 • COMMON TRAPS

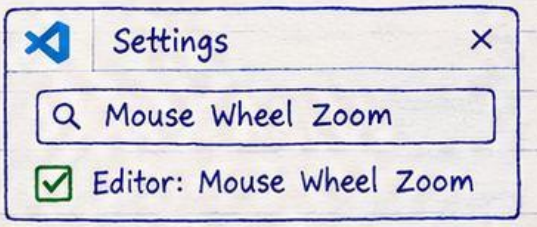
⚠ Hidden/ignored files may be excluded by search settings.
⚠ Regex punctuation has special meanings.
⚠ Use F2 for symbol renaming when supported.

Useful: Ctrl + P searches file names; Ctrl + Shift + F searches file contents.

MOUSE ZOOM & WORD WRAP

1 • ENABLE MOUSE-WHEEL ZOOM

- Open Settings with `Ctrl + comma.`
- Search `Mouse Wheel Zoom`.
- Enable `Editor: Mouse Wheel Zoom`.

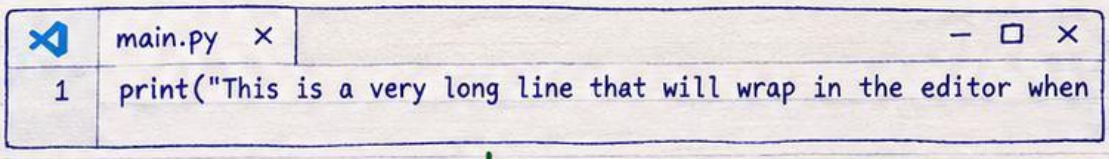


- Now hold `Ctrl` and scroll the mouse wheel over the editor to enlarge or shrink code text. Normal wheel movement scrolls.

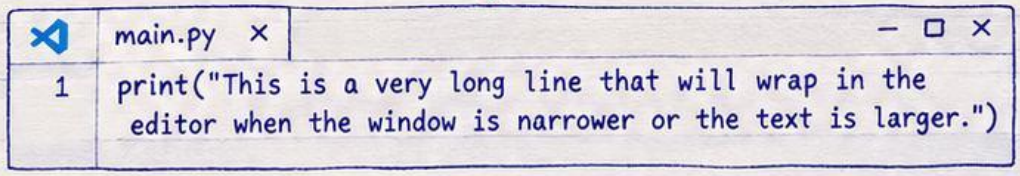
2 • KEEP LONG LINES ON SCREEN

- `Alt + Z` toggles Word Wrap for this session.
- For a lasting setting, search `Word Wrap` and set `Editor: Word Wrap` to `on.`.
- As the editor becomes narrower or the text becomes larger, long lines wrap to fit. No real newline is inserted into your file.

Before:



After:



visual wrap, not a new code line

3 • THREE DIFFERENT SIZES

- Editor font: code text only; set `Editor: Font Size`.
- Window zoom: entire VS Code interface; `Ctrl + plus / minus.` Reset via `View > Appearance > Reset Zoom`.
- Terminal font: set `Terminal > Integrated: Font Size` separately.

COPYABLE SETTINGS JSON

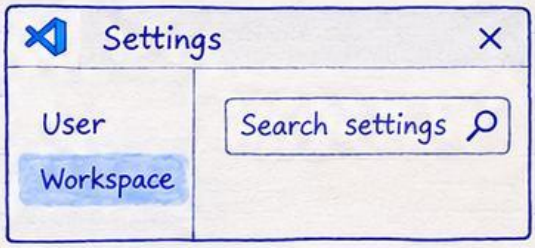
```
{
    "editor.mouseWheelZoom": true,
    "editor.wordWrap": "on",
    "editor.fontSize": 16
}
```

Remember:
Word Wrap changes display; formatting changes code layout.

SAVE & PERSONALIZE

1 • USER vs WORKSPACE

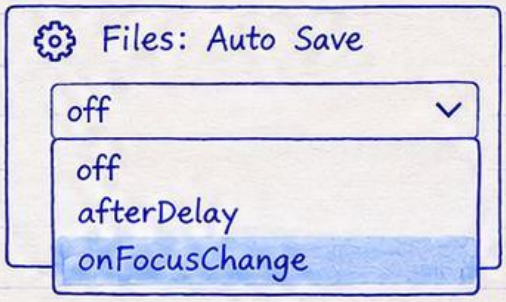
- User settings apply generally.
- Workspace settings apply to this project and normally override matching User settings.
- A single-folder project's settings live in `.vscode/settings.json`
- Open `Ctrl + comma` and choose User or Workspace.
- Language-specific settings can override general settings.



2 • AUTO SAVE

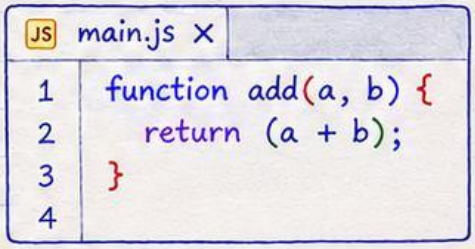
- File > Auto Save toggles automatic saving.
- In Settings search `Files: Auto Save`.
- `afterDelay` saves after a short pause; `onFocusChange` saves when you leave the editor.

⚠ Auto Save is not a backup or a Git commit.



3 • READABLE EDITOR

- Search these settings:
  - `Editor: Minimap Enabled` — hide for more space
  - `Editor: Sticky Scroll Enabled` — keep scope context visible
  - `Editor: Bracket Pair Colorization Enabled` — match brackets by color
  - `Editor: Render Whitespace` — selection helps spot spaces
- `Ctrl + K`, then `Ctrl + T`: choose a color theme.



4 • SMALL STARTER JSON

```
{
  "files.autoSave": "onFocusChange",
  "editor.minimap.enabled": false,
  "editor.bracketPairColorization.enabled": true
}
```

- Use Command Palette > `Preferences: Open User Settings (JSON)`.
- Merge into existing settings; do not paste a second outer object.

5 • INDENTATION

- Click `Spaces / Tab Size` in the status bar.
- Follow the project's style; avoid mixing tabs and spaces.

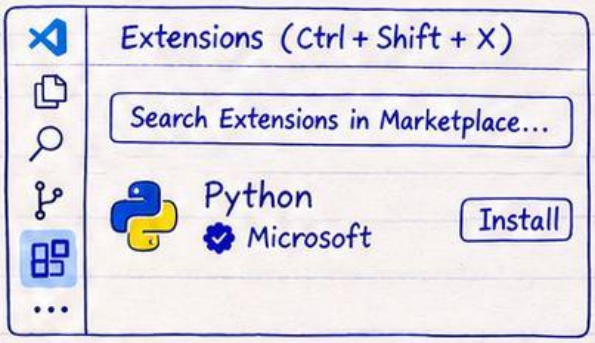


EXTENSIONS & LANGUAGE SETUP

1 • INSTALL ONLY WHAT YOU NEED

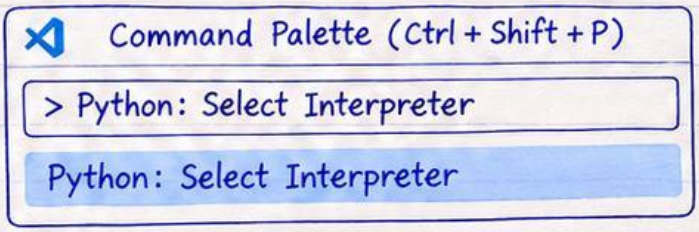
- `Ctrl + Shift + X` opens Extensions.
- Search, check publisher and purpose, then Install.
- Disable or uninstall unused extensions.

⚠ Extensions can execute code: use trusted publishers.



2 • PYTHON SETUP

1. Install Python separately.
2. Install Python by Microsoft.
3. Use Python: Select Interpreter in Command Palette.
4. Choose the project's environment.



→ Python Debugger supports debugging; Pylance provides rich language help. Check that the needed extensions are installed/enabled.

3 • FORMATTING CHOICES

- Python: Black Formatter by Microsoft.
- HTML / CSS / JavaScript: Prettier by Prettier.

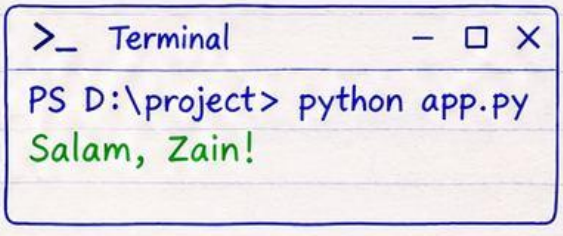
→ Choose one default formatter per language to avoid conflicts.

4 • WHAT AN EXTENSION IS NOT

- VS Code is an editor. A language extension does not automatically replace Python, Node.js, Git, or other required tools.

5 • CHECK THE SETUP

- In a new terminal: `python --version`
- Create `app.py` with `print('Salam, Zain!')`.
- Run: `python app.py`
- Expected output: Salam, Zain!



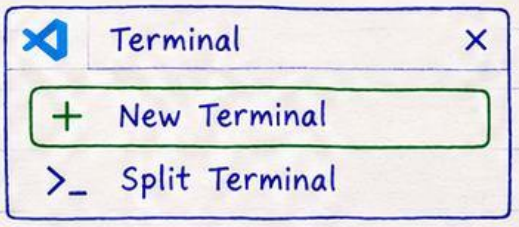
6 • IF HELP IS MISSING

- Check the file extension, selected language mode, interpreter, extension status and Workspace Trust.

TERMINAL & RUNNING CODE

1 • OPEN THE TERMINAL

- `Ctrl + backtick` toggles the integrated terminal. Backtick is the ` key on many keyboards.
- Use `Terminal > New Terminal` if your keyboard differs.
- The terminal is a real shell such as PowerShell; it is separate from the code editor.



2 • BASIC POWERSHELL COMMANDS

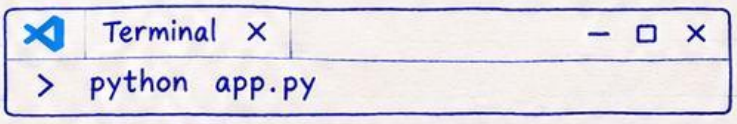
- `pwd` — show current folder
- `ls` — list files
- `cd folder` — enter a folder
- `cd ..` — move up one level
- `clear` — clear the display

Quote paths with spaces:

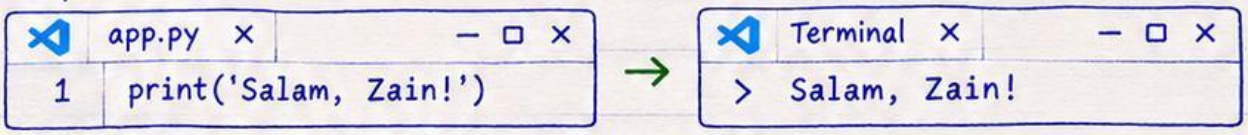
```
cd "my project"
```

3 • RUN PYTHON

- Save `app.py` first. In its folder run:



- Example code:



- For JavaScript, `node app.js` requires Node.js.

4 • CONTROL THE SESSION

- `Ctrl + C` stops a running command when no text is selected.
- `Plus` creates a terminal.
- `Trash` ends that terminal.
- Hiding the panel does not stop its process.
- Choose shell: `Command Palette > Terminal: Select Default Profile`; then open a new terminal.

5 • TROUBLESHOOT

- File not found? Check folder and file name.
- Command not found? Check installation / PATH, then reopen the terminal.
- Do not paste commands you do not understand.

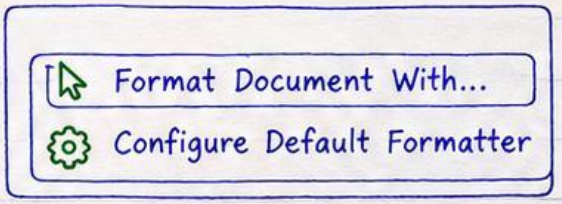
Remember:

Do not paste commands you do not understand.

FORMAT & FIX PROBLEMS

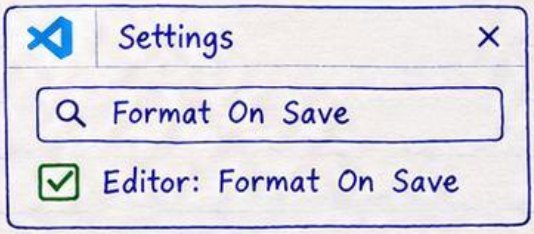
1 • FORMAT DOCUMENT

- `Shift + Alt + F` formats the current file using a supported formatter.
- Right-click > `Format Document With...` > `Configure Default Formatter` chooses one.



2 • FORMAT ON SAVE

- Enable `Editor: Format On Save`.
- Save manually with `Ctrl + S` to check it works.
- Formatting arranges code; it does not prove the code is correct.

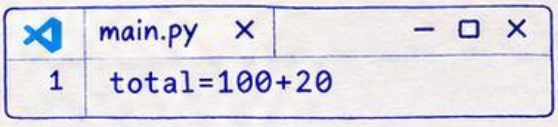


3 • PYTHON EXAMPLE SETTINGS

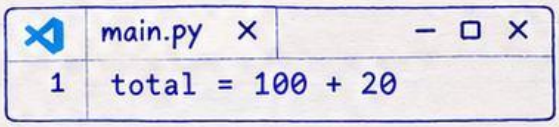
- Install `Black Formatter` first.
- In settings JSON:

```
{
  "[python]": {
    "editor.defaultFormatter":
      "ms-python.black-formatter",
    "editor.formatOnSave": true
  }
}
```

Before:

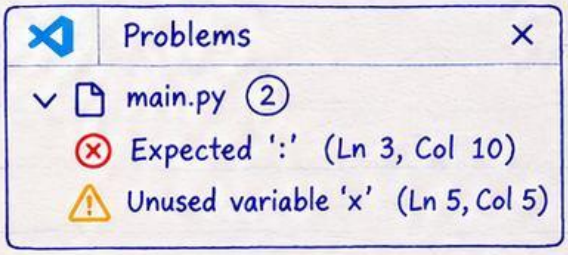


After:



4 • ERRORS & WARNINGS

- `Ctrl + Shift + M` opens Problems.
- `F8` / `Shift + F8` moves to next / previous problem.
- Hover over a squiggle to read the message.
- `Ctrl + .` shows available fixes; understand them before applying.



IF FORMATTING FAILS

- Save with the correct file extension.
- Select the intended formatter.
- Fix syntax errors.
- Check `Output` for formatter messages.

Remember:

Formatting, linting and testing are different:

- layout,
- suspicious code, and
- behavior checks.

DEBUG STEP BY STEP

1 • SET UP A SMALL EXAMPLE

- Python file example:

main.py

```
1  price = 100
2  quantity = 2
3  total = price * quantity
4  print(total)
```

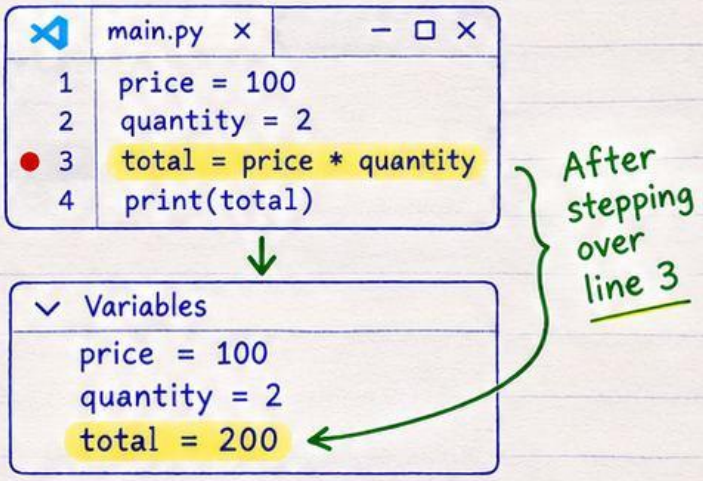
- Select the Python interpreter and install `Python Debugger`.
- Open `Run and Debug`.
- Press `F5` and choose `Python File` when prompted.
- Some projects need `launch.json` configuration.

2 • BREAKPOINT

- Click the gutter beside line 3, or press `F9` there.
- The debugger pauses before that line runs.
- Inspect `price = 100` and `quantity = 2` in Variables.

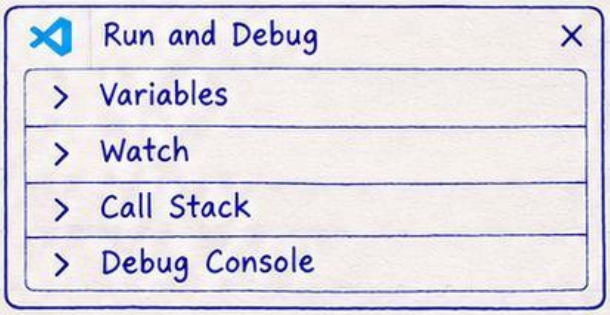
3 • STEP & OBSERVE

- `F10` : Step Over — execute current line.
- `F11` : Step Into — enter a function when possible.
- `Shift + F11` : Step Out — finish current function.
- `F5` : Continue — run to next breakpoint or end.
- `Shift + F5` : Stop.



4 • DEBUG PANELS

- `Variables`: current values.
- `Watch`: expressions to monitor.
- `Call Stack`: route to current function.
- `Debug Console`: inspect expressions in the paused context.



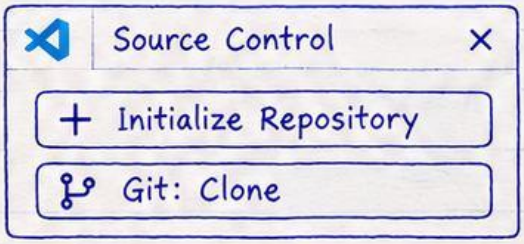
5 • RUN vs DEBUG

- `Ctrl + F5` runs without debugging.
- Use debugging when you need to see why a result is wrong.

GIT BASICS & DAILY WORKFLOW

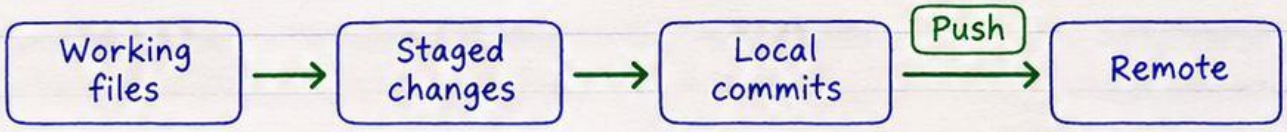
1 • GET READY

- Install Git separately.
- Open the project folder.
- `Ctrl + Shift + G` opens Source Control.
- Initialize Repository for a new project, or `Git: Clone` for an existing repository.



2 • THE DAILY FLOW

- `Edit` > `Review diff` > `Stage selected changes` > `Write a clear message` > `Commit`
- A commit saves a local snapshot.
- `Push` uploads commits to a remote.
- `Pull` brings remote changes into your branch.
- `Sync Changes` may pull and push.



3 • REVIEW BEFORE COMMIT

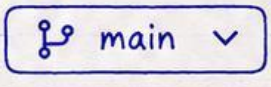
- Click a changed file to see its diff.
- Stage only intended files using the plus button `+`.
- Example commit message: `Add welcome message.`

4 • IGNORE PRIVATE / GENERATED FILES

- Use `.gitignore` for `.venv/`, `node_modules/` and `.env` as appropriate.
- Ignoring does not untrack an already tracked file.
- Never commit passwords or API keys.

5 • BRANCHES & CONFLICTS

- Use the branch name in the status bar to create/switch branches.
- Save or commit work before switching.
- Review comflicts, resolve them, test, then commit.



6 • FINAL HABITS

- Open the right folder. Save. Read Problems. Run/test. Review diffs. Commit small changes. Push when ready.
- `Discard Changes` can remove uncommitted work—review before using it.