

Sidebar

Programming · VS Code Debugging and launch.json Cheat Sheet

Debugging controls, breakpoint types, the structure of `launch.json`, the Python (debugpy) attributes, and the `debug.*` settings. Exact ID, what it does, and its default. Sheet 9 in the VS Code series.

RUNNING & STEPPING

- F5** Start debugging (or continue when paused)
- Ctrl+F5** Run without debugging
- Shift+F5 / Ctrl+Shift+F5** Stop / restart the debug session
- F10 / F11 / Shift+F11** Step over / step into / step out
- Ctrl+Shift+D** Open the Run and Debug view
- Ctrl+Shift+Y** Open the Debug Console
- Run > Add Configuration...** Add an entry to `launch.json` from a template

BREAKPOINTS

- F9** Toggle a breakpoint on the current line (or click in the gutter)
- Add Conditional Breakpoint...** Right-click the gutter: stop only when an expression is true, or after a hit count
- Add Logpoint...** Right-click the gutter: print a message, with `{variable}` in braces, without stopping
- Shift+F9** Add an inline breakpoint at the cursor (several per line)
- Breakpoints view** Enable, disable, edit or remove breakpoints; checkbox on each, and a button to toggle all
- debug.gutterMiddleClickAction** Middle-click on the gutter: `logpoint` (default), `conditionalBreakpoint`, `triggeredBreakpoint`, `none`
- debug.allowBreakpointsEverywhere** Allow breakpoints in any file type (default `false`)
- debug.showBreakpointsInOverviewRuler** Mark breakpoints in the scroll bar (default `false`)

LAUNCH.JSON BASICS

- .vscode/launch.json** The file in your project that stores debug configurations
- "version": "0.2.0"** The version field at the top of the file
- configurations** The array of debug setups that appear in the Run and Debug dropdown
- type** Required: which debugger to use, e.g. `debugpy`, `node`, `msedge`
- request** Required: `launch` (start the program) or `attach` (join one that is running)
- name** Required: the label shown in the dropdown
- compounds** Start several configurations together:
`{ "name": "...", "configurations": [...], "stopAll": true }`

COMMON LAUNCH.JSON ATTRIBUTES

- program** The file or executable to run
- args** Arguments passed to the program (an array)
- cwd** Working folder for the run
- env / envFile** Environment variables inline / loaded from a file
- console** `internalConsole`, `integratedTerminal`, `externalTerminal`
- stopOnEntry** Pause as soon as the program starts
- preLaunchTask / postDebugTask** Run a task (from `tasks.json`) before / after the session
- internalConsoleOptions** When the Debug Console opens: `neverOpen`, `openOnSessionStart`, `openOnFirstSessionStart`
- presentation** Order, group and hidden state in the dropdown
- serverReadyAction** Do something (e.g. open a browser) when the app says its server is ready
- windows / linux / osx** Platform blocks that override other attributes (but not `type`)

VARIABLES YOU CAN USE IN LAUNCH.JSON

<code>\${workspaceFolder}</code>	The folder you opened in VS Code
<code>\${file}</code>	The file in the active editor
<code>\${relativeFile}</code>	That file, relative to the workspace folder
<code>\${fileBasename}</code>	File name with extension, e.g. <code>app.py</code>
<code>\${fileBasenameNoExtension}</code>	File name without extension, e.g. <code>app</code>
<code>\${fileDirname}</code>	Folder of the active file
<code>\${env:NAME}</code>	The value of an environment variable
<code>\${input:id}</code>	A value you are asked for when the session starts (defined in <code>inputs</code>)
<code>\${command:id}</code>	The result of a VS Code command

PYTHON DEBUGGER (TYPE: DEBUGPY)

<code>program</code>	A Python file to run, e.g. <code>\${file}</code>
<code>module</code>	Run a module instead, e.g. <code>"uvicorn"</code>
<code>args</code>	For FastAPI: <code>["main:app", "--reload"]</code> with <code>module</code> set to <code>uvicorn</code>
<code>console</code>	<code>integratedTerminal</code> (default), <code>externalTerminal</code> , <code>internalConsole</code>
<code>justMyCode</code>	<code>true</code> (default): skip library code when stepping; set <code>false</code> to step into it
<code>stopOnEntry</code>	Pause on the first line (default <code>false</code>)
<code>subProcess</code>	Also debug child processes (default <code>false</code>)
<code>showReturnValue</code>	Show a function's return value after stepping over it (default <code>true</code>)
<code>redirectOutput</code>	Send program output to the Debug Console (default <code>true</code>)
<code>django / jinja / pyramid</code>	Switches that enable framework-specific debugging features
<code>python / pythonArgs</code>	Use a specific interpreter / pass options to it

DEBUG BEHAVIOUR (DEBUG.*)

<code>debug.openDebug</code>	When to open the Debug view: <code>openOnDebugBreak</code> (default), <code>openOnSessionStart</code> , <code>openOnFirstSessionStart</code> , <code>neverOpen</code>
<code>debug.saveBeforeStart</code>	Save before a session: <code>allEditorsInActiveGroup</code> (default), <code>nonUntitledEditorsInActiveGroup</code> , <code>none</code>
<code>debug.focusEditorOnBreak</code>	Bring the editor into focus when paused (default <code>true</code>)
<code>debug.focusWindowOnBreak</code>	Bring the window to the front when paused (default <code>true</code>)
<code>debug.onTaskErrors</code>	If the pre-launch task fails: <code>prompt</code> (default), <code>debugAnyway</code> , <code>showErrors</code> , <code>abort</code>
<code>debug.confirmOnExit</code>	Ask before closing VS Code during a session: <code>never</code> (default) or <code>always</code>
<code>debug.inlineValues</code>	Show variable values beside the code: <code>auto</code> (default), <code>on</code> , <code>off</code>
<code>debug.showVariableTypes</code>	Show types in the Variables view (default <code>false</code>)
<code>debug.toolbarLocation</code>	<code>floating</code> (default), <code>docked</code> , <code>commandCenter</code> , <code>hidden</code>
<code>debug.showInStatusBar</code>	<code>onFirstSessionStart</code> (default), <code>always</code> , <code>never</code>
<code>debug.enableStatusBarColor</code>	Change the status bar colour while debugging (default <code>true</code>)
<code>debug.terminal.clearBeforeReusing</code>	Clear a reused debug terminal first (default <code>false</code>)

DEBUG CONSOLE (DEBUG.CONSOLE.*)

<code>debug.console.fontSize</code>	Text size (default 14; 12 on macOS)
<code>debug.console.fontFamily</code>	Font (default: follows the editor)
<code>debug.console.lineHeight</code>	<code>0</code> (default) means automatic
<code>debug.console.wordWrap</code>	Wrap long lines (default <code>true</code>)
<code>debug.console.maximumLines</code>	Lines kept before old ones drop off (default 10000)
<code>debug.console.collapseIdenticalLines</code>	Fold repeated identical lines into one with a count (default <code>true</code>)
<code>debug.console.historySuggestions</code>	Suggest earlier input as you type (default <code>true</code>)
<code>debug.console.acceptSuggestionOnEnter</code>	Enter accepts the suggestion: <code>off</code> (default) or <code>on</code>
<code>debug.console.closeOnEnd</code>	Close the console when the session ends (default <code>false</code>)

OTHER COMMON DEBUGGER TYPES (EXAMPLES)

```
"type": "node", "program": "${workspaceFolder}/app.js"
```

Node.js: run a script; add `skipFiles` to avoid stepping into Node's own code

```
"type": "msedge", "url": "http://localhost:3000"
```

Browser debugging in Edge, with `webRoot` pointing at your source

```
"type": "chrome", "url": "http://localhost:3000"
```

The same idea for Chrome

```
request: attach
```

Join a program that is already running in debug mode, using `port` or `processId`

A STARTING POINT FOR DEBUGGING A FASTAPI APP

In `.vscode/launch.json`, inside `configurations`: `{ "name": "FastAPI", "type": "debugpy", "request": "launch", "module": "uvicorn", "args": ["main:app", "--reload"], "jinja": true }`. Adjust `main:app` to your own file and app name. With `--reload` the server restarts on file changes; if breakpoints behave oddly, try removing it.

BREAKPOINTS IN LIBRARY CODE ARE SKIPPED? THAT'S JUSTMYCODE

For Python, `justMyCode` defaults to `true`, so the debugger steps over code in installed packages. Set `"justMyCode": false` in the configuration when you want to step into a library.