Neovim The Editor

table of content

Introduction

If you're reading this, you might be familiar with code editor like Visual Studio Code, Sublime or IDEs like Dev C++, Pycharm,... They all have one thing in common is that they are resource-heavy and bloated. Sure some of them are indeed light-weight and some are heavy because of built-in features in them, but what if I told you Neovim can do the same thing but better and faster?

So what Is Neovim?

Neovim is a highly extensible keyboard-based editor on terminal-based where your mouse is almost completely useless, and because of that, it can can improve your productivity since you won't need to touch the mouse. Neovim originally came from Vim or Vi and also a drop-in replacement of it, meaning Neovim are 99% identical to Vim but most of it's code base are refactored to be better, this also help the community to make faster and stronger plugins for Neovim.

Why use Neovim?

Here are **3 reasons** why I pick Neovim for my daily coding editor:

1. It's insanely fast in navigation and minimalist workflow.
2. The Neovim customization is beyond your own limits (or someone else plugins/config).
3. Neovim is light-weight which can be run in everywhere (especially server).

What is Neovim capable of?

To be honest Neovim can be use to write anything you want not just coding. You can write diary, note, plan, etc and with plugins, you can improve the experience of writing which we will cover in the last section.

Most of the time I use Neovim for coding because how fast and strong it is and sometime I use Neovim for writing like the one you reading right now. But you can see Neovim as a empty Notebook, a Journal or even just a paper.

Installing Neovim

Enough with the introduction, now we will install Neovim into our system.

Linux

Neovim exist in almost Linux distributions repositories due to its popularity so most likely you can install it using your distrubution's package manager.

1. Arch (btw)

```
$ sudo pacman -S neovim
```

2. Ubuntu/Debian

```
$ sudo apt install neovim
```

3. Fedora

```
$ sudo dnf install -y neovim python3-neovim
```

Windows

If you using windows especially windows 11 then `winget` (A package manager for windows) already installed in your computer. Simply use this command to install.

```
winget install Neovim.Neovim
```

There's also alternative package manager for windows like Scoop or Chocolatey and if you using any of it, use these command

1. Scoop

```
scoop bucket add main
scoop install neovim
```

2. Chocolatey

```
choco install neovim
```

If you use neither of those then you can visit Neovim official repo and look for `nvim-win64.msi`

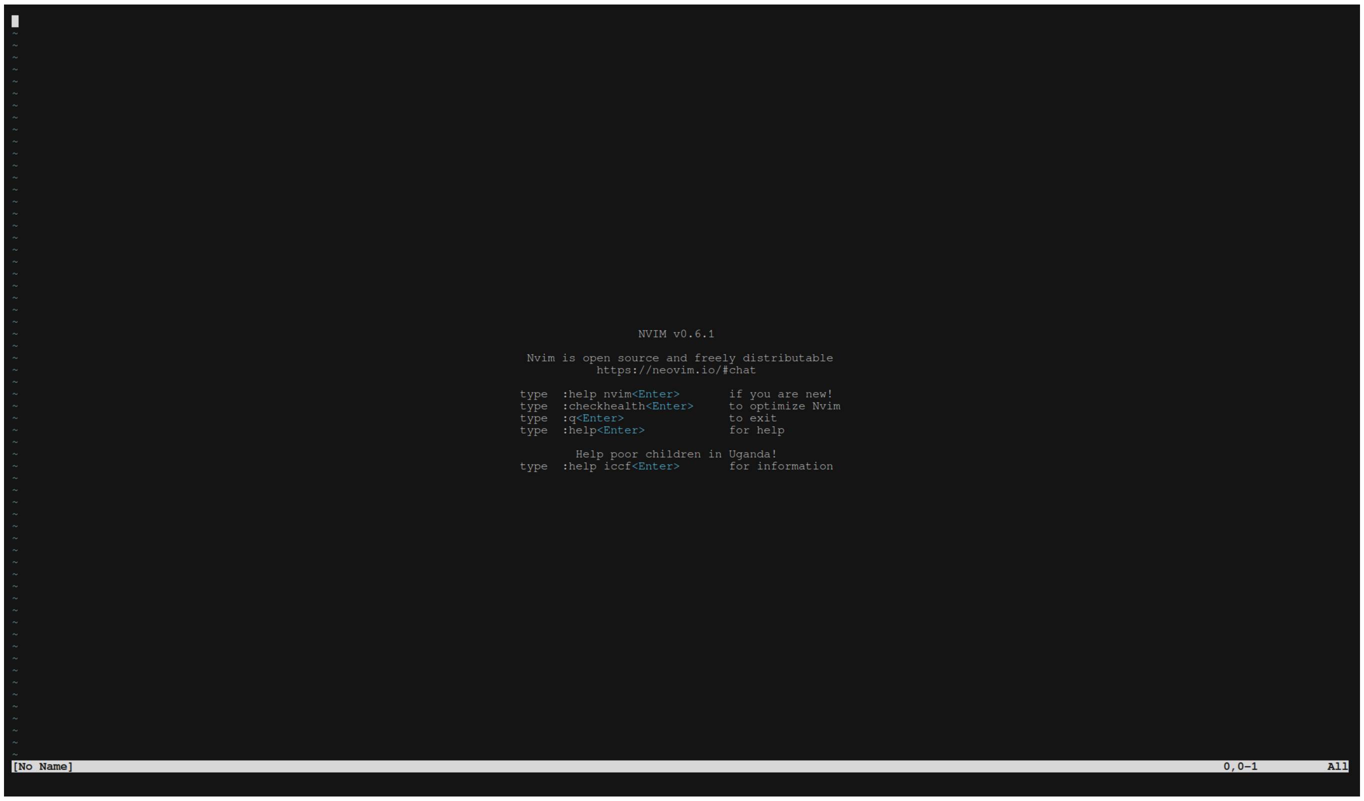
MacOS

If you using homebrew (which you should be), you can install Neovim with the following command:

```
brew install neovim
```

Navigation

When you first open up Neovim and what first pop up is:



```
                       NVIM v0.6.1

        Nvim is open source and freely distributable
                  https://neovim.io/#chat

       type  :help nvim<Enter>       if you are new!
       type  :checkhealth<Enter>     to optimize Nvim
       type  :q<Enter>               to exit
       type  :help<Enter>            for help

               Help poor children in Uganda!
       type  :help iccf<Enter>       for information
```

```
[No Name]                                              0,0-1          All
```

You may think: **"How can this post claim Neovim outperforms my current editor, I literally can't even type!, how can I move around?, I can't do anything! HOW DO I EXIT THIS?!?!?"**.

That's because you haven't set everything up so it's can usable. Neovim becomes powerful when you learn how to use it and invest time customizing or install plugins to it.

Basic Motion

Unlike any editors, you have to use your mouse or arrow keys to move the cursor. Here in Neovim, you can move around using the keyboard thus improve your speed. So in this section, we will cover the basic motion of Neovim.

First to open a file with Neovim use the following command:

```
nvim <file_name>
```

If the file doesn't exist then Neovim will create a new one upon saving.

There are 4 main modes in Neovim:

NORMAL Mode (Default Mode)

- Used for navigating and executing commands.

- Press `<ESC>` to return to this mode from other modes.

INSERT Mode (Typing Mode)

- Allows you to type like a normal editor.
- Press `i` to enter INSERT mode.

VISUAL Mode (Selection Mode)

- Used to select texts.
- Press `v` for selection.
- Press `V` for select entire line.

Command-line Mode (For Command)

- Press `:` to enter this mode.
- Used for command like saving and quitting.

Now we'll move on to Navigation or how to move around in **NORMAL** mode.

Basic movements:

- `h` → Move left
- `l` → Move right
- `j` → Move down
- `k` → Move up

Vertical movement:

- `Ctrl + u` → Move **up** half a page
- `Ctrl + d` → Move **down** half a page
- `G` → Jump to the **bottom** of the file
- `gg` → Jump to the **top** of the file
- `{` → Move **up** by paragraph
- `}` → Move **down** by paragraph

Searching

- `/word` → Search **forward** for "word"
- `?word` → Search **backward** for "word"
- `n` → Jump to the **next** match
- `N` → Jump to the **previous** match

- `*` → Search for the **word under the cursor** forward
- `#` → Search for the **word under the cursor** backward

Horizontal movements:

- `w` → Jump **forward** to the start of the next word
- `e` → Jump **forward** to the end of the current/next word
- `b` → Jump **backward** to the start of the previous word
- `f<char>` → Jump to the next occurrence of `<char>` in the current line
- `F<char>` → Jump to the previous occurrence of `<char>` in the current line
- `;` → Repeat the last `f` or `F` search
- `,` → Repeat the last `f` or `F` search in reverse
- `$` → Jump to the **end** of the line
- `0` → Jump to the **beginning** of the line

Editing Commands

- `d` → Delete selected text
- `dd` → Delete the current line
- `y` → Yank (copy) selected text
- `yy` → Yank the current line
- `p` → Paste after the cursor

Saving and Quitting

- `:w` → Save the file
- `:q` → Quit Neovim
- `:wq` or `ZZ` → Save and quit
- `:q!` → Quit without saving

Example

```
interface User {
  name: string;
  id: number;
}

class UserAccount {
  name: string;
  id: number;

  constructor(name: string, id: number) {
    this.name = name;
    this.id = id;
  }
}

const user: User = new UserAccount("Murphy", 1);
```

I have this example Typescipt code and I want to change `"Murphy"` into something else. The most simple way to this is use `G` to go the last line then `fM` to move into the string and `ve` to select the string, next press `d` to delete then `i` to change whatever you want.

Link to demo

Saving and quitting

To quit a file simply enter command mode then type `wq`

- `w` stands for saving a file.
- `q` stands for quitting a file.

So if you want to save or quit then use only one of them in command mode.

And this is almost every basic Neovim motion, it might hard at first that you cannot remember all of the keybinds but you can always learn to use it effectively but for now pick some that you think you will use the most and goes on till you master it. Also you can use `:help` to help you understand more about some about vim.

Plugins

Now this one might be the most fun part because now you will try to install plugins into it to make it more powerfull. To make things easier, we will use a Neovim distro called kickstart.nvim. Unlike others distro where they did everything for you, this one only install the bare minimum for it like a package manager, LSP server,..., and some pre-configuration.

Requirements

- Neovim >= 0.8.0 (needs to be built with LuaJIT)
- Git >= 2.19.0 (for partial clones support)
- a Nerd Font (optional but recommend for better experience)

Installation

First you need to locate where your Neovim configuration is locate

- For Linux simply go to `~/.config/nvim`.
- For Windows go to `Appdata/Local/nvim`. Or `%localappdata%\nvim\`
- For MacOS, it's basically the same for Linux `~/config/nvim`.

If it not exist, create one your own.

The recommended way to install Kickstart.nvim is to fork one your own from this link then install it by cloning the fork to your nvim folder by using this command below.

Linux/Mac

```
git clone https://github.com/nvim-lua/kickstart.nvim.git
"${XDG_CONFIG_HOME:-$HOME/.config}"/nvim
```

Windows

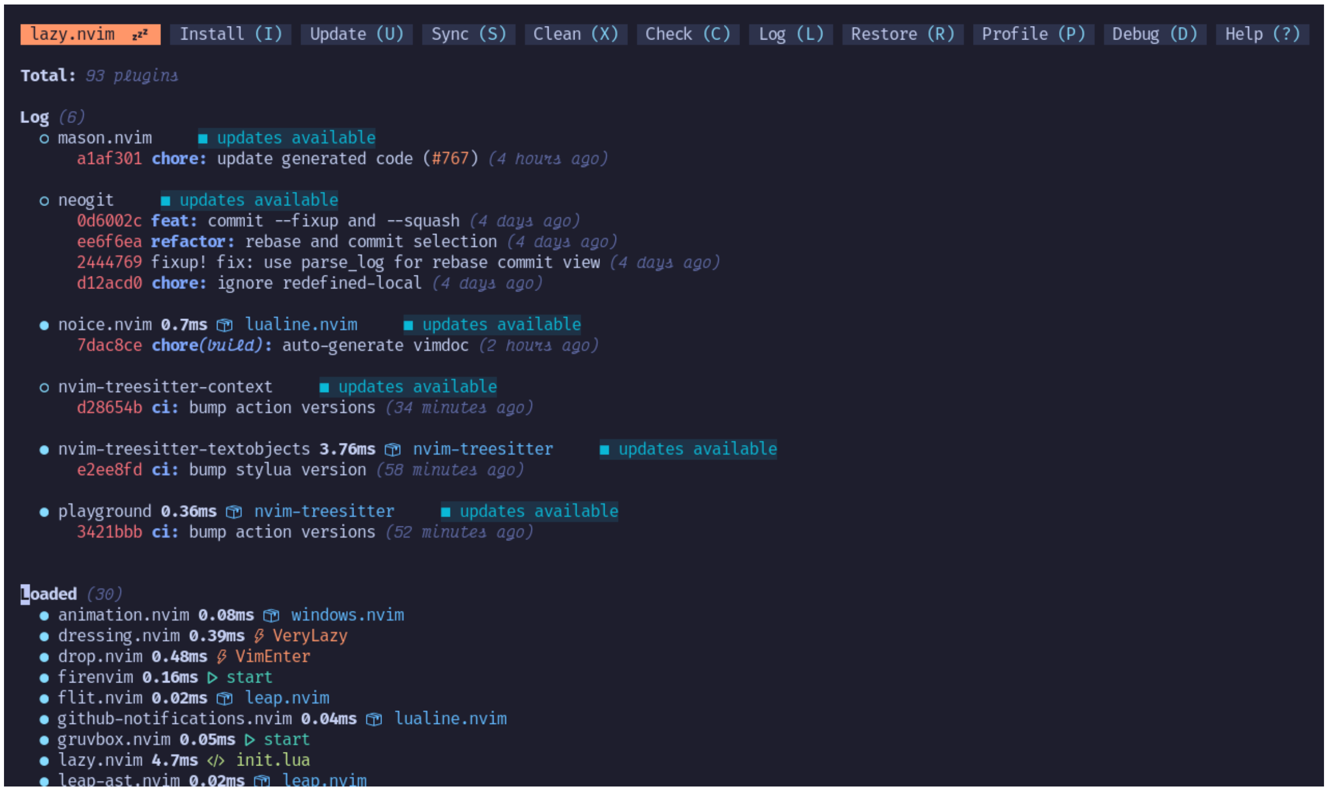
If you use `cmd.exe`

```
git clone https://github.com/nvim-lua/kickstart.nvim.git
"%localappdata%\nvim"
```

If you use `powershell.exe`

```
git clone https://github.com/nvim-lua/kickstart.nvim.git
"${env:LOCALAPPDATA}\nvim"
```

After you installed it Lazy.nvim (a package manager) will install all the plugins you have. Use `:Lazy` to view all the plugins status. Hit `q` to close the window.



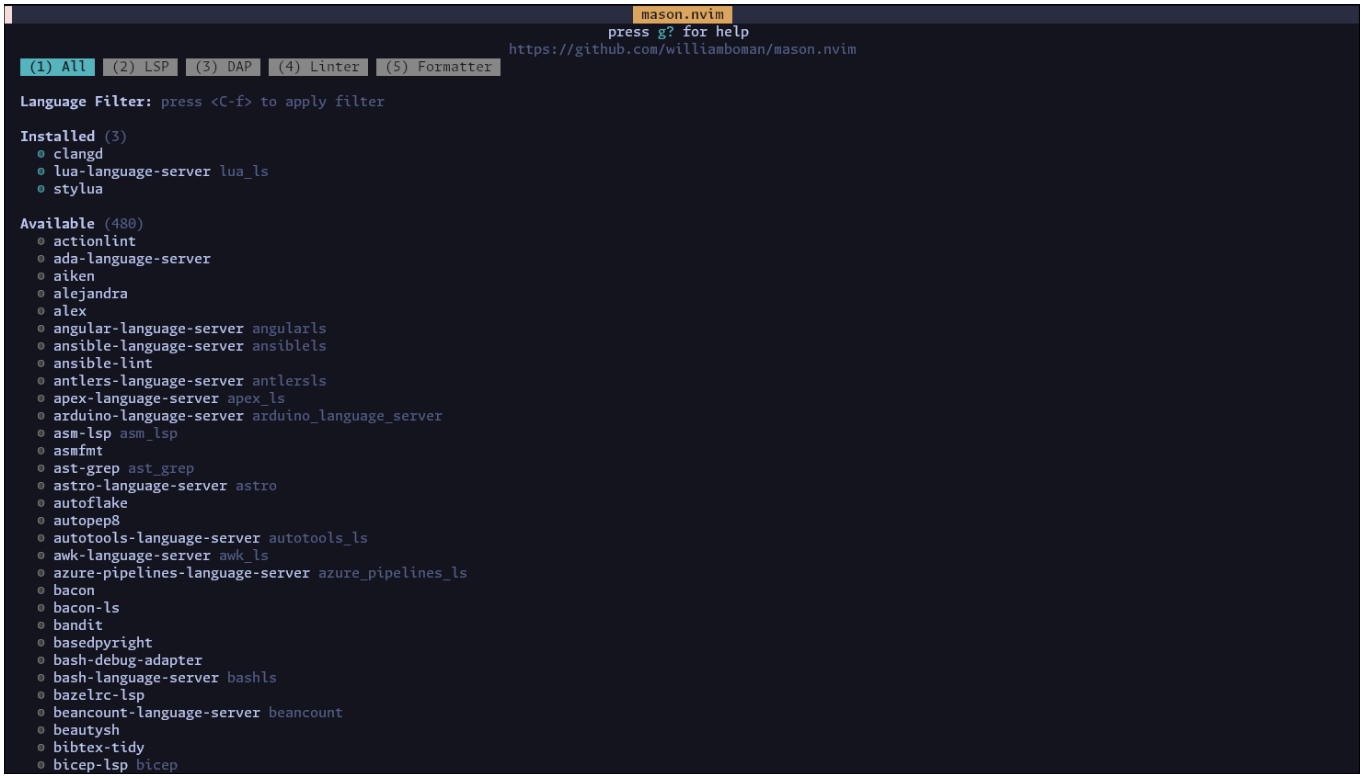
```
 lazy.nvim 💤   Install (I)   Update (U)   Sync (S)   Clean (X)   Check (C)   Log (L)   Restore (R)   Profile (P)   Debug (D)   Help (?)

Total: 93 plugins

Log (6)
  ○ mason.nvim     ■ updates available
      a1af301 chore: update generated code (#767) (4 hours ago)

  ○ neogit     ■ updates available
      0d6002c feat: commit --fixup and --squash (4 days ago)
      ee6f6ea refactor: rebase and commit selection (4 days ago)
      2444769 fixup! fix: use parse_log for rebase commit view (4 days ago)
      d12acd0 chore: ignore redefined-local (4 days ago)

  ● noice.nvim 0.7ms 📦 lualine.nvim     ■ updates available
      7dac8ce chore(build): auto-generate vimdoc (2 hours ago)

  ○ nvim-treesitter-context     ■ updates available
      d28654b ci: bump action versions (34 minutes ago)

  ● nvim-treesitter-textobjects 3.76ms 📦 nvim-treesitter     ■ updates available
      e2ee8fd ci: bump stylua version (58 minutes ago)

  ● playground 0.36ms 📦 nvim-treesitter     ■ updates available
      3421bbb ci: bump action versions (52 minutes ago)


Loaded (30)
  ● animation.nvim 0.08ms 📦 windows.nvim
  ● dressing.nvim 0.39ms ⚡ VeryLazy
  ● drop.nvim 0.48ms ⚡ VimEnter
  ● firenvim 0.16ms ▷ start
  ● flit.nvim 0.02ms 📦 leap.nvim
  ● github-notifications.nvim 0.04ms 📦 lualine.nvim
  ● gruvbox.nvim 0.05ms ▷ start
  ● lazy.nvim 4.7ms </> init.lua
  ● leap-ast.nvim 0.02ms 📦 leap.nvim
```

Lazy.nvim Example

Then you should briefly read the Friendly Document in `init.lua` from you nvim config folder to further understand about Neovim configuration.

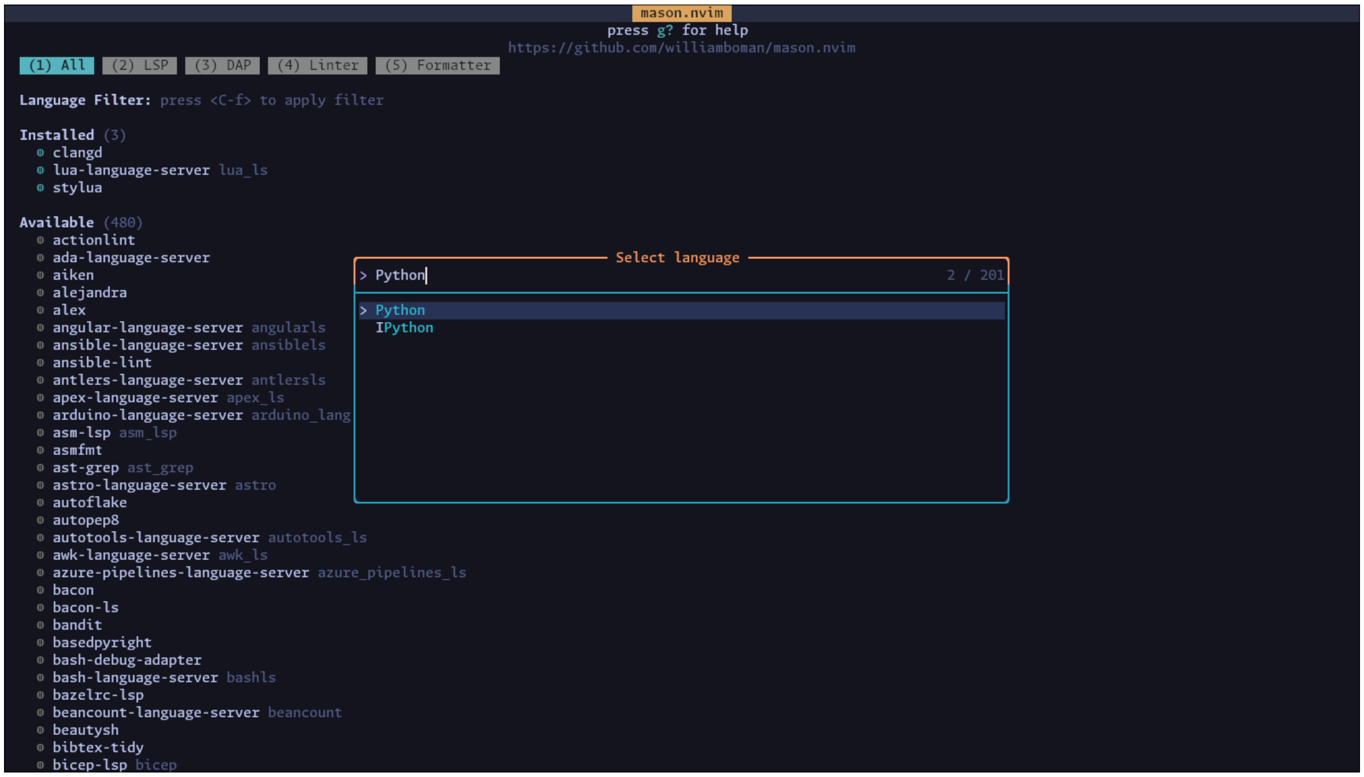
Kickstart.nvim already setup for LSP server or IntelliSense you normally see in VS code and other IDE.

For example: To install new LSP like Python use command `:Mason` it'll have a popup that contain available LSP.



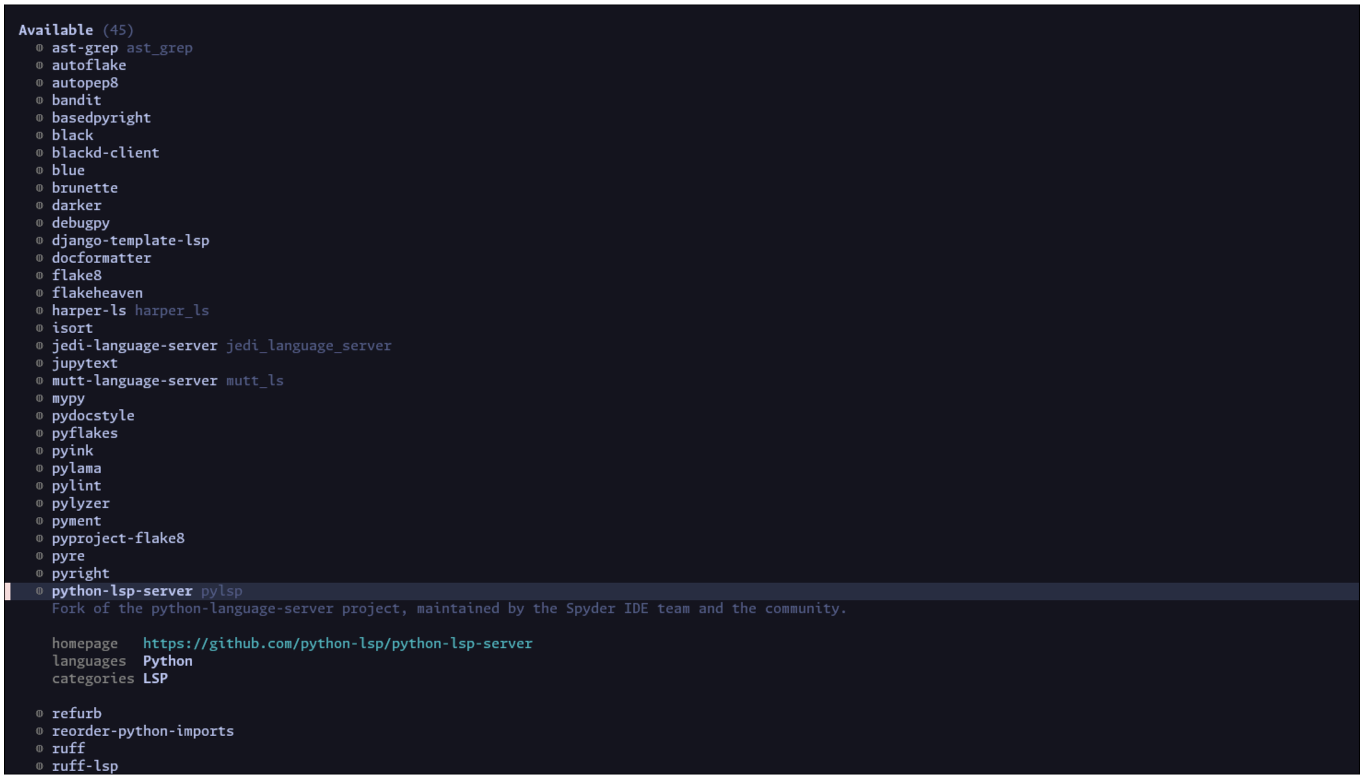
```
mason.nvim
press g? for help
https://github.com/williamboman/mason.nvim
(1) All  (2) LSP  (3) DAP  (4) Linter  (5) Formatter

Language Filter: press <C-f> to apply filter

Installed (3)
  o clangd
  o lua-language-server lua_ls
  o stylua

Available (480)
  o actionlint
  o ada-language-server
  o aiken
  o alejandra
  o alex
  o angular-language-server angularls
  o ansible-language-server ansiblels
  o ansible-lint
  o antlers-language-server antlersls
  o apex-language-server apex_ls
  o arduino-language-server arduino_language_server
  o asm-lsp asm_lsp
  o asmfmt
  o ast-grep ast_grep
  o astro-language-server astro
  o autoflake
  o autopep8
  o autotools-language-server autotools_ls
  o awk-language-server awk_ls
  o azure-pipelines-language-server azure_pipelines_ls
  o bacon
  o bacon-ls
  o bandit
  o basedpyright
  o bash-debug-adapter
  o bash-language-server bashls
  o bazelrc-lsp
  o beancount-language-server beancount
  o beautysh
  o bibtex-tidy
  o bicep-lsp bicep
```

Then press `Ctrl + f` to filter out languages and find Python.



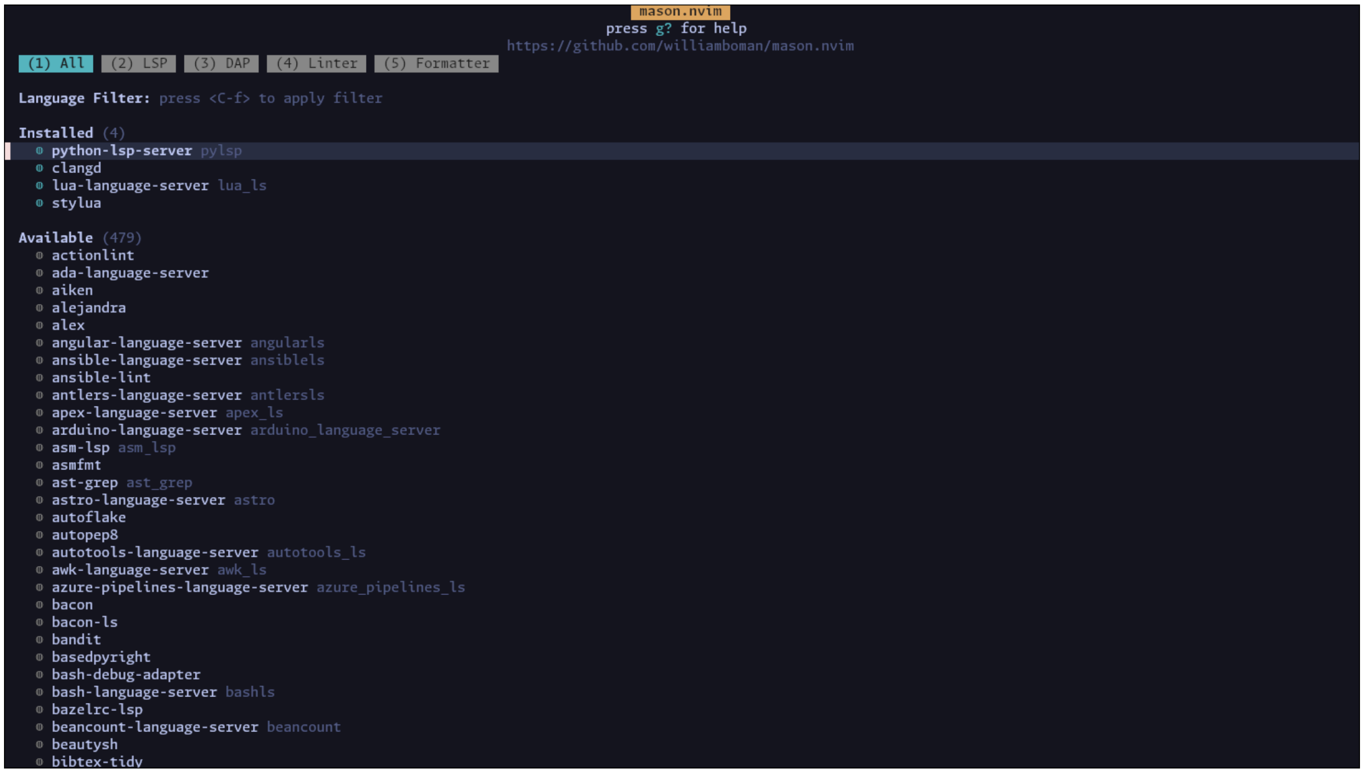
```
mason.nvim
press g? for help
https://github.com/williamboman/mason.nvim
(1) All  (2) LSP  (3) DAP  (4) Linter  (5) Formatter

Language Filter: press <C-f> to apply filter

Installed (3)
  o clangd
  o lua-language-server lua_ls
  o stylua

Available (480)
  o actionlint
  o ada-language-server
  o aiken              ─────── Select language ───────
  o alejandra          > Python                           2 / 201
  o alex               
  o angular-language-server angularls   > Python
  o ansible-language-server ansiblels     IPython
  o ansible-lint
  o antlers-language-server antlersls
  o apex-language-server apex_ls
  o arduino-language-server arduino_lang
  o asm-lsp asm_lsp
  o asmfmt
  o ast-grep ast_grep
  o astro-language-server astro
  o autoflake
  o autopep8
  o autotools-language-server autotools_ls
  o awk-language-server awk_ls
  o azure-pipelines-language-server azure_pipelines_ls
  o bacon
  o bacon-ls
  o bandit
  o basedpyright
  o bash-debug-adapter
  o bash-language-server bashls
  o bazelrc-lsp
  o beancount-language-server beancount
  o beautysh
  o bibtex-tidy
  o bicep-lsp bicep
```

There're lots of LSP here but scroll down a bit and you'll see one called python-lsp-server.



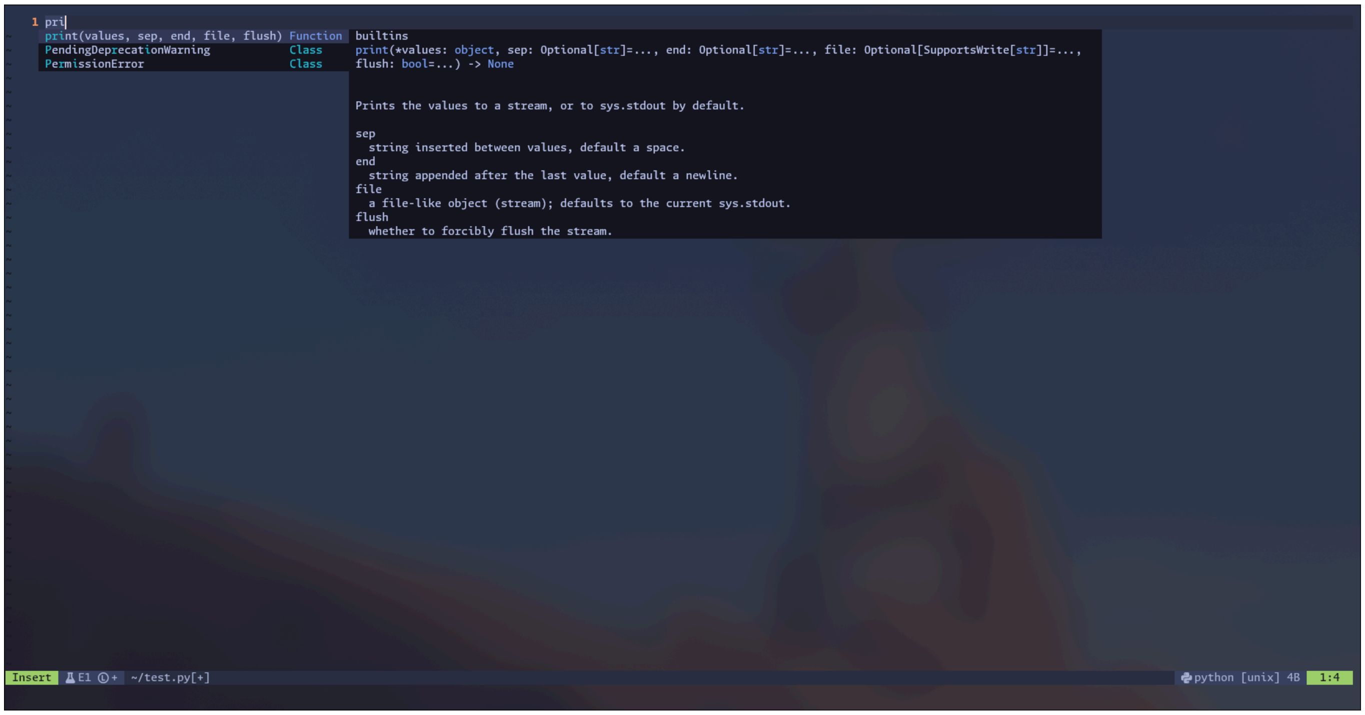
```
Available (45)
  ○ ast-grep ast_grep
  ○ autoflake
  ○ autopep8
  ○ bandit
  ○ basedpyright
  ○ black
  ○ blackd-client
  ○ blue
  ○ brunette
  ○ darker
  ○ debugpy
  ○ django-template-lsp
  ○ docformatter
  ○ flake8
  ○ flakeheaven
  ○ harper-ls harper_ls
  ○ isort
  ○ jedi-language-server jedi_language_server
  ○ jupytext
  ○ mutt-language-server mutt_ls
  ○ mypy
  ○ pydocstyle
  ○ pyflakes
  ○ pyink
  ○ pylama
  ○ pylint
  ○ pylyzer
  ○ pyment
  ○ pyproject-flake8
  ○ pyre
  ○ pyright
  ○ python-lsp-server pylsp
    Fork of the python-language-server project, maintained by the Spyder IDE team and the community.

    homepage   https://github.com/python-lsp/python-lsp-server
    languages  Python
    categories LSP

  ○ refurb
  ○ reorder-python-imports
  ○ ruff
  ○ ruff-lsp
```

Then press `i` to install it. After that Python should appear here.



```
                                        mason.nvim
                                    press g? for help
                          https://github.com/williamboman/mason.nvim
(1) All  (2) LSP  (3) DAP  (4) Linter  (5) Formatter

Language Filter: press <C-f> to apply filter

Installed (4)
  ○ python-lsp-server pylsp
  ○ clangd
  ○ lua-language-server lua_ls
  ○ stylua

Available (479)
  ○ actionlint
  ○ ada-language-server
  ○ aiken
  ○ alejandra
  ○ alex
  ○ angular-language-server angularls
  ○ ansible-language-server ansiblels
  ○ ansible-lint
  ○ antlers-language-server antlersls
  ○ apex-language-server apex_ls
  ○ arduino-language-server arduino_language_server
  ○ asm-lsp asm_lsp
  ○ asmfmt
  ○ ast-grep ast_grep
  ○ astro-language-server astro
  ○ autoflake
  ○ autopep8
  ○ autotools-language-server autotools_ls
  ○ awk-language-server awk_ls
  ○ azure-pipelines-language-server azure_pipelines_ls
  ○ bacon
  ○ bacon-ls
  ○ bandit
  ○ basedpyright
  ○ bash-debug-adapter
  ○ bash-language-server bashls
  ○ bazelrc-lsp
  ○ beancount-language-server beancount
  ○ beautysh
  ○ bibtex-tidy
```

Now whenever you open a python file the LSP should work out of the box like this:



```
pri
print(values, sep, end, file, flush) Function  builtins
PendingDeprecationWarning            Class     print(*values: object, sep: Optional[str]=..., end: Optional[str]=..., file: Optional[SupportsWrite[str]]=...,
PermissionError                      Class     flush: bool=...) -> None


                                               Prints the values to a stream, or to sys.stdout by default.

                                               sep
                                                 string inserted between values, default a space.
                                               end
                                                 string appended after the last value, default a newline.
                                               file
                                                 a file-like object (stream); defaults to the current sys.stdout.
                                               flush
                                                 whether to forcibly flush the stream.

Insert  E1 + ~/test.py[+]                                                                  python [unix] 4B  1:4
```

Right now Neovim should work like a charm with autocomplete and suggestion but if you want to extend even more, you can visit this github repo to find your favorite plugins

Also here are my list of current plugins I currently use:

- 'numToStr/Comment.nvim'
- 'stevearc/oil.nvim'
- 'windwp/nvim-autopairs'
- 'tribela/transparent.nvim'
- 'folke/twilight.nvim'
- 'folke/zen-mode.nvim'
- 'ThePrimeagen/vim-be-good'

If you read the Friendly Document then you should know where to put plugins so Lazy could install it. In case you don't, use search and find `require('lazy')` or just `lazy`.

Conclusion

Neovim is indeed a strong tool for your writing but only you actually thinking about using it right. If you don't feel like it you can still use your favorite code editors or IDEs but it is a thing that you should at least try once especially you're in IT field because I, myself learnt a lots of how things work under the hood when trying it.