Learning Haskell. For beginners there is a self study course at https://github.com/system-f/fp-course.
If you want a gentle video class, you could try the class by Andres Löh, Introduction to Haskell
If you want an even gentler video class, you could try the class by Tea Leaves, Haskell for dilettantes, which works through some of class materials from Joachim Breitner's UPenn class CS194.

Installing Haskell. On linux or mac, install Haskell with ghcup, from

```
https://www.haskell.org/ghcup/install/
```

Then you can upgrade and check the installation with

```
% ghcup upgrade
% ghcup tui
```

In the tui display, one check means installed, two checks mean set for use. Generally, I set the versions listed as 'recommended' in that display. See https://www.haskell.org/ghcup/steps/ on using external packages in `ghci`, etc.

(On windows-11, I was able to install ghcup on windows subsystem ubuntu, just as in a standard linux, after first installing: build-essential findutils binutils curl gcc g++ libgmp-dev libc-dev libffi-dev make musl-dev ncurses-dev perl tar xz-utils.)

Optional: Literate Haskell To process literate Haskell files (*.lhs), install `lhs2TeX` and `unlit`, used to produce this documentation:

```
% stack install unlit
% stack install lhs2tex
```

Then `unlit` produces Haskell (*.hs) from literate Haskell (*.lhs). For the `installingHaskellEtc.lhs` file I am editing now,

```
% unlit -i installingHaskellEtc.lhs -o installingHaskellEtc.hs
```

For code documentation, line-by-line, `lhs2tex` produces a latex file for generating pdf, the pdf which you are presumably looking at now. This file, `installingHaskellEtc.pdf`, is produced and opened with these steps:

```
% lhs2tex -o installingHaskellEtc.tex installingHaskellEtc.lhs
% pdflatex -shell-escape installingHaskellEtc.tex
% open installingHaskellEtc.pdf
```

Here in the pdf format, the Haskell code which is now in `installingHaskellEtc.hs` looks like this:

$main :: IO\ ()$
$main = putStrLn$ `"Hello, world!"`

For code publication, the Haskell code in `installingHaskellEtc.hs` can be much more colorful and perhaps more beautiful with `minted`:

```haskell
main :: IO ()
main = putStrLn "Hello, world!"
```

The `-shell-escape` flag for `pdflatex` is required to use the `minted` package.

Optional: emacs with haskell language server. This is the development environment I prefer. After installing a recent version of emacs, inside emacs, I use package-install to install: "haskell-mode" Then I get an easy connection with language servers for syntax highlighting, code completion, etc. by adding the eglot initialization recommended here:

https://haskell-language-server.readthedocs.io/en/latest/configuration.html#emacs

Optional: tmux+neovim with haskell language server Nvim also provides easy connection with language servers for syntax highlighting, code completion, etc. Tmux allows you to split screen to run editor plus session (plus anything else) all at once, with easy switching. The combination is a very flexible and extensible IDE, one that is actively used and developed recently.

Ubuntu nvim Intro youtube vids, useful for those not already familiar with nvim and tmux:

- "Dreams of Code" vid on nvim config and basics

- "Dreams of Code" vid on tmux config and basics

On Ubuntu 24.04, I installed a nerdfont, ripgrep, tmux, tmux package manager (tpm), etc

```
% sudo apt install ripgrep
% sudo apt install nodejs npm
% sudo apt install luarocks
% sudo apt install tmux
% git clone https://github.com/tmux-plugins/tpm ~/.tmux/plugins/tpm
% wget -P ~/.local/share/fonts https://github.com/ryanoasis/nerd-fonts/releases/latest/download/Meslo.zip
% cd ~/.local/share/fonts
% unzip Mezlo.zip
% rm Mezlo.zip
% sudo fc-cache -fv
% sudo npm i -g hls
% sudo npm i -g pyright
% ghcup install hls
% sudo luarocks install haskell-tools
% stack install haskell-dap ghci-dap haskell-debug-adapter
```

Then in a terminal window, in text settings, I selected the Meslo mono font. Then following the instructions here:

https://nvchad.com/docs/quickstart/install/

In nvim, when running

```
:MasonInstallAll,
```

you can scroll down to install haskell-language-server and pyright (and whatever else you want). And in nvim, add syntax highlighting for whatever languages you want:

```
:TSInstall haskell
:TSInstall python
:TSInstall texlab
```

Then add these lines to `.config/nvim/init.lua`:

```
require 'lspconfig'.pyright.setup{}
require 'lspconfig'.hls.setup{}
```

And in that same file, inside `require("lazy").setup({ ...})` add

```
  {
    'mrcjkb/haskell-tools.nvim',
     version = '^3', -- Recommended
     lazy = false, -- This plugin is already lazy
  },
```

Then create dir and file ~/`.config/nvim/after/ftplugin/haskell.lua` with contents:

```
-- ~/.config/nvim/after/ftplugin/haskell.lua
local ht = require('haskell-tools')
local bufnr = vim.api.nvim_get_current_buf()
local opts = { noremap = true, silent = true, buffer = bufnr, }
-- haskell-language-server relies heavily on codeLenses,
-- so auto-refresh (see advanced configuration) is enabled by default
vim.keymap.set('n', '<space>cl', vim.lsp.codelens.run, opts)
-- Hoogle search for the type signature of the definition under the cursor
vim.keymap.set('n', '<space>hs', ht.hoogle.hoogle_signature, opts)
-- Evaluate all code snippets
vim.keymap.set('n', '<space>ea', ht.lsp.buf_eval_all, opts)
-- Toggle a GHCi repl for the current package
vim.keymap.set('n', '<leader>rr', ht.repl.toggle, opts)
-- Toggle a GHCi repl for the current buffer
vim.keymap.set('n', '<leader>rf', function() ht.repl.toggle(vim.api.nvim_buf_get_name(0)) end, opts)
vim.keymap.set('n', '<leader>rq', ht.repl.quit, opts)
```

Mac nvim   On Mac, first,

- At the time of this writing (May 2024) something is very wrong with the lua screen and color control for my default Mac terminal, so I installed the kitty terminal, tmux, etc:

  ```
  brew install kitty tmux tpm
  ```

- I installed all the homebrew nerd fonts:

  ```
  brew tap homebrew/cask-fonts
  brew search '/font-.*-nerd-font/' \ensuremath{\Varid{awk}\;\text{\ttfamily '\char123 ~print~\$1~\char125 '}} xargs -I{} brew install --cask {}
  ```

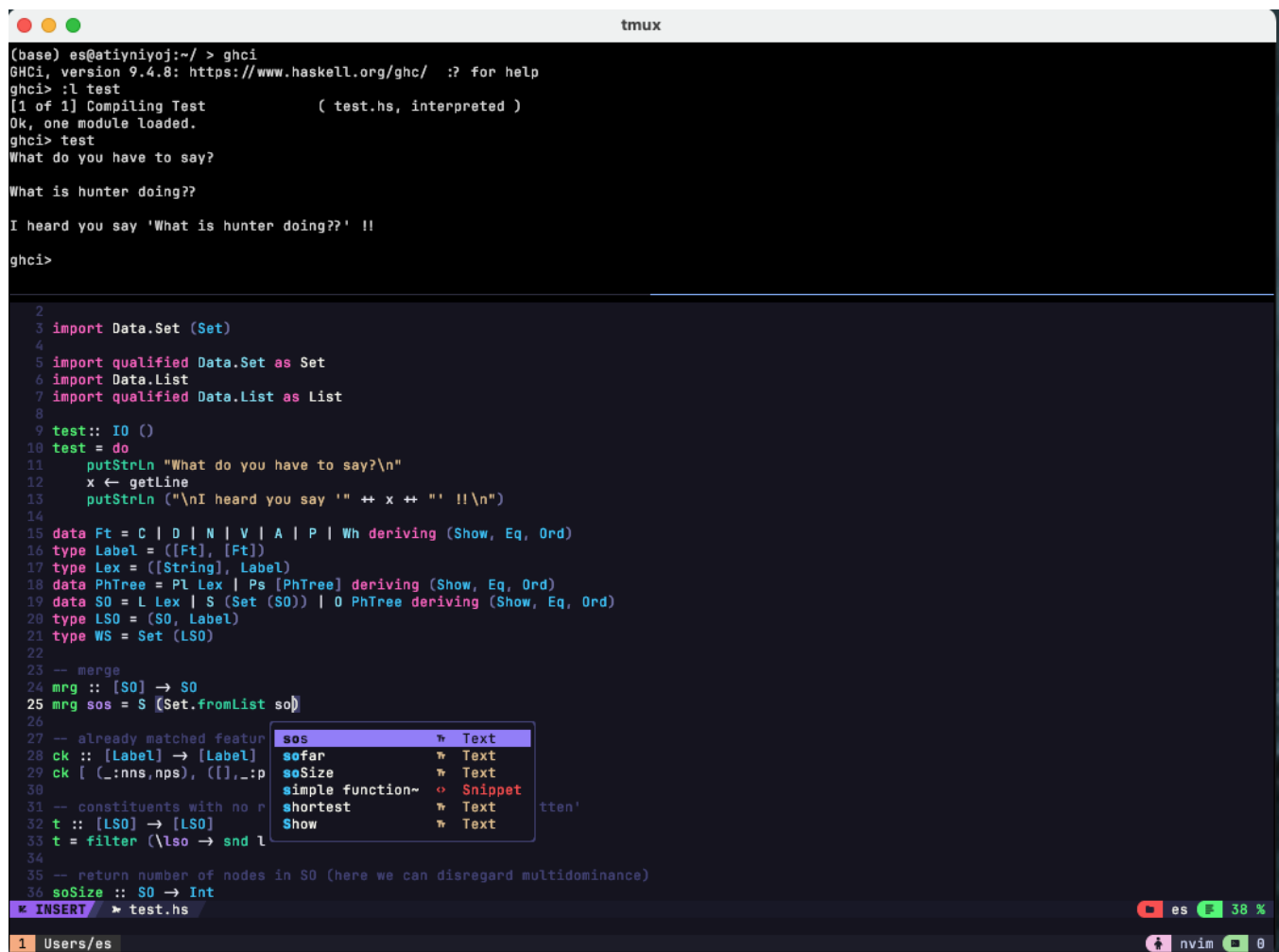- I installed ripgrep, haskell-language-server, debug tools

  ```
  brew install ripgrep
  ghcup install hls
  stack install haskell-dap ghci-dap haskell-debug-adapter
  ```

- Opening a kitty terminal, in settings, I add

  ```
  font_family       JetBrainsMono Nerd Font Mono
  bold_font         JetBrainsMono Nerd Font Mono ExtraBold
  bold_italic_font  JetBrainsMono Nerd Font Mono ExtraBold Italic
  ```

Then, closing and reopening, in a kitty terminal, I set up the NvChad configuration following the suggestions used on the webpage mentioned above, here, and then tell nvim to do syntax highlighting, add lines to `init.lua`, create `haskell.lua` as described for Ubuntu, above.

Then start tmux in terminal; split screen into 2 panes; start nvim in one of them; write, edit, save code in nvim; run code in the other pane as it is developed. An alternative that avoids learning how tmux works is just to use nvim (with the NvChad configuration), opening a terminal inside nvim with ⌴h.

tmux

`<cntl>-b`	default `<prefix>`
`<cntl>-␣`	my `<prefix>`

sessions

`tmux new -s mysession1`	new mysession1
`tmux a mysession1`	attach mysession1
`tmux ls`	list sessions
`<cntl>-␣ $`	rename session
`<cntl>-␣ d`	detach from session

windows

`<cntl>-␣ c`	create new window
`<cntl>-␣ p`	previous window
`<cntl>-␣ n`	next window
`<cntl>-␣ 1...9`	window by number
`<cntl>-␣ &`	close current window

panes

`<cntl>-␣ "`	split horizontal
`<cntl>-␣ %`	split vertical
`<cntl>-␣ z`	toggle zoom
`<cntl>-␣ ←↑↓→`	switch pane (or mouse)
`<cntl>-␣ h j k l`	switch pane (or mouse)
`<cntl>-␣ !`	convert pane to window
`<cntl>-␣ x`	close current pane

neovim

motion

`0`	zero moves cursor to beginning of line
`$`	move cursor to end of line
`M`	move cursor to middle line on screen
`gg`	move to first line
`G`	move to last line
`#G`	goto line #
`#H`	goto #th line in window

general

`:u`	undo last command
`:undo`	undo last command
`:redo`	redo last command
`.`	repeat last command (in normal mode)
`:sp`	split window (close w `:q`)
`:vsp`	split window vertically
`:only`	make current window the only one
`:new file`	split and open file

save, write, quit

`:q  :q!`	**quit, quit quick**
`:w file`	write to file
`:wq`	save and quit
`<cntl>-s`	save

search

`/pattern`	search forward to pattern
`?pattern`	search backward to pattern

cut and paste

`yy`	yank 1 line
`Y`	yank 1 line
`Y$`	yank to end of line
`#Y`	yank # of lines
`p`	put before cursor
`P`	put after cursor

edit

`x`	delete symbol under cursor
`X`	delete backwards from cursor
`R`	replace to end of line
`D`	delete to end of line
`dd`	delete current line
`i a`	insert before, after cursor
`I A`	insert at beginning, end of line
`esc`	**escape insert mode**
`J`	join next line w no space
`gJ`	join next line w space
`:s/old/new/`	replace next old w new
`:s/old/new/g`	replace old w new in line
`:%s/old/new/g`	replace old w new in file
`:%s/old/new/gc`	replace with confirmations
`~`	change case of char under cursor
`vEU/u`	uppercase/lowercase word
`vU/u`	uppercase/lowercase word

insert

`:r file`	insert file here

NvChad

`␣`	my `<prefix>`

motion

`<cntl>-␣`	move cursor to next word
`<cntl>-e`	move screen forward 1 line
`<cntl>-y`	move screen forward 1 line
`<cntl>-d`	move forward $\frac{1}{2}$ screen
`<cntl>-u`	move backward $\frac{1}{2}$ screen
`<cntl>-f`	move forward $\frac{1}{2}$ screen
`<cntl>-b`	move backward $\frac{1}{2}$ screen

nvimtree

`<cntl>-n`	toggle nvimtree
`<return>`	in tree, open window on file under cursor
`a`	in tree, add new file in current dir

telescope

`␣th`	select themes
`␣fh`	find help
`␣fb`	show buffers

general

`␣ch`	**toggle cheat sheet**
`␣n`	toggle line numbers
`␣x`	close current buffer
`␣h`	**split and open terminal**

haskell

`:MasonInstallAll`	manage lsp's etc
`:checkhealth lsp`	check lsp
`:LspInfo`	check lsp
`␣rr`	split and open ghci session
`␣rq`	quit ghci session
`␣ea`	evaluate code snippets
`␣hs`	hoogle search sig under cursor
`␣cl`	code lens refresh