

with-mistral

in state of neovim lua configuration, suggest me several custom functions that are plug-and-play

awesome, try to discover more useful yet ergonomic functions

based on both previous responses, suggest me functions that are still not mentioned yet

what about more functions, but inspired by some plugins' functionality

Here are several **plug-and-play** custom Lua functions for your Neovim configuration. These functions are designed to be practical, reusable, and easy to integrate into your `init.lua` or Lua modules. Each function is self-contained and addresses common needs in Neovim workflows.

1. Toggle Line Numbers

Purpose: Toggle between relative and absolute line numbers.

```
function ToggleLineNumbers()  
  if vim.wo.relativenumber then  
    vim.wo.number = true  
    vim.wo.relativenumber = false  
  else  
    vim.wo.number = true  
    vim.wo.relativenumber = true  
  end  
end
```

Usage: Call `ToggleLineNumbers()` in a keymap or command.

2. Quick Buffer Navigation

Purpose: Navigate between buffers using `<leader>bn` and `<leader>bp`.

```
function NextBuffer()  
  vim.cmd("bnext")  
end  
  
function PrevBuffer()  
  vim.cmd("bprevious")  
end
```

Usage: Map these functions to keys in your config:

```
vim.keymap.set('n', '<leader>bn', NextBuffer, { desc = "Next buffer" })  
vim.keymap.set('n', '<leader>bp', PrevBuffer, { desc = "Previous buffer" })
```

3. Toggle Terminal

Purpose: Open or close a terminal in a new split.

```
function ToggleTerminal()
  local term_buf = vim.fn.bufnr("$TERM_BUF")
  if term_buf == -1 then
    vim.cmd("split | term")
    vim.fn.setbufvar(vim.fn.bufnr(), "term_title", "Terminal")
  else
    vim.cmd("bdelete " .. term_buf)
  end
end
```

Usage: Call `ToggleTerminal()` with a keymap.

4. Copy File Path

Purpose: Copy the current file's absolute or relative path to the clipboard.

```
function CopyFilePath(mode)
  local path = vim.fn.expand("%:" .. (mode == "relative" and "." or ":p"))
  vim.fn.setreg("+", path)
  print("Copied: " .. path)
end
```

Usage:

```
vim.keymap.set('n', '<leader>cf', function() CopyFilePath("full") end, { desc =
"Copy full file path" })
vim.keymap.set('n', '<leader>cr', function() CopyFilePath("relative") end, { desc =
"Copy relative file path" })
```

5. Toggle Spell Check

Purpose: Enable or disable spell checking.

```
function ToggleSpellCheck()
  if vim.wo.spell then
    vim.wo.spell = false
  else
    vim.wo.spell = true
  end
end
```

Usage: Map to a key or command.

6. Search and Replace in Visual Mode

Purpose: Replace the selected text in visual mode.

```
function VisualReplace()  
  local old_word = vim.fn.expand("<cword>")  
  local new_word = vim.fn.input("Replace " .. old_word .. " with: ")  
  if new_word ~= "" then  
    vim.cmd("%s/" .. old_word .. "/" .. new_word .. "/g")  
  end  
end
```

Usage: Map to a visual mode keymap:

```
vim.keymap.set('v', '<leader>r', VisualReplace, { desc = "Replace selected text" })
```

7. Toggle Wrap

Purpose: Toggle line wrapping.

```
function ToggleWrap()  
  if vim.wo.wrap then  
    vim.wo.wrap = false  
  else  
    vim.wo.wrap = true  
  end  
end
```

Usage: Map to a key or command.

8. Open Neovim Config

Purpose: Quickly open your Neovim config file.

```
function OpenConfig()  
  vim.cmd("edit ~/.config/nvim/init.lua")  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>vc', OpenConfig, { desc = "Open Neovim config" })
```

9. Clear Search Highlights

Purpose: Clear search highlights.

```
function ClearSearch()  
  vim.cmd("nohlsearch")  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>c', ClearSearch, { desc = "Clear search highlights" })
```

10. Toggle Zen Mode

Purpose: Toggle a distraction-free "zen mode" (requires [zen-mode.nvim](#)).

```
function ToggleZenMode()  
  require("zen-mode").toggle()  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>z', ToggleZenMode, { desc = "Toggle Zen Mode" })
```

Integration Tips:

- Place these functions in a Lua file (e.g., `~/.config/nvim/lua/custom/functions.lua`) and `require` them in your `init.lua`.
- Customize keymaps and descriptions to fit your workflow.

Absolutely! Here are more **ergonomic and useful** custom Lua functions for Neovim, designed to streamline your workflow and enhance productivity. Each function is plug-and-play and addresses common pain points or repetitive tasks.

1. Toggle Transparency

Purpose: Toggle the transparency of the Neovim window (useful for terminal users).

```
function ToggleTransparency()  
  if vim.g.transparency_enabled then  
    vim.cmd("hi Normal guibg=NONE ctermbg=NONE")  
    vim.g.transparency_enabled = false  
  else  
    vim.cmd("hi Normal guibg=#1e1e2e ctermbg=235")  
    vim.g.transparency_enabled = true  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tt', ToggleTransparency, { desc = "Toggle transparency" })
```

2. Quick Fix Navigation

Purpose: Navigate through quickfix list entries.

```
function NextQuickFix()  
  vim.cmd("cnext")  
end  
  
function PrevQuickFix()  
  vim.cmd("cprev")  
end
```

Usage: Map to keys:

```
vim.keymap.set('n', '<leader>qn', NextQuickFix, { desc = "Next quickfix" })  
vim.keymap.set('n', '<leader>qp', PrevQuickFix, { desc = "Previous quickfix" })
```

3. Insert Date or Time

Purpose: Insert the current date or time at the cursor position.

```
function InsertDate()  
  local date = os.date("%Y-%m-%d")  
  vim.api.nvim_put({ date }, "c", true, true)  
end  
  
function InsertTime()  
  local time = os.date("%H:%M:%S")  
  vim.api.nvim_put({ time }, "c", true, true)  
end
```

Usage: Map to keys:

```
vim.keymap.set('n', '<leader>id', InsertDate, { desc = "Insert date" })  
vim.keymap.set('n', '<leader>it', InsertTime, { desc = "Insert time" })
```

4. Toggle Indent Guides

Purpose: Toggle indent guides (requires [indent-blankline.nvim](#)).

```
function ToggleIndentGuides()  
  local ok, blankline = pcall(require, "indent_blankline")  
  if ok then  
    if vim.g.indent_blankline_enabled then  
      blankline.disable()  
      vim.g.indent_blankline_enabled = false  
    else  
      blankline.enable()  
      vim.g.indent_blankline_enabled = true  
    end  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>ti', ToggleIndentGuides, { desc = "Toggle indent guides" })
```

5. Open GitHub Repository

Purpose: Open the current file's GitHub repository in the browser (if in a Git repo).

```
function OpenGitHubRepo()
  local git_root = vim.fn.system("git rev-parse --show-toplevel 2>/dev/null | tr -d '\n'")
  if git_root ~= "" then
    local remote = vim.fn.system("git remote get-url origin 2>/dev/null | tr -d '\n'")
    if remote:match("github%.com") then
      local repo_url = remote:gsub("git@github%.com:", "https://github.com/"):gsub("%%.git$", "")
      vim.fn.jobstart("open " .. repo_url, { detach = true })
    end
  end
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>gh', OpenGitHubRepo, { desc = "Open GitHub repo" })
```

6. Toggle Diagnostic Virtual Text

Purpose: Toggle LSP diagnostic virtual text.

```
function ToggleDiagnosticVirtualText()
  if vim.diagnostic.config().virtual_text then
    vim.diagnostic.config({ virtual_text = false })
  else
    vim.diagnostic.config({ virtual_text = true })
  end
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>td', ToggleDiagnosticVirtualText, { desc = "Toggle diagnostics" })
```

7. Quickly Resize Splits

Purpose: Resize splits with arrow keys.

```
function ResizeSplitLeft()
  vim.cmd("vertical resize -2")
end

function ResizeSplitRight()
  vim.cmd("vertical resize +2")
end

function ResizeSplitUp()
  vim.cmd("resize -2")
end

function ResizeSplitDown()
  vim.cmd("resize +2")
end
```

Usage: Map to keys:

```
vim.keymap.set('n', '<C-Left>', ResizeSplitLeft, { desc = "Resize split left" })
vim.keymap.set('n', '<C-Right>', ResizeSplitRight, { desc = "Resize split right" })
vim.keymap.set('n', '<C-Up>', ResizeSplitUp, { desc = "Resize split up" })
vim.keymap.set('n', '<C-Down>', ResizeSplitDown, { desc = "Resize split down" })
```

8. Toggle Color Column

Purpose: Toggle the color column (e.g., for line length limits).

```
function ToggleColorColumn()
  if vim.wo.colorcolumn == "" then
    vim.wo.colorcolumn = "80"
  else
    vim.wo.colorcolumn = ""
  end
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tc', ToggleColorColumn, { desc = "Toggle color column"
})
```

9. Quickly Open Recent Files

Purpose: Open a Telescope picker for recent files.

```
function OpenRecentFiles()
  require("telescope.builtin").oldfiles()
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>fr', OpenRecentFiles, { desc = "Open recent files" })
```

10. Toggle Cursor Line Highlight

Purpose: Toggle the highlight of the current line.

```
function ToggleCursorLine()  
  if vim.wo.cursorline then  
    vim.wo.cursorline = false  
  else  
    vim.wo.cursorline = true  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tl', ToggleCursorLine, { desc = "Toggle cursor line" })
```

11. Quickly Open Terminal in Current Directory

Purpose: Open a terminal in the current file's directory.

```
function OpenTerminalInDir()  
  local dir = vim.fn.expand("%:p:h")  
  vim.cmd("split | term cd " .. dir)  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>td', OpenTerminalInDir, { desc = "Open terminal in dir"  
  })
```

12. Toggle Sign Column

Purpose: Toggle the sign column (e.g., for Git signs).

```
function ToggleSignColumn()  
  if vim.wo.signcolumn == "yes" then  
    vim.wo.signcolumn = "no"  
  else  
    vim.wo.signcolumn = "yes"  
  end  
end
```

Usage: Map to a key:


```
vim.keymap.set('n', '<leader>ts', ToggleSignColumn, { desc = "Toggle sign column" })
```

13. Quickly Open File in Default Application

Purpose: Open the current file in the default system application.

```
function OpenFileInDefaultApp()  
  local file = vim.fn.expand("%:p")  
  vim.fn.jobstart("open " .. file, { detach = true })  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>od', OpenFileInDefaultApp, { desc = "Open file in  
default app" })
```

14. Toggle Fold Column

Purpose: Toggle the fold column.

```
function ToggleFoldColumn()  
  if vim.wo.foldcolumn == "0" then  
    vim.wo.foldcolumn = "1"  
  else  
    vim.wo.foldcolumn = "0"  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tf', ToggleFoldColumn, { desc = "Toggle fold column" })
```

15. Quickly Open Neovim's Lazy Plugin Manager

Purpose: Open [lazy.nvim](https://github.com/folke/lazy.nvim) to manage plugins.

```
function OpenLazy()  
  vim.cmd("Lazy")  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>pl', OpenLazy, { desc = "Open Lazy plugin manager" })
```

Integration Tips:

- **Group related functions** in a Lua module (e.g., `~/ .config/nvim/lua/custom/utils.lua`) and `require` them in your `init.lua` .
- **Customize keymaps** to avoid conflicts and match your workflow.
- **Explore plugins** like [Telescope](#), [WhichKey](#), and [Lualine](#) to enhance these functions.

Here are **fresh, ergonomic, and highly practical** Neovim Lua functions that haven't been mentioned yet. These are designed to address niche but common workflow needs, improve quality of life, and integrate seamlessly into your configuration.

1. Toggle Inlay Hints

Purpose: Toggle LSP inlay hints (e.g., type hints, parameter names) for supported languages.

```
function ToggleInlayHints()  
  local ok, inlay = pcall(vim.lsp.inlay_hint, 0)  
  if not ok then  
    vim.notify("Inlay hints not supported for this LSP.", vim.log.levels.WARN)  
  else  
    vim.lsp.inlay_hint.enable(not vim.lsp.inlay_hint.is_enabled({ bufnr = 0 }))  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>th', ToggleInlayHints, { desc = "Toggle inlay hints" })
```

2. Quickly Jump to LSP Definition in Split

Purpose: Open the LSP definition in a vertical split instead of the current window.

```
function LspDefinitionSplit()  
  vim.cmd("vsplit")  
  vim.lsp.buf.definition()  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', 'gd', LspDefinitionSplit, { desc = "Go to definition (split)" })
```

3. Toggle Auto-Format on Save

Purpose: Toggle automatic formatting on save for the current buffer.

```
function ToggleAutoFormat()  
  local buf = vim.api.nvim_get_current_buf()  
  if vim.b[buf].auto_format == false then  
    vim.b[buf].auto_format = true  
    vim.notify("Auto-format enabled for this buffer.", vim.log.levels.INFO)  
  end  
end
```

```

else
    vim.b[buf].auto_format = false
    vim.notify("Auto-format disabled for this buffer.", vim.log.levels.INFO)
end
end

```

Usage: Map to a key and integrate with your `autocmd` for `BufWritePre` .

4. Quickly Insert a Code Block

Purpose: Insert a pre-defined code block (e.g., for React, Python, or Lua) at the cursor.

```

function InsertCodeBlock(lang)
    local blocks = {
        react = {
            "import React from 'react';",
            "",
            "const [state, setState] = React.useState();",
            "",
            "export default function () {",
            "    return (",
            "        <div>",
            "            ",
            "        </div>",
            "    );",
            "}",
        },
        python = {
            "def main():",
            "    pass",
            "",
            "if __name__ == '__main__':",
            "    main()",
        },
        lua = {
            "local function main()",
            "    -- TODO: Implement",
            "end",
            "",
            "main()",
        },
    }
    if blocks[lang] then
        vim.api.nvim_put(blocks[lang], "l", true, true)
    end
end

```

Usage: Map to keys:

```

vim.keymap.set('n', '<leader>ir', function() InsertCodeBlock("react") end, { desc =
    "Insert React block" })

```

```
vim.keymap.set('n', '<leader>ip', function() InsertCodeBlock("python") end, { desc =  
"Insert Python block" })  
vim.keymap.set('n', '<leader>il', function() InsertCodeBlock("lua") end, { desc =  
"Insert Lua block" })
```

5. Toggle Background Color Scheme

Purpose: Toggle between light and dark color schemes.

```
function ToggleBackground()  
  if vim.o.background == "dark" then  
    vim.o.background = "light"  
  else  
    vim.o.background = "dark"  
  end  
  vim.cmd("colorscheme " .. vim.g.colors_name)  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tb', ToggleBackground, { desc = "Toggle background" })
```

6. Quickly Open Neovim's Health Check

Purpose: Open Neovim's `:checkhealth` in a floating window.

```
function OpenHealthCheck()  
  vim.cmd("enew | setlocal buftype=nofile bufhidden=wipe noswapfile")  
  vim.fn.termopen("nvim --headless -c 'checkhealth' -c 'qa!'")  
  vim.cmd("normal! G")  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>ch', OpenHealthCheck, { desc = "Open health check" })
```

7. Toggle Conceal Level

Purpose: Toggle conceal level for Markdown or other filetypes.

```
function ToggleConceal()  
  if vim.o.conceallevel == 0 then  
    vim.o.conceallevel = 2  
  else  
    vim.o.conceallevel = 0  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tc', ToggleConceal, { desc = "Toggle conceal" })
```

8. Quickly Open Neovim's Log File

Purpose: Open Neovim's log file for debugging.

```
function OpenLogFile()  
  local log_path = vim.fn.stdpath("log") .. "/nvim.log"  
  vim.cmd("edit " .. log_path)  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>n1', OpenLogFile, { desc = "Open Neovim log" })
```

9. Toggle List Characters

Purpose: Toggle the display of whitespace characters (tabs, trailing spaces, etc.).

```
function ToggleListChars()  
  if vim.o.list then  
    vim.o.list = false  
  else  
    vim.o.list = true  
    vim.o.listchars = "tab:↪ ,trail:·,nbsp:␣"  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>t1', ToggleListChars, { desc = "Toggle list chars" })
```

10. Quickly Open Neovim's Runtime Directory

Purpose: Open Neovim's runtime directory (e.g., for plugins, colorschemes).

```
function OpenRuntimeDir()  
  local runtime_path = vim.fn.stdpath("config")  
  vim.cmd("edit " .. runtime_path)  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>nr', OpenRuntimeDir, { desc = "Open runtime dir" })
```

11. Toggle Diff Mode

Purpose: Toggle diff mode for the current buffer (useful for Git diffs).

```
function ToggleDiffMode()  
  if vim.wo.diff then  
    vim.wo.diff = false  
  else  
    vim.wo.diff = true  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>td', ToggleDiffMode, { desc = "Toggle diff mode" })
```

12. Quickly Open Neovim's Man Page

Purpose: Open Neovim's man page for a given topic.

```
function OpenManPage(topic)  
  vim.cmd("Man " .. topic)  
end
```

Usage: Map to a key or use in command mode:

```
vim.keymap.set('n', '<leader>mn', function() OpenManPage(vim.fn.input("Man page: "))  
end, { desc = "Open man page" })
```

13. Toggle Mouse Support

Purpose: Toggle mouse support in Neovim.

```
function ToggleMouse()  
  if vim.o.mouse == "" then  
    vim.o.mouse = "a"  
  else  
    vim.o.mouse = ""  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tm', ToggleMouse, { desc = "Toggle mouse" })
```

14. Quickly Open Neovim's API Documentation

Purpose: Open Neovim's Lua API documentation in the browser.

```
function OpenAPIDocs()  
  vim.fn.jobstart("open https://neovim.io/doc/user/api.html", { detach = true })  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>ad', OpenAPIDocs, { desc = "Open API docs" })
```

15. Toggle Paste Mode

Purpose: Toggle paste mode to avoid auto-indent issues when pasting.

```
function TogglePasteMode()  
  if vim.o.paste then  
    vim.o.paste = false  
  else  
    vim.o.paste = true  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tp', TogglePasteMode, { desc = "Toggle paste mode" })
```

16. Quickly Open Neovim's FAQ

Purpose: Open Neovim's FAQ in the browser.

```
function OpenFAQ()  
  vim.fn.jobstart("open https://github.com/neovim/neovim/wiki/FAQ", { detach = true  
  })  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>af', OpenFAQ, { desc = "Open FAQ" })
```

17. Toggle Virtual Edit Mode

Purpose: Toggle virtual edit mode (allow cursor to move beyond end of line).

```
function ToggleVirtualEdit()  
  if vim.o.virtualedit == "all" then  
    vim.o.virtualedit = ""  
  else
```

```
vim.o.virtualedit = "all"  
end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tv', ToggleVirtualEdit, { desc = "Toggle virtual edit"  
})
```

18. Quickly Open Neovim's Issue Tracker

Purpose: Open Neovim's GitHub issue tracker in the browser.

```
function OpenIssueTracker()  
  vim.fn.jobstart("open https://github.com/neovim/neovim/issues", { detach = true })  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>ai', OpenIssueTracker, { desc = "Open issue tracker" })
```

19. Toggle Line Wrapping with Word Break

Purpose: Toggle line wrapping with word break for long lines.

```
function ToggleWrapBreak()  
  if vim.o.wrap and vim.o.linebreak then  
    vim.o.wrap = false  
  else  
    vim.o.wrap = true  
    vim.o.linebreak = true  
  end  
end
```

Usage: Map to a key:

```
vim.keymap.set('n', '<leader>tw', ToggleWrapBreak, { desc = "Toggle wrap/break" })
```

20. Quickly Open Neovim's Wiki

Purpose: Open Neovim's community wiki in the browser.

```
function OpenWiki()  
  vim.fn.jobstart("open https://github.com/neovim/neovim/wiki", { detach = true })  
end
```

Usage: Map to a key:


```
vim.keymap.set('n', '<leader>aw', OpenWiki, { desc = "Open wiki" })
```

Integration Tips:

- **Group functions** by category (e.g., UI, LSP, Git) in separate Lua files for clarity.
- **Use `vim.notify`** for user feedback when toggling features.
- **Combine with plugins** like [WhichKey](#) to display available keymaps.

Great idea! Below are **custom Lua functions inspired by popular Neovim plugins**, designed to replicate or enhance their core functionality without requiring the full plugin. These are plug-and-play, ergonomic, and integrate seamlessly into your workflow.

1. Harpoon-like Quick File Navigation

Inspiration: [Harpoon](#) **Purpose:** Quickly navigate between frequently used files.

```
local harpoon_files = {}

function HarpoonAddFile()
  local file = vim.fn.expand("%:p")
  table.insert(harpoon_files, file)
  vim.notify("Added: " .. file, vim.log.levels.INFO)
end

function HarpoonNavigate(index)
  if harpoon_files[index] then
    vim.cmd("edit " .. harpoon_files[index])
  else
    vim.notify("No file at index " .. index, vim.log.levels.WARN)
  end
end
```

Usage:

```
vim.keymap.set('n', '<leader>ha', HarpoonAddFile, { desc = "Harpoon: Add file" })
vim.keymap.set('n', '<leader>h1', function() HarpoonNavigate(1) end, { desc =
"Harpoon: Navigate to 1" })
vim.keymap.set('n', '<leader>h2', function() HarpoonNavigate(2) end, { desc =
"Harpoon: Navigate to 2" })
-- Add more as needed
```

2. Telescope-like Fuzzy Finder for Buffers

Inspiration: [Telescope](#) **Purpose:** Fuzzy find and switch between open buffers.

```
function FuzzyFindBuffers()
  local buffers = vim.fn.getbufinfo({ buflisted = 1 })
  local names = {}
  for _, buf in ipairs(buffers) do
```

```

        table.insert(names, buf.name)
    end
    vim.ui.select(names, {
        prompt = "Select buffer:",
    }, function(choice)
        if choice then
            vim.cmd("edit " .. choice)
        end
    end)
end
end

```

Usage:

```
vim.keymap.set('n', '<leader>fb', FuzzyFindBuffers, { desc = "Fuzzy find buffers" })
```

3. Comment Toggle (Like Comment.nvim)

Inspiration: [Comment.nvim](#) **Purpose:** Toggle comments for the current line or selection.

```

function ToggleComment()
    local ft = vim.bo.filetype
    local comment_strings = {
        lua = "-- ",
        python = "# ",
        javascript = "// ",
        rust = "// ",
        c = "// ",
        cpp = "// ",
        sh = "# ",
    }
    local cs = comment_strings[ft] or "# "
    local line = vim.fn.line(".")
    local lines = vim.fn.getline(line, line)
    if lines[1]:match("^%s*" .. cs) then
        -- Uncomment
        vim.cmd("s/\\v^\\s*" .. vim.fn.escape(cs, "/") .. "//e")
    else
        -- Comment
        vim.cmd("s/^/" .. cs .. "/")
    end
end
end

```

Usage:

```

vim.keymap.set('n', '<leader>/', ToggleComment, { desc = "Toggle comment" })
vim.keymap.set('v', '<leader>/', ":<C-u>lua ToggleComment()<CR>", { desc = "Toggle comment (visual)" })

```

4. Auto-Pairs (Like nvim-autopairs)

Inspiration: [nvim-autopairs](#) **Purpose:** Automatically close pairs (e.g., `()`, `[]`, `{}`, `"`).

```
function AutoPair(char)
  local pairs = { ["("] = ")", ["["] = "]", [{""] = "}"}, [{"'"] = "'", ["""] = "" }
  local close_char = pairs[char]
  if close_char then
    vim.api.nvim_put({ close_char }, "c", false, true)
    vim.cmd("normal! h")
  end
end
```

Usage: Bind to `InsertCharPre` autocmd:

```
vim.api.nvim_create_autocmd("InsertCharPre", {
  callback = function()
    local char = vim.fn.nr2char(vim.fn.getchar())
    AutoPair(char)
  end,
})
```

5. Surround Text (Like nvim-surround)

Inspiration: [nvim-surround](#) **Purpose:** Add, change, or delete surrounding delimiters (e.g., `"` → `"text"`).

```
function SurroundAdd(delimiter)
  local esc = vim.fn.escape(delimiter, "/\\")
  vim.cmd("normal! `>a" .. esc .. "<ESC>`<i" .. esc .. "<ESC>")
end

function SurroundChange(old, new)
  local esc_old = vim.fn.escape(old, "/\\")
  local esc_new = vim.fn.escape(new, "/\\")
  vim.cmd("%s/" .. esc_old .. "\\(. *\\)" .. esc_old .. "/" .. esc_new .. "\\1" ..
  esc_new .. "/g")
end
```

Usage:

```
vim.keymap.set('v', '<leader>sa', [[:lua SurroundAdd(vim.fn.input("Delimiter: "))
<CR>]], { desc = "Surround: Add" })
vim.keymap.set('n', '<leader>sc', [[:lua SurroundChange(vim.fn.input("Old: "),
vim.fn.input("New: "))<CR>]], { desc = "Surround: Change" })
```

6. Git Signs (Like gitsigns.nvim)

Inspiration: [gitsigns.nvim](#) **Purpose:** Show Git diff signs in the sign column.

```
function ShowGitSigns()
  vim.cmd("sign define GitSignAdd text=+ texthl=GitSignsAdd")
  vim.cmd("sign define GitSignChange text=~ texthl=GitSignsChange")
  vim.cmd("sign define GitSignDelete text=_ texthl=GitSignsDelete")
  vim.cmd("sign place 100 line=1 name=GitSignAdd file=" .. vim.fn.expand("%:p"))
end
```

Usage: Call this function in a `BufEnter` autocmd for Git files.

7. LSP Signature Help (Like `lspsaga.nvim`)

Inspiration: [lspsaga.nvim](#) **Purpose:** Show LSP signature help in a floating window.

```
function ShowSignatureHelp()
  vim.lsp.buf.signature_help()
end
```

Usage:

```
vim.keymap.set('n', '<leader>ls', ShowSignatureHelp, { desc = "LSP: Signature help"
})
```

8. Quickly Open Terminal in Project Root

Inspiration: [toggleterm.nvim](#) **Purpose:** Open a terminal in the project root directory.

```
function OpenTerminalInRoot()
  local root = vim.fn.system("git rev-parse --show-toplevel 2>/dev/null | tr -d
'\n'")
  if root == "" then
    root = vim.fn.getcwd()
  end
  vim.cmd("split | term cd " .. root)
end
```

Usage:

```
vim.keymap.set('n', '<leader>tr', OpenTerminalInRoot, { desc = "Terminal: Open in
root" })
```

9. Toggle LSP Diagnostics (Like `trouble.nvim`)

Inspiration: [trouble.nvim](#) **Purpose:** Toggle a quickfix list of LSP diagnostics.

```
function ToggleDiagnostics()
  vim.diagnostic.setqflist({ open = not vim.fn.getqflist({ size = 1 }).size })
end
```

Usage:

```
vim.keymap.set('n', '<leader>td', ToggleDiagnostics, { desc = "Toggle diagnostics"
})
```

10. Quickly Open Neovim's Config in a New Tab

Inspiration: [nvim-tree.lua](#) **Purpose:** Open your Neovim config in a new tab.

```
function OpenConfigInTab()
  vim.cmd("tabnew ~/.config/nvim/init.lua")
end
```

Usage:

```
vim.keymap.set('n', '<leader>ct', OpenConfigInTab, { desc = "Open config in tab" })
```

11. Toggle LSP Semantic Tokens

Inspiration: [nvim-semantic-tokens](#) **Purpose:** Toggle LSP semantic tokens for syntax highlighting.

```
function ToggleSemanticTokens()
  local clients = vim.lsp.get_active_clients({ bufnr = 0 })
  for _, client in ipairs(clients) do
    if client.supports_method("textDocument/semanticTokens") then
      client.server_capabilities.semanticTokensProvider = not
client.server_capabilities.semanticTokensProvider
      vim.cmd("LspRestart")
    end
  end
end
```

Usage:

```
vim.keymap.set('n', '<leader>ts', ToggleSemanticTokens, { desc = "Toggle semantic
tokens" })
```

12. Quickly Open Neovim's Plugin Manager UI

Inspiration: [lazy.nvim](#) **Purpose:** Open your plugin manager UI (e.g., Lazy, Packer).

```
function OpenPluginManager()
  vim.cmd("Lazy")
end
```

Usage:

```
vim.keymap.set('n', '<leader>pm', OpenPluginManager, { desc = "Open plugin manager"
})
```

13. Toggle LSP Code Lens

Inspiration: [lsp-lens.nvim](#) **Purpose:** Toggle LSP code lens (e.g., references, implementations).

```
function ToggleCodeLens()
  vim.lsp.codelens.refresh()
  if vim.lsp.codelens.is_enabled() then
    vim.lsp.codelens.clear()
  else
    vim.lsp.codelens.display()
  end
end
```

Usage:

```
vim.keymap.set('n', '<leader>cl', ToggleCodeLens, { desc = "Toggle code lens" })
```

14. Quickly Open Neovim's Help in a Vertical Split

Inspiration: [nvim-helpview](#) **Purpose:** Open Neovim's help in a vertical split.

```
function OpenHelpSplit(topic)
  vim.cmd("vsplit | help " .. topic)
end
```

Usage:

```
vim.keymap.set('n', '<leader>hs', function() OpenHelpSplit(vim.fn.input("Help topic:
")) end, { desc = "Open help (split)" })
```

15. Toggle LSP Inlay Hints for Current Buffer

Inspiration: [lsp-inlayhints.nvim](#) **Purpose:** Toggle LSP inlay hints for the current buffer.

```
function ToggleInlayHintsBuf()
  local buf = vim.api.nvim_get_current_buf()
  if vim.lsp.inlay_hint.is_enabled({ bufnr = buf }) then
    vim.lsp.inlay_hint.enable(buf, false)
  else
    vim.lsp.inlay_hint.enable(buf, true)
  end
end
```

Usage:

```
vim.keymap.set('n', '<leader>ih', ToggleInlayHintsBuf, { desc = "Toggle inlay hints (buffer)" })
```

Integration Tips:

- **Combine with** `vim.ui.select` for interactive selection (e.g., buffers, files).
 - **Use** `vim.notify` for user feedback.
 - **Map to intuitive keymaps** and use [WhichKey](#) for discoverability.
-