# Contents

local wk = require(“which-key”)

vim.cmd([[ function! PandocPDFBib() silent exec “Dispatch pandoc –pdf-engine=xelatex –variable mainfont=”Palatino” –variable sansfont=”Helvetica” –variable monofont=”Menlo” –variable fontsize=10pt –variable version=2.0 –toc –toc-depth=2 -V geometry:margin=2cm –bibliography=references.bib -o” . expand(“%:r”). “.pdf -s” . expand(“%”) . ” && open ” . expand(“%:r”) . “.pdf&” endfunction]])

vim.cmd([[ function! PandocPDF() silent exec “Dispatch pandoc –pdf-engine=xelatex –variable mainfont=”Palatino” –variable sansfont=”Helvetica” –variable monofont=”Menlo” –variable fontsize=10pt –variable version=2.0 –toc –toc-depth=2 -V geometry:margin=2cm -o” . expand(“%:r”). “.pdf -s” . expand(“%”) . ” && open ” . expand(“%:r”) . “.pdf&” endfunction]])

vim.cmd([[ function! OrganizeImports() silent exec PyrightOrganizeImports silent exec “!isort %” silent exec “!autoflake –remove-all-unused-imports –remove-unused-variables –ignore-init-module-imports –in-place –exclude ‘**init**.py’ %s” endfunction]])

vim.cmd([[ function! PandocPDFNoContents() silent exec “Dispatch pandoc –pdf-engine=xelatex –variable mainfont=”Palatino” –variable sansfont=”Helvetica” –variable monofont=”Menlo” –variable fontsize=10pt –variable version=2.0 -V geometry:margin=2cm -o” . expand(“%:r”). “.pdf -s” . expand(“%”) . ” && open ” . expand(“%:r”) . “.pdf&” endfunction]])

vim.cmd([[ function! PandocHTML() silent exec “Dispatch pandoc -o” . expand(“%:r”). “.html -s” . expand(“%”) . “; open” . expand(“%:r”) . “.html” endfunction]])

vim.cmd([[ function! PublishMedium() silent exec “Dispatch pandoc -o” . expand(“%:r”). “.html -s” . expand(“%”) . “; ~/go/bin/md-publisher publish –medium-token `cat $HOME/bin/.medium_token`” . expand(“%:r”) . “.html”]])

vim.cmd([[ function! PandocPDFLandscape() silent exec “Dispatch pandoc –pdf-engine=xelatex –variable mainfont=”Palatino” –variable sansfont=”Helvetica” –variable monofont=”Menlo” –variable fontsize=10pt –variable version=2.0 –toc –toc-depth=2 -V geometry:margin=1cm -V geometry:landscape -o” . expand(“%:r”). “.pdf -s” . expand(“%”) . ” && open ” . expand(“%:r”) . “.pdf&” endfunction]])

vim.cmd([[ function! PandocPDFLandscapeNoContents() silent exec “Dispatch pandoc –pdf-engine=xelatex –variable mainfont=”Palatino” –variable sansfont=”Helvetica” –variable monofont=”Menlo” –variable fontsize=10pt –variable version=2.0 -V geometry:margin=1cm -V geometry:landscape -o” . expand(“%:r”). “.pdf -s” . expand(“%”) . ” && open ” . expand(“%:r”) . “.pdf&” endfunction]])

vim.cmd([[ function! ToggleFileFold() if &foldlevel == 0 norm zR else norm zM endif endfunction]]) vim.cmd([[ function! SetupWrappingSoft() if &wrap == 1 set nowrap set textwidth=0 nnoremap j j nnoremap k k nnoremap 0 0 nnoremap ^ ^ nnoremap $ $ else set wrap nnoremap j gj nnoremap k gk nnoremap 0 g0 nnoremap ^ g^ nnoremap $ g$ endif endfunction]])

vim.cmd([[ function! SetupWrappingHard() if &wrap == 1 set nowrap set textwidth=0 nnoremap j j nnoremap k k else set wrap set wm=2 set textwidth=79 nnoremap j gj nnoremap k gk endif endfunction]]) vim.cmd([[ function! SetBackground() if &background == ‘dark’ ” if &colors != ‘gruvbox’ ” execute “set colors=gruvbox” ” endif execute “set background=light” else execute “set background=dark” endif endfunction]])

wk.setup {} wk.register({ [“”] = { “Alpha”, “Home” }, [“E”] = { “NvimTreeToggle”, “explorer on root” }, [“e”] = { “NvimTreeFindFileToggle”, “explorer” },

```
["f"] = { "<cmd>lua telescope_files_or_git_files()<cr>", "files/git" },
[";"] = { "<cmd>Telescope file_browser<cr>", "file browser" },
["h"] = { "<cmd>let @/ = ''<cr>", "no highlight" },
["w"] = { "<cmd>wa<cr>", "save all" },
["q"] = { "<C-W>c", "quit window" },
["-"] = { "<C-W>s", "split below" },
["\\"] = { "<C-W>v", "split right" },
["="] = { "<C-W>=", "balance windows" },
["/"] = { "<cmd>FzfLua grep_curbuf<cr>", "fuzzy search buffer" },
["u"] = { "<cmd>UndotreeToggle<cr>", "undo tree" },
```

```
["<space>"] = { "<cmd>SnipRun<cr>", "run" },
["1"] = { "<Cmd>BufferLineGoToBuffer 1<CR>", "Buf 1" },
["2"] = { "<Cmd>BufferLineGoToBuffer 2<CR>", "Buf 2" },
["3"] = { "<Cmd>BufferLineGoToBuffer 3<CR>", "Buf 2" },
["4"] = { "<Cmd>BufferLineGoToBuffer 4<CR>", "Buf 2" },
["5"] = { "<Cmd>BufferLineGoToBuffer 5<CR>", "Buf 2" },
["6"] = { "<Cmd>BufferLineGoToBuffer 6<CR>", "Buf 2" },
["7"] = { "<Cmd>BufferLineGoToBuffer 7<CR>", "Buf 2" },
["8"] = { "<Cmd>BufferLineGoToBuffer 8<CR>", "Buf 2" },
["9"] = { "<Cmd>BufferLineGoToBuffer 9<CR>", "Buf 2" },
["z"] = { "<C-W>m", "zoom window" },
j = {
    name = "+jupyter",

    s = { "<cmd>call StartPyShell()<cr>", "start" },
    S = { "<cmd>call StopPyShell()<cr>", "stop" },
    l = { "<cmd>call PyShellSendLine()<cr>", "line" },
    c = { "<cmd>call RunTmuxPythonCell()<cr>", "cell" },
    a = { "<cmd>call RunTmuxPythonAllCellsAbove()<cr>", "cell" },
    b = { "<cmd>call RunTmuxPythonAllCellsBelow()<cr>", "cell" },
},
m = {
    name = "+markdown",
    c = { "<cmd>call PandocPDF()<cr>", "compile (portrait)" },
    h = { "<cmd>call PandocHTML()<cr>", "compile (html)" },
    P = { "<cmd>call PublishMedium()<cr>", "publish (medium)" },
    l = { "<cmd>call PandocPDFLandscape()<cr>", "compile (landscape)" },
    v = { "<cmd>MarkdownPreview<cr>", "preview" },
    p = { "<cmd>MarkdownPreview<cr>", "preview" },
    C = {
        "<cmd>call PandocPDFNoContents()<cr>",
        "compile (portrait - no contents)"
    },
    L = {
        "<cmd>call PandocPDFLandscapeNoContents()<cr>",
        "compile (landscape - no contents)"
    }
},
b = {
    name = "+buffer",
    [">"] = { "<cmd>bnext<cr>", "next" },
    ["<"] = { "<cmd>bprevious<cr>", "prev" },
    ["$"] = { "<cmd>blast<cr>", "last" },
    ["^"] = { "<cmd>bfirst<cr>", "first" },
    b = { "<cmd>Telescope buffers<cr>", "pick buffer" },
    d = { "<cmd>Bdelete<cr>", "delete-buffer" },
    n = { "<cmd>bnext<cr>", "next-buffer" },
    p = { "<cmd>bprevious<cr>", "previous-buffer" },
    o = { "<cmd>BufOnly<cr>", "close others" }
},
d = {
    name = "+debug",
    t = { "<cmd>lua require('dapui').toggle()<cr>", "toggle breakpoint" },
    e = { "<Cmd>lua require('dapui').eval()<CR>", "evaluate" },
    b = { "<cmd>DebugToggleBreakpoint<cr>", "toggle breakpoint" },
    l = { "<cmd>DebugStepInto<cr>", "step into" },
```

```
    h = { "<cmd>DebugStepOut<cr>", "step out" },
    j = { "<cmd>DebugStepOver<cr>", "step over" },
    r = { "<cmd>DebugToggleRepl<cr>", "toggle repl" },
    g = { "<cmd>DebugStart<cr>", "start/continue" },
    k = { "<cmd>DebugHover<cr>", "hover" },
    s = { "<cmd>DebugScopes<cr>", "scopes" },
    v = { "<cmd>DebugVHover<cr>", "visual-hover" },
    z = { "<cmd>DebugLast<cr>", "last" },
    p = { "<cmd>DebugPause<cr>", "pause" },
    u = { "<cmd>DebugUp<cr>", "up" },
    d = { "<cmd>DebugDown<cr>", "down" },
    D = {
        "<cmd>DebugSetExceptionBreakpointsDefault<cr>", "exceptions-default"
    },
    N = { "<cmd>DebugSetExceptionBreakpointsNone<cr>", "exceptions-none" },
    R = { "<cmd>DebugSetExceptionBreakpointsRaised<cr>", "exceptions-raised" },
    U = {
        "<cmd>DebugSetExceptionBreakpointsUncaught<cr>",
        "exceptions-uncaught"
    },
    L = { "<cmd>DebugLogPoint<cr>", "log" },
    C = { "<cmd>DebugConditionalBreakpoint<cr>", "conditional breakpoint" },
},
F = {
    name = "+fold",
    O = { "<cmd>set foldlevel=20<cr>", "open all" },
    C = { "<cmd>set foldlevel=0<cr>", "close all" },
    c = { "<cmd>foldclose<cr>", "close" },
    o = { "<cmd>foldopen<cr>", "open" },
    ["1"] = { "<cmd>set foldlevel=1<cr>", "level1" },
    ["2"] = { "<cmd>set foldlevel=2<cr>", "level2" },
    ["3"] = { "<cmd>set foldlevel=3<cr>", "level3" },
    ["4"] = { "<cmd>set foldlevel=4<cr>", "level4" },
    ["5"] = { "<cmd>set foldlevel=5<cr>", "level5" },
    ["6"] = { "<cmd>set foldlevel=6<cr>", "level6" }
},

s = {
    name = "+search",
   ["."] = { "<cmd>lua require\"telescope.builtin\".find_files({ hidden = true })<CR>", "files (hidden)" },
    [";"] = { "<cmd>Telescope filetypes<cr>", "filetypes" },
    B = { "<cmd>Telescope git_branches<cr>", "git branches" },
    d = {
        "<cmd>Telescope diagnostics bufnr=0<cr>",
        "document diagnostics"
    },
    D = {
        "<cmd>Telescope lsp_workspace_diagnostics<cr>",
        "workspace diagnostics"
    },
    f = { "<cmd>Telescope find_files<cr>", "files" },
    ["*"] = { "<cmd>Telescope grep_string<cr>", "search current word" },
    F = { "<cmd>Telescope file_browser<cr>", "file browser" },
    ["/"] = { "<cmd>Telescope command_history<cr>", "history" },
    h = { "<cmd>Telescope git_bcommits<cr>", "file history" },
    i = { "<cmd>Telescope media_files<cr>", "media files" },
```

```
    m = { "<cmd>Telescope marks<cr>", "marks" },
    M = { "<cmd>Telescope man_pages<cr>", "man_pages" },
    o = { "<cmd>Telescope vim_options<cr>", "vim_options" },
    t = { "<cmd>Telescope live_grep<cr>", "text" },
      z = { "<cmd>lua require'telescope.builtin'.live_grep{ vimgrep_arguments = { 'rg', '--
color=never', '--no-heading', '--with-filename', '--line-number', '--column', '--smart-case', '-
u', '--no-ignore', '--hidden' } }<cr>",
        "test (everywhere)" },
    r = { "<cmd>Telescope registers<cr>", "registers" },
    c = { "<cmd>Telescope colorscheme<cr>", "colors" },
    v = { "<cmd>TelescopeVim<cr>", "vim-config" },
    l = { "<cmd>TelescopeLatex<cr>", "Latex Symbols" },
    e = { "<cmd>TelescopeEmoji<cr>", "Emoji Symbols" },
    H = { "<cmd>TelescopeHelp<cr>", "help" },
    w = { "<cmd>TelescopeWord<cr>", "word" },
    s = { "<cmd>CtrlSF<cr>", "CtrlSF" },
    S = { ":CtrlSF ", "CtrlSF" },
    a = { "<cmd>Telescope find_files<cr>", "find files" },
    P = { "<cmd>Telescope project<cr>", "projects" },
    p = { "<cmd>Telescope projects<cr>", "recent projects" },
},

S = {
    name = "+Session",
    s = { "<cmd>SessionSave", "save session" },
    l = { "<cmd>DSessionLoad<cr>", "load Session" }
},
g = {
    name = "+git",
    a = { "<cmd>Git add .<cr>", "add all" },
    A = { "<cmd>Git add %<cr>", "add current" },
    b = { "<cmd>Git blame<cr>", "blame" },
    B = { "<cmd>GBrowse<cr>", "browse" },
    c = { "<cmd>Git commit<cr>", "commit" },
    d = { "<cmd>Git diff<cr>", "diff" },
    D = { "<cmd>Gdiffsplit<cr>", "diff split" },
    G = { "<cmd>GGrep<cr>", "git grep" },
    g = { "<cmd>Git<cr>", "status" },
    i = { "<cmd>!git init<cr>", "init" },
    l = { "<cmd>Git log<cr>", "log" },
    p = { "<cmd>Git push<cr>", "push" },
    P = { "<cmd>Git pull<cr>", "pull" },
    r = { "<cmd>GRemove<cr>", "remove" },
    S = { "<cmd>!git status<cr>", "status" },
    v = { "<cmd>GV<cr>", "view commits" },
    V = { "<cmd>GV!<cr>", "view buffer commits" },
    m = { "<cmd>MergetoolToggle<cr>", "mergetool" },
    X = { "<cmd>!git checkout --conflict=diff3<cr>", "reverse mergetool" }
},

i = {
    name = "+trouble",
    i = { "<cmd>Trouble<cr>", "Show" },
    w = { "<cmd>Trouble lsp_workspace_diagnostics<cr>", "Worspace Diagnostics" },
    d = { "<cmd>Trouble lsp_document_diagnostics<cr>", "Document Diagnostics" },
    l = { "<cmd>Trouble loclist<cr>", "Location List" },
```

```
    q = { "<cmd>Trouble quickfix<cr>", "Quick Fix" },
    r = { "<cmd>Trouble lsp_references<cr>", "References" }
},

c = {
    name = "+lsp",
    a = { "<cmd>Lspsaga code_action<cr>", "code action" },
    A = { "<cmd>Lspsaga range_code_action<cr>", "selected action" },
    d = {
        "<cmd>Telescope diagnostics bufnr=0<cr>",
        "document diagnostics"
    },
    D = {
        "<cmd>Telescope diagnostics<cr>",
        "workspace diagnostics"
    },
    -- f = { "<cmd>LspFormatting<cr>", "format" },
    -- f = { "<cmd>Neoformat<cr>", "format" },
    f = { "<cmd>Format<cr>", "format" },
    n = { "<cmd>LspInfo<cr>", "lsp info" },
    -- r = {"<cmd>lua vim.lsp.buf.rename()<cr>", "rename"},
    r = { "<cmd>RunFile<cr>", "Rund file" },
    R = { "<cmd>RunFile tab<cr>", "Rund file (tab)" },
    x = { "<cmd>RunCode<cr>", "Rund code" },
    m = { "<cmd>LspInstallInfo<cr>", "manage LSP" },
    v = { "<cmd>LspVirtualTextToggle<cr>", "lsp toggle virtual text" },
    i = { "<cmd>Lspsaga lsp_finder<cr>", "lsp finder" },
    c = { "<cmd>Telescope lsp_dynamic_workspace_symbols<cr>", "Workspace symbols" },
    s = { "<cmd>Telescope lsp_document_symbols<cr>", "Document Symbols" },
    w = { "<cmd>Telescope lsp_dynamic_workspace_symbols<cr>", "Workspace symbols" },
    -- C = {"<cmd>Telescope lsp_document_symbols<cr>", "document symbols"},
    l = { "<cmd>Lspsaga show_line_diagnostics<cr>", "line_diagnostics" },
    [">"] = {
        "<cmd>Lspsaga diagnostics_jump_next<cr>", "next diagnostic"
    },
    ["<"] = {
        "<cmd>Lspsaga diagnostics_jump_prev<cr>", "previous diagnostic"
    },
    p = { "<cmd>Lspsaga preview_definition<cr>", "preview definition" },
    q = { "<cmd>Telescope quickfix<cr>", "quickfix" },
    h = { "<cmd>Lspsaga signature_help<cr>", "signature help" },
    T = { "<cmd>LspTypeDefinition<cr>", "type defintion" },
    S = { "<cmd>!ssort %<cr>", "Sort Code" },
    I = { "<cmd>PyrightOrganizeImports<cr>", "Sort Imports" },
    o = { "<cmd>SymbolsOutline<cr>", "Symbols sidebar" },
    _ = { "<Plug>(pydocstring)<cr>", "doc string" },
    g = { "<cmd>Neogen func<Cr>", "Func Doc" },
    G = { "<cmd>Neogen class<Cr>", "Class Doc" }
},

v = {
    name = "+view",
    w = { "<cmd>StripWhitespace<cr>", "strip whitespace" },
    b = { "<cmd>call SetBackground()<cr>", "background" },
    h = { "<cmd>call GoHighlight()<cr>", "highlight" },
    o = { "<c-w>o", "only window" },
```

```
    q = { "<c-w>q", "close window" },
    r = { "<cmd>call SetupWrappingSoft()<cr>", "wrap text (soft)" },
    R = { "<cmd>call SetupWrappingHard()<cr>", "wrap text (hard)" },
    s = { "<cmd>set invspel<cr>", "spell checker" },
    c = { "<cmd>ContextToggle<cr>", "show context" },
    t = { "<cmd>TransparentToggle<cr>", "Toggle transparency" },
    v = { "<cmd>SymbolsOutline<cr>", "Symbol Sidebar" },
    S = { "<cmd>set scrollbind!<cr>", "bind scrolling" }
},

y = {
    name = "+RIPPLE",
    o = { "<Plug>(ripple-openRepl)<cr>", "open REPL" },
    y = { "<Plug>(ripple-send-motion)<cr>", "send motion" },
    l = { "<Plug>(ripple-send-line)<cr>", "send motion" },
    b = { "<Plug>(ripple-send-buffer)<cr>", "send motion" },
    p = { "<Plug>(ripple-send-previous)<cr>", "send last" }
},

T = {
    name = "+table",
    m = { "<cmd>TableModeToggle<cr>", "toggle" },
    e = { "<cmd>TableModeEnable<cr>", "enable" },
    d = { "<cmd>TableModeDisable<cr>", "disable" },
    r = { "<cmd>TableModeRealign<cr>", "align" },
    a = { "<cmd>TableModeRealign<cr>", "align" },
    t = { "<cmd><Plug>(table-mode-tabelize)<cr><cr>", "make table" }
},
x = {
    name = "+terminal",
    X = { "<cmd>FloatermNew<cr>", "new" },
    x = { "<cmd>FloatermToggle<cr>", "toggle" },
    f = { "<cmd>FloatermNew  --width=0.95 --height=0.95  vifm<cr>", "vifm" },
    t = { "<cmd>FloatermNew  --width=0.95 --height=0.95 bpytop<cr>", "bpytop" },
    h = { "<cmd>FloatermNew  --width=0.95 --height=0.95 htop<cr>", "htop" },
    P = { "<cmd>FloatermNew ipython<cr>", "python" },
    n = { "<cmd>FloatermNext<cr>", "next" },
    p = { "<cmd>FloatermPrev<cr>", "prev" },
    r = { "<cmd>FloatermSend<cr>", "send" },
    g = { "<cmd>FloatermNew  --width=0.95 --height=0.95 lazygit<cr>", "lazygit" },
    ["]"] = { "<cmd>FloatermNext<cr>", "next" },
    ["["] = { "<cmd>FloatermPrev<cr>", "prev" },
    ["\\"] = {
        "<cmd>FloatermNew --wintype=normal --position=right<cr>", "right"
    },
    ["-"] = {
        "<cmd>FloatermNew --wintype=normal --height=0.3 --position=bottom<cr>",
        "bottom"
    }
},

t = {
    name = "+test",
    n = { "<cmd>TestNearest<cr>", "nearest" },
    f = { "<cmd>TestFile<cr>", "file" },
    s = { "<cmd>TestSuite<cr>", "suite" },
```

```
    t = { "<cmd>TestSuite --lf -vv<cr>", "failed" },
    l = { "<cmd>TestLast<cr>", "last" },
    g = { "<cmd>TestVisit<cr>", "visit" },
    v = { "<cmd>TestNearest -vv<cr>", "nearest vv" },
    d = { "<cmd>lua require('dap-python').test_method()<cr>", "debug" },
    N = { "<Plug>(ultest-run-nearest)<cr>", "nearest" },
    F = { "<Plug>Ultest<cr>", "file" },
    ["]"] = { "<Plug>(ultest-next-fail)<cr>", "next failure" },
    ["["] = { "<Plug>(ultest-prev-fail)<cr>", "prev failure" },
    S = { "<Plug>(ultest-summary-toggle)<cr>", "toggle summary" },
    G = { "<Plug>(ultest-summary-jump)<cr>", "goto summary" },
    O = { "<Plug>(ultest-output-show)<cr>", "show output" },
    J = { "<Plug>(ultest-output-jump)<cr>", "jump to output" },
    X = { "<Plug>(ultest-stop-file)<cr>", "stop file" },
    x = { "<Plug>(ultest-stop-nearest)<cr>", "stop nearest" },
    a = { "<Plug>(ultest-attach)<cr>", "stop nearest" }
},

l = {
    name = "+LateX",
    v = { "<cmd>VimtexView<cr>", "View" },
    c = { "<cmd>VimtexCompile<cr>", "Auto Compile" },
    S = { "<cmd>VimtexStop<cr>", "Stop" },
    x = { "<cmd>VimtexCompileSelected<cr>", "CompileSelected" },
    X = { "<cmd>VimtexStopAll<cr>", "StopAll" },
    C = { "<cmd>VimtexCompileSS<cr>", "Compile once" },
    n = { "<cmd>VimtexClean<cr>", "Clean" },
    m = { "<cmd>VimtexContextMenu<cr>", "ContextMenu" },
    o = { "<cmd>VimtexCompileOutput<cr>", "CompileOutput" },
    s = { "<cmd>VimtexStatus<cr>", "Status" },
    t = { "<cmd>VimtexTocToggle<cr>", "TocToggle" },
    r = { "<cmd>VimtexReload<cr>", "Reload" },
    R = { "<cmd>VimtexReloadState<cr>", "ReloadState" },
    L = { "<cmd>VimtexCountLetters<cr>", "CountLetters" },
    W = { "<cmd>VimtexCountWords<cr>", "CountWords" },
    l = { "<cmd>VimtexLog<cr>", "Log" },
    i = { "<cmd>VimtexInfo<cr>", "Info" },
    p = { "<cmd>VimtexDocPackage<cr>", "DocPackage" },
    P = { "<cmd>call PresentWithPympress()<cr>", "Present" }
},

a = {
    name = "+actions",
    h = { "<cmd>let @/ = ''<cr>", "remove search highlight" },
    n = { "<cmd>set nonumber!<cr>", "line-numbers" },
    r = { "<cmd>set norelativenumber!<cr>", "relative line nums" },
    t = { "<cmd>FloatermToggle<cr>", "terminal" },
    w = { "<cmd>StripWhitespace<cr>", "strip whitespace" },
    s = { "<cmd>Ds1z=`]<cr>", "correct spelling" },
    y = { "<cmd>let @+= expand(' % ')<cr>", "copy path" },
    l = { "<cmd>%s/^/\\=printf('%-4d', line('.'))<cr>", "write line numbers" },
    x = { "<cmd>cclose<cr>", "close quickfix" },
},
```

}, { prefix = “” })