開発環境キー操作チートシート

Neovim + tmux + Python(ipdb / cProfile / logging) + Conform(ruff)

環境前提

- **Neovim leader:** `Space`
- **ウィンドウ移動:** `Ctrl-h/j/k/l`
- **バッファ移動:** `Space n` / `Space p`
- **フォーマット: 保存時自動** + `Alt-F` + `Space fmt`
- **tmux prefix:** `Ctrl-a`
- **ペイン/ウィンドウ番号は1始まり**

tmux

セッション

- **新規:** `tmux new -s <name>`
- **一覧:** `tmux ls`
- **接続:** `tmux attach -t <name>`
- **デタッチ:** `Ctrl-a d`
- **終了:** `exit`

ウィンドウ

- **新規:** `Ctrl-a c`
- **次/前:** `Ctrl-a n` / `Ctrl-a p`
- **番号移動:** `Ctrl-a 1`
- **名前変更:** `Ctrl-a ,`

ペイン

- **縦分割:** `Ctrl-a %`
- **横分割:** `Ctrl-a "`
- **移動:** `Ctrl-a ←/→/↑/↓`
- **レイアウト調整:** `Ctrl-a space`
- **閉じる:** `exit`

コピーモード

- **開始:** `Ctrl-a [`
- **終了:** `q`

Neovim 基本操作

モード

- **ノーマル**
- **挿入:** `i a o O`
  - **i カーソル位置の直前**
  - **a カーソル位置の直後**
  - **o 現在行の下に新しい行→挿入モード**
  - **O 現在行の上に新しい行→挿入モード**
- **ビジュアル:** `v V Ctrl-v`
- **コマンド:** `:`

移動

- **基本:** `h j k l`
- **単語:** `w` / `b`
- **行頭/末尾:** `0` `^` `$`
- **ファイル:** `gg G`
- **半ページ:** `Ctrl-d Ctrl-u`
- **括弧対応:** `%`

編集

- **1文字削除:** `x`
- **1行削除:** `dd`
- **コピー:** `yy`
- **貼り付け:** `p P`
- **Undo/Redo:** `u` `Ctrl-r`

保存/終了

- **保存:** `:w`
- **終了:** `:q`
- **保存終了:** `:wq`
- **強制終了:** `:q!`

コメントアウト

- `I` **行頭へ移動して挿入モード +** `#`
- `Ctrl-v`（または`Ctrl-q`）**で矩形選択**
- `I#` **と入力、** `Esc`
- **コメント記号を変える場合は上記の** `#` **を** `//` **などに変更**

演算子 + 動作

- **削除:** `d`
- **変更:** `c`
- **コピー:** `y`
- **インデント:** `>` `<`

例:

- `dw` **ワードを削除**
- `c$` **現在位置から行末まで削除**
- `y3j` **下に3行ヤンク**
- `ciw` **ワードを削除し挿入モード**
- `ci"` **"の中身を削除**
- `di(` **()の中身を削除してノーマルモード**
- `dap` **段落を丸ごと削除**

繰り返し:

- `.`

検索・置換

- **検索:** `/pattern`
- **次/前:** `n` / `N`
- **全体置換:** `:%s/old/new/g`
- **確認付き:** `:%s/old/new/gc`
- **範囲指定:** `:10,20s/foo/bar/g`
- **ビジュアル範囲:** `:'<,'>s/foo/bar/g`

インデント・整形

- **右/左:** `>>` `<<`
- **複数行:** `V` → `>` / `<`
- **全体整形:** `gg=G`

矩形編集

- **開始:** `Ctrl-v`
- **入力:** `I`
- **確定:** `Esc`

レジスタ・マクロ

- **レジスタ保存:** `"ayy`
- **レジスタ貼付:** `"ap`
- **マクロ開始:** `qa`
- **マクロ終了:** `q`
- **実行:** `@a`
- **複数回実行:** `10@a`

Neovim 独自キーマップ

ウィンドウ/バッファ

- **ウィンドウ移動:** `Ctrl-h/j/k/l`
- **次バッファ:** `Space n`
- **前バッファ:** `Space p`

Telescope

- **ファイル検索:** `Space ff`
- **文字列検索:** `Space fq`
- **バッファー覧:** `Space fb`
- **ヘルプ検索:** `Space fh`

LSP

- **定義へ:** `gd` / `Space df`
- **参照:** `gr` / `Space rf`
- **リネーム:** `Space rn`
- **フォーマット:** `Space fmt`
- **アウトライン:** `Space o`
- **シンボル検索:** `Space s`
- **コードアクション:** `Space a`
- **ホバー:** `K` / `Space h`
- **診断表示:** `Space d`
- **診断移動:** `[d` `]d`
- **宣言:** `gD`
- **実装:** `gi`
- **型定義:** `gt`
- **シグネチャ:** `Ctrl-k`

Conform + Ruff

フォーマット実行

- **保存時自動**
- **手動:** `Alt-F`
- **手動(LSP経由):** `Space fmt`

確認コマンド

- **filetype確認:** `:set ft?`
- **Conform状況:** `:ConformInfo`
- **ruff確認:** `:echo executable("ruff")` / `:echo exepath("ruff")`

ipdb

導入

- `pip install ipdb`

ブレークポイント

```
import ipdb; ipdb.set_trace()
```

実行

- `python -m ipdb script.py`
- `pytest --pdb`

基本操作

- **次へ:** `n`
- **中へ:** `s`
- **続行:** `c`
- **表示:** `l`
- **値表示:** `p x`
- **pretty表示:** `pp x`
- **スタック:** `w`
- **上/下フレーム:** `u` / `d`
- **終了:** `q`

cProfile

全体計測

```
python -m cProfile script.py
```

主要列

- `ncalls`
- `tottime`（自己時間）
- `cumtime`（累積時間）
- `percall`

pstats解析

```
import pstats

p = pstats.Stats("result.prof")
p.sort_stats("cumtime").print_stats(20)
p.sort_stats("tottime").print_stats(10)
```

ソートキー: `"tottime"` / `"cumtime"` / `"calls"`

snakeviz

```
pip install snakeviz
uv run python -m cProfile -o result.prof main.py
snakeviz result.prof -s -H 0.0.0.0 -p 8080 result.prof
# ↓□□□□□URL□□□□□
```

gprof2dot

```
pip install gprof2dot
gprof2dot -f pstats result.prof | dot -Tpng -o graph.png
```

logging

最小構成

```
import logging
logging.basicConfig(level=logging.INFO)
```

レベル: DEBUG / INFO / WARNING / ERROR / CRITICAL

logger生成

```
import logging
logger = logging.getLogger(__name__)
```

推奨フォーマット

```
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s %(levelname)s %(name)s %(message)s"
)
```

例外ログ

```
try:
    1 / 0
except Exception:
    logger.exception("unexpected error")
```

環境変数制御

```
import os
import logging

level = os.getenv("LOG_LEVEL", "INFO").upper()

logging.basicConfig(
    level=level,
    format="%(asctime)s %(levelname)s %(name)s %(message)s"
)
```

実行:

```
LOG_LEVEL=DEBUG python app.py
```

Handler分離

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

formatter = logging.Formatter(
    "%(asctime)s %(levelname)s %(name)s %(message)s"
)

ch = logging.StreamHandler()
ch.setLevel(logging.INFO)
ch.setFormatter(formatter)

fh = logging.FileHandler("app.log")
fh.setLevel(logging.DEBUG)
fh.setFormatter(formatter)

logger.addHandler(ch)
logger.addHandler(fh)
```