Pythonista のための Cursor 使い方ガイド

1. 導入・概要

Cursor とは？

- Visual Studio Code (VS Code) の fork である IDE
- 生成AI機能が標準搭載されている
- コード補完
- 相談（Ask）
- 編集（Agent）

そもそも VS Code とは？

- Microsoft が開発しているフリーの IDE・エディタ
- 言語を問わず使用可能（Python、JavaScript、小説執筆等のプログラミング以外の用途にも)
- 拡張機能により各自の用途に合わせてカスタマイズ
- オープンソース（そのため Cursor が VS Code を fork することが可能）

なぜ Cursor を学ぶのか？

- AI活用のしやすさ
  - Cursor の AI 機能は非常に便利
  - GitHub Copilotだけなのは心細い
- IDEとしての魅力
  - PyCharm が複雑化し、初心者には使いにくくなった面がある
  - VS Code 系 IDE は共通性が高い（一つ覚えれば他も使える）
  - VS Code, Windsurf, その他の VS Code 系エディタ, ...
  - Python 以外の開発にも活用可能
    - ps1, .bat, .js, ...
- 選択肢は多い方が良い

この勉強会で学べること

- VS Code 系 IDE の基本的な使い方
- Cursor のAI機能の活用方法

対象者

- Python の基本文法に慣れている方
- Git・GitHub の基本操作を理解している方
- PyCharm 等の IDE での開発経験がある方

端末の前提条件

- Python がインストール済みの環境
- Git がインストール済みの環境

2. インストールと初期設定

2.1 Cursor のインストール

1. 公式サイトからダウンロード
2. インストール実行
3. アカウント登録

2.2 初期に導入したい拡張機能

拡張機能名	開発者	機能・説明
Python	Microsoft	Python 開発の基本機能
Python Debugger	Microsoft	デバッグ機能
Pylance	Microsoft	Python 言語サーバー
GitLens	GitKraken	Git 機能の強化
Git Graph	mhutchie	Git履歴の可視化
Japanese Language Pack for Visual Studio Code	MS-CEINTL	日本語化
PowerShell	Microsoft	Windows環境の場合

Python Extension Pack (提供元はMicrosoftなはず)を入れると上記のうちPython系複数をいっきに入れられる。
ただし、提供元を確認すること！
pluginには、ごく稀に悪意をもって導入させようとしているものもあるので)

- 知らない拡張機能を導入するときは...
  - 提供元の表示を見る
  - 提供元のサイトを見る
  - 評価を確認する
  - 最終更新日が古すぎないか確認

2.3 初回起動時の基本設定

- テーマの選択
- フォントサイズの調整
- 自動保存の設定 (files.autoSave)

3. 画面構成とUI

3.1 基本レイアウト

名称	説明
メニューバー	従来のメニュー
ファイル検索/コマンドパレット	上部中央のインプット領域
アクティビティバー	左端のアイコン群
サイドバー	エクスプローラなどのパネル
コードペイン	コード編集領域
ターミナルペイン	下部のターミナル・デバッグコンソールなど
Cursor ペイン	AI機能専用パネル

3.2 Cursor/PyCharmの操作上での重要な違い

3.2.1 ファイルの保存について

- Cursor/VS Code では明示的に保存（Ctrl+S）しないとファイルが保存されない
- 未保存ファイルはタブに白丸（●）が表示される

3.2.2 .py ファイルの実行について

PyCharm等のIntelliJ系IDEとの重要な違い - Cursor/VS Code では表示されている .py が実行される - PyCharm ではファイルのタブを選んで実行(表示されているファイルとは限らない)

4. 基本操作

4.1 最初のプロジェクト作成

1. フォルダを開く（File → Open Folder）
2. Python ファイルを作成
3. コードを記述

4.2 コードの実行

Visual Basic Editor のショートカットを思い出せば難しくない。

操作	ショートカット	説明
通常実行	F5	デバッグモードで実行（ブレークポイントで停止）
実行（デバッグなし）	Ctrl + F5	デバッグモードを使わずに直接実行
ステップオーバー	F10	現在行を実行し、次の行へ移動（関数内には入らない）
ステップイン	F11	現在行を実行し、関数内部に入る
ステップアウト	Shift + F11	現在の関数から抜け出す
ブレークポイント設定/解除	F9	現在行にブレークポイントを設定または解除

F5とCtrl + F5の違い:

ショートカット	モード	説明
F5	デバッグモードで実行	ブレークポイントで停止し、変数の値を確認できる
Ctrl + F5	通常実行	デバッグ機能を使わずに高速で実行

実行にかかる設定ファイルを作れる(後述)。
コマンドライン引数を受け取る実行の動作確認等に使える。なので、Django プロジェクトなどではこれを活用することになる。(`python manage.py test hoge --tag=fuga --parallel 3` とか)

5. AI機能の活用

5.1 Cursor ペインの使い方

機能	説明
Agent	コードの編集や生成を AI に依頼
Ask	質問・相談
Manual	手動でのやり取り

5.2 モデル選択

モード	説明
Auto	自動選択
MAX Mode	高性能モデル（使用制限あり）

5.3 AI編集の基本フロー

Cursor ペインから

1. Cursor ペインで編集内容を指示
2. AI が提案するコード変更を確認
3. 変更を受け入れるか拒否するかを選択
    - ファイル内で個別に選択
    - ファイルごとに一括選択

コードペインから

1. 範囲選択
2. Quick Edit
3. 修正方針を述べる
4. 変更を受け入れるか拒否するかを選択
    - ファイル内で個別に選択
    - ファイルごとに一括選択

Tips: 変更受け入れはこまめにしたほうがよい(ハルシネーションで急に精度が落ちることがある)

5.4 ファイル・ログのドロップ機能

ファイルや選択範囲、エラーログを Cursor ペインにドロップして相談可能

6. ターミナルペイン

6.1 ターミナルの起動

- メニュー「ターミナル」から
- 複数ターミナルを起動できる

7. 設定とカスタマイズ

7.1 設定画面へのアクセス

- File → Preferences → Settings
- または Ctrl + ,（カンマ）

7.2 主要な設定項目

- **一般設定**: エディタの動作設定
- **Cursor Settings**: AI機能に関する設定

VSCodeのグローバル設定は以下の場所にあります：

OS	パス
Windows	`%APPDATA%\Code\User\settings.json`
macOS	`~/Library/Application Support/Code/User/settings.json`
Linux	`~/.config/Code/User/settings.json`

8. プロジェクト管理、グローバル設定管理

設定は基本的に .json ファイル。編集方法は以下のいずれか。

1. ファイル → ユーザ設定 ... から設定画面で編集
2. 当該設定を記述した .json を直接編集(ない場合は新規作成)

上記 2. の方法での編集は、コマンドパレットから該当編集対象の .json を見つけてきてコードペイン上で行うこと。
(.jsonの編集はしんどいが、AIに作らせれば楽。文法間違いもAIに見つけさせて修正させればOK)

8.1 プロジェクトの概念

- Windows のフォルダ単位で管理（PyCharm と同様）
- 特殊フォルダ：`.vscode` と `.cursor`

8.2 .vscode フォルダ

プロジェクトごとの設定ファイルが格納される。以下は例。

ファイル名	用途
settings.json	各種設定
launch.json	実行・デバッグ設定（Django プロジェクト等で重要）
tasks.json	プロジェクト起動時の自動実行タスク

8.3 .cursor フォルダ

Cursor 特有の設定が格納される。以下は例。

- ***.mdc**: Cursor への指示ルール

8.4. VSCodeのグローバル設定

`%APPDATA%\Code\User\settings.json`

例: C:\Users\山田太郎\Code\User\settings.json

8.5 Cursorのグローバル設定

`%APPDATA%\Cursor`

例: C:\Users\山田太郎\AppData\Roaming\Cursor

9. ファイル検索/コマンドパレットの活用

9.1 基本的な使い方

ファイル検索/コマンドパレットの違いは、先頭に">を入力したかどうかの違いだけ。
言い換えると、ファイル検索で "> と入力するとコマンドパレットモードに切り替わる。

コマンドの例:

- Python: Select Interpreter（Python環境の選択）
- Python: Create Environment（仮想環境の作成）
- Git: Clone（リポジトリのクローン）

演習: 新しいプロジェクトを作って、仮想環境をコマンドパレットから作ってみよう！

1. [Ctrl] + [Shift] + [P]
2. 「Python仮想環境の作成」を選択
   - requirements.txt がある場合は検知して導入を促してくれる

10. ショートカット一覧

以下では複数箇所に登場するショートカットもある。

Ctrl + C, Ctrl + V, Shift + 下/上 といったテキスト操作の基本のショートカット等はいちいち示さない。
Cursorのメニュー 編集 にあるものはほぼすべて無意識に使えるようになっておくべきである。(Emmetを除く)
テキスト操作のショートカットはプログラミング以外の機会にも積極的に活用して各自習得に励むこと。

10.1 ペイン表示/非表示切り替え系

ショートカット	機能	説明
Ctrl + B	サイドバー表示/非表示	エクスプローラなどのサイドバーを切り替え
Ctrl + Shift + E	エクスプローラを表示	エクスプローラをアクティブに
Ctrl + Shift + F	検索/置換ダイアログを表示	PJ内のコードから横断検索
Ctrl + J	パネル表示/非表示	ターミナルなどの下部パネルを切り替え
Ctrl + @	ターミナル表示/非表示	統合ターミナルを開く/閉じる
Ctrl + Shift + @	新しいターミナル	新しいターミナルインスタンスを作成
Ctrl + I	Cursor ペイン表示/非表示	Cursor の基本AI機能
Ctrl + @	ターミナル表示/非表示	ターミナルパネルにフォーカス

10.2 Cursor AI操作系

ショートカット	機能	説明
Ctrl + I	Cursor ペイン表示/非表示	Cursor の基本AI機能
Ctrl + Shift + I	新スレッドでCursor ペインを表示	Cursor の基本AI機能
Ctrl + L	選択範囲をAIの参照に追加	コードペイン、ターミナルペインで範囲選択可能

ショートカット	機能	説明
Ctrl + Shift + L	選択範囲をAIの参照に追加しつつ新スレ起動	コードペイン、ターミナルペインで範囲選択可能
Ctrl + K	選択範囲について即座に相談	Cursor の基本AI機能
Ctrl + E	クイックオープンダイアログを表示	Cursor の基本AI機能

10.3 コマンドパレット操作系

ショートカット	機能	説明
Ctrl + P	クイックオープン	ファイル名で検索してファイルを開く
Ctrl + E	クイックオープン	Ctrl + P と同じ
Ctrl + Shift + P	コマンドパレット	すべてのコマンドにアクセス

10.4 ターミナル操作系

ショートカット	機能	説明
Ctrl + @	ターミナル表示/非表示	ターミナルパネルにフォーカス
Ctrl + Shift + @	ターミナルを追加起動	ターミナルパネルにフォーカス
Ctrl + PgUp	ひとつ上のターミナルをアクティブに	ターミナル切り替え
Ctrl + PgDn	ひとつ下のターミナルをアクティブに	ターミナル切り替え

10.5 実行系

ショートカット	機能	説明
F5	デバッグ実行	デバッグモードで実行
Ctrl + F5	実行	実行(高速 & ブレークポイントを無視)
F9	ブレークポイントの切り替え	デバッグ時の基本操作
F10	ステップオーバー	デバッグ時の次の行実行
F11	ステップイン	デバッグ時の関数内部実行
Shift + F11	ステップアウト	デバッグ時の呼び出し元への遷移

10.6 その他コードペインに関連するもの

ショートカット	機能	説明
Ctrl + /	コメント/非コメント化	コメント/非コメント化
Ctrl + G	指定された行に移動	行番号を指定して移動
F12	定義に移動	定義に移動
Ctrl + F4	コードタブを閉じる	タブの操作

10.7 表示設定系

ショートカット	機能	説明
Ctrl + ;	フォントサイズ拡大	IDE全体のフォントサイズを大きくする
Ctrl + -	フォントサイズ縮小	IDE全体のフォントサイズを小さくする

11. よくあるトラブルとその対策

以下は、Python系拡張機能がインストールされているという前提。

11.1 Python 環境が認識されない場合

1. Ctrl + Shift + P → "Python: Select Interpreter"
2. 正しいPython環境を選択

うまく行かない場合はやりなおしたほうがよい。

11.2 デバッガーが動かない場合

1. launch.json の設定を確認
2. Python Debugger 拡張機能がインストールされているか確認
3. 実行したいファイルが正しく指定されているか確認

11.3 拡張機能の競合

- 似たような機能の拡張機能が複数インストールされていないか確認
- 不要な拡張機能を無効化または削除

11.4 ショートカットが期待どおりの動作をしない

- vscode の設定でショートカットを変更していないか確認
- 拡張機能がオーバーライドしていないか確認

11.5 AIが急に言うことを聞かなくなった

- スレッドが長く続くと前のほうの指示を忘れがちになる
- 矯正不可能と諦めてさっさと別スレを立てるのが懸命

12. 参考資料

公式ドキュメント

- Cursor 公式サイト
- VS Code 公式ドキュメント