

# Python基礎学習プロジェクト - セットアップガイド

## 目次

1. 必要なツールのインストール
  2. VSCodeのセットアップ
  3. 仮想環境の作成と有効化
  4. プロジェクトのセットアップ
  5. コードの実行方法
  6. テストの実行方法
  7. デバッグの方法
  8. よくあるエラーと対処法
- 

## 必要なツールのインストール

### 1. Python のインストール

- Python公式サイトから最新版(3.9以上推奨)をダウンロード
- インストール時に「Add Python to PATH」にチェックを入れる
- インストール確認:

```
bash

python --version
# または
python3 --version
```

### 2. VSCode のインストール

- VSCode公式サイトからダウンロード
- インストール後、VSCodeを起動

### 3. VSCode拡張機能のインストール

VSCodeの左サイドバーの「拡張機能」アイコンをクリックし、以下を検索してインストール:

1. **Python** (Microsoft製) - 必須
2. **Pylance** (Microsoft製) - 必須、コード補完が強力
3. **Python Debugger** (Microsoft製) - デバッグ用
4. **Python Test Explorer** - テスト実行用(オプション)

---

## VSCodeのセットアップ

### プロジェクトフォルダの作成と開く

1. 任意の場所にフォルダを作成（例: `python_learning`）
  2. VSCodeで「ファイル」→「フォルダーを開く」から作成したフォルダを開く
  3. VSCodeのターミナルを開く: 「表示」→「ターミナル」または `Ctrl + ``（バッククォート）
- 

## 仮想環境の作成と有効化

### 仮想環境とは？

仮想環境は、プロジェクトごとに独立したPython環境を作る仕組みです。  
これにより、プロジェクト間でパッケージのバージョン競合を避けられます。

### Windows の場合

```
bash

# 1. 仮想環境を作成 (venv という名前で作成)
python -m venv venv

# 2. 仮想環境を有効化
venv\Scripts\activate

# 有効化されると、プロンプトに (venv) が表示されます
# (venv) PS C:\Users\YourName\python_learning>
```

### macOS / Linux の場合

```
bash

# 1. 仮想環境を作成
python3 -m venv venv

# 2. 仮想環境を有効化
source venv/bin/activate

# 有効化されると、プロンプトに (venv) が表示されます
# (venv) username@computer:~/python_learning$
```

### VSCodeで仮想環境を選択

1. `Ctrl + Shift + P` (macOS: `Cmd + Shift + P`) でコマンドパレットを開く

2. 「Python: Select Interpreter」を検索
  3. `./venv/bin/python` または `.\venv\Scripts\python.exe` を選択
- 

## プロジェクトのセットアップ

### 1. ファイル構成を作成

プロジェクトフォルダに以下のファイルを作成:

```
python_learning/  
├── venv/           # 仮想環境（自動生成）  
├── python_basics_tutorial.py  # メインコード  
├── test_student_pytest.py    # pytestテストコード  
├── test_student_unittest.py  # unittestテストコード  
├── requirements.txt         # 依存パッケージリスト  
└── README.md               # このファイル
```

### 2. requirements.txt を作成

VSCodeで新規ファイル `requirements.txt` を作成し、以下を記述:

```
pytest==7.4.3
```

### 3. パッケージをインストール

```
bash  
  
# 仮想環境が有効化されていることを確認してから実行  
pip install -r requirements.txt  
  
# インストール確認  
pip list
```

---

## コードの実行方法


### 方法1: ターミナルから実行（推奨）

```
bash
```

```
# 仮想環境が有効化されていることを確認
# プロンプトに (venv) が表示されているか確認

# メインプログラムを実行
python python_basics_tutorial.py
```

## 方法2: VSCodeの実行ボタン

1. `python_basics_tutorial.py` を開く
2. 右上の「」（実行ボタン）をクリック
3. または、ファイル上で右クリック → 「ターミナルで Python ファイルを実行」

## 実行結果の確認

- ターミナルに実行結果が表示される
- 同じフォルダに `app.log` ファイルが作成され、ログが記録される

## テストの実行方法

### pytest の実行

```
bash

# すべてのテストを実行
pytest

# 詳細な出力で実行
pytest -v

# 特定のテストファイルのみ実行
pytest test_student_pytest.py

# 特定のテスト関数のみ実行
pytest test_student_pytest.py::test_student_creation

# カバレッジ（テスト網羅率）を確認（要 pytest-cov）
pip install pytest-cov
pytest --cov=python_basics_tutorial
```

### unittest の実行

```
bash
```

# すべてのテストを実行

```
python -m unittest discover
```

# 特定のテストファイルを実行

```
python -m unittest test_student_unittest
```

# 特定のテストクラスを実行

```
python -m unittest test_student_unittest.TestStudent
```

# 特定のテストメソッドを実行

```
python -m unittest test_student_unittest.TestStudent.test_student_creation
```

# 詳細な出力で実行

```
python -m unittest -v test_student_unittest
```

## VSCodeでテストを実行

1. 左サイドバーの「テスト」アイコン（ビーカーマーク）をクリック
2. 「テストの構成」→「pytest」または「unittest」を選択
3. テスト一覧が表示されるので、実行したいテストをクリック

## デバッグの方法

### デバッグプリント（簡易デバッグ）

コード内の `print(f"[DEBUG] ...")` でデバッグ情報を出力しています。開発中は積極的にprintを使って変数の値や処理の流れを確認しましょう。

### VSCodeデバッガーの使用（推奨）

#### 1. ブレークポイントの設定

- コードの行番号の左側をクリックすると赤い丸が表示される
- この行で実行が一時停止する

#### 2. デバッグ設定ファイルの作成

1. 左サイドバーの「実行とデバッグ」アイコンをクリック
2. 「launch.jsonファイルを作成します」をクリック
3. 「Python」→「Python ファイル」を選択

以下の内容が自動生成されます（`.vscode/launch.json`）：

```
json
```

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Python: Current File",
      "type": "debugpy",
      "request": "launch",
      "program": "${file}",
      "console": "integratedTerminal"
    }
  ]
}
```

### 3. デバッグの実行

1. デバッグしたいPythonファイルを開く
2. ブレークポイントを設定
3. **F5** キーを押す、または「実行とデバッグ」から「デバッグの開始」
4. ブレークポイントで実行が停止する

### 4. デバッグ操作

- **F5**: 続行（次のブレークポイントまで実行）
- **F10**: ステップオーバー（次の行へ）
- **F11**: ステップイン（関数の中に入る）
- **Shift + F11**: ステップアウト（関数から出る）
- **左側のパネル**: 変数の値を確認できる
- **デバッグコンソール**: 変数の値を確認・評価できる

### ログファイルの確認

プログラム実行後、**app.log** ファイルを開いてログを確認できます。

```
bash

# ログファイルの内容を表示 (Windows)
type app.log

# ログファイルの内容を表示 (macOS/Linux)
cat app.log

# VSCodeでログファイルを開く
code app.log
```

---

## ⚠ よくあるエラーと対処法

### エラー1: `ModuleNotFoundError: No module named 'pytest'`

原因: 仮想環境が有効化されていない、またはpytestがインストールされていない

対処法:

```
bash

# 仮想環境を有効化
# Windows
venv\Scripts\activate
# macOS/Linux
source venv/bin/activate

# pytestをインストール
pip install pytest
```

### エラー2: `'python' is not recognized as an internal or external command`

原因: PythonがPATHに追加されていない

対処法:

- Pythonを再インストールし、「Add Python to PATH」にチェック
- または、`python`の代わりに`py`コマンドを使用（Windows）

```
bash

py --version
py python_basics_tutorial.py
```

### エラー3: `ImportError: cannot import name 'Student'`

原因: ファイル名が間違っている、またはファイルが同じフォルダにない

対処法:

- メインファイルが`python_basics_tutorial.py`という名前か確認
- テストファイルとメインファイルが同じフォルダにあるか確認

### エラー4: VSCodeでPythonインタープリターが選択できない

原因: Python拡張機能がインストールされていない

### 対処法:

1. VSCodeの拡張機能で「Python」(Microsoft製)をインストール
2. VSCodeを再起動
3. `Ctrl + Shift + P` → 「Python: Select Interpreter」

## エラー5: 仮想環境がアクティブ化できない (Windows PowerShell)

原因: 実行ポリシーの制限

### 対処法:

```
powershell
```

```
# PowerShellを管理者として実行し、以下を実行
```

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

```
# その後、通常通り仮想環境を有効化
```

```
venv\Scripts\activate
```

---

## 学習のポイント

### 実行してみよう

1. まずはメインプログラムを実行して全体の動きを確認
2. コードを1行ずつ読んで、コメントと照らし合わせる
3. 値を変更して再実行し、動作の違いを確認

### デバッグを試そう

1. ブレークポイントを設定して実行を止めてみる
2. 変数の値をデバッガーで確認する
3. ステップ実行で1行ずつ動作を追う

### テストを書こう

1. まずはサンプルのテストを実行
2. 既存のテストを修正して動作を確認
3. 新しいテスト関数を追加してみる

### 改造してみよう

以下のような拡張に挑戦:



- 新しいメソッドを追加（例: 最高点/最低点を取得）
  - 新しいクラスを作成（例: Teacherクラス）
  - データをファイルに保存/読み込み
  - グラフ化（matplotlibを使用）
- 

## このプロジェクトで学べること

- ✓ **基本文法:** 変数、データ型、演算子、制御構文
  - ✓ **クラスとオブジェクト指向:** クラス、インスタンス、メソッド、継承
  - ✓ **関数:** 定義、引数、戻り値、型ヒント
  - ✓ **モジュール:** import、標準ライブラリの使用
  - ✓ **例外処理:** try-except-finally、独自例外
  - ✓ **ログ:** loggingモジュールの使用
  - ✓ **テスト:** pytest、unittest、テスト駆動開発
  - ✓ **デバッグ:** printデバッグ、VSCodeデバッガー
  - ✓ **仮想環境:** venvの作成と管理
  - ✓ **VSCode:** エディタの使い方、拡張機能
- 

## 次のステップ

このプロジェクトをマスターしたら:

1. **Webフレームワーク:** Flask、FastAPI、Django
2. **データ分析:** pandas、NumPy、matplotlib
3. **機械学習:** scikit-learn、TensorFlow
4. **自動化:** Selenium、requests、Beautiful Soup
5. **API開発:** REST API、GraphQL

頑張ってください！ 🎉