0. 環境構築: Windows 編

プログラミング・データサイエンス I

2024/4

1　環境構築

環境構築

- 拡張子を表示
- Python のインストール
- 講義で必要なライブラリのインストール
- テキストエディタの導入
- GitHub との連携

最初に行うことは、プログラミングの環境を整備することです。最初に全体の流れを説明しておきます。

コンピュータで扱うファイルには、拡張子がついているものが多数あります。拡張子の情報は、そのファイルを扱うアプリケーションを指定するととても重要なものです。拡張子を表示する設定から始めます。

次に、Python をインターネットからダウンロードしてインストールします。更に、講義で必要なライブラリをインストールします。

プログラムを書き、実行する環境も大切です。今回は、Visual Studio Code というテキストエディタを使います。

最後に、サンプルプログラムを取得する環境を作り、動作を確認します。

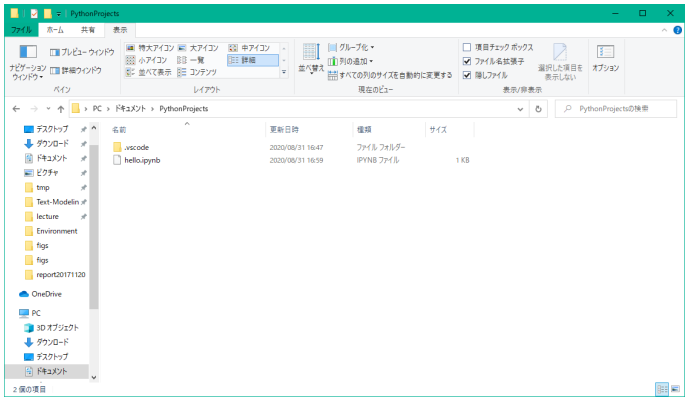
2　拡張子を表示する

拡張子を表示する

- ファイルの拡張子は、アプリケーションとの対応を示す重要な情報
- 常に、拡張子を表示するように設定変更
- エクスプローラーで設定

拡張子は、ファイルの種類を表すと同時に、そのファイルを扱うアプリケーションとの対応付けをする大切なものです。ファイルのアイコンも重要な情報ですが、コンピュータウィルス等がアイコンを偽装することもできます。エクスプローラーで拡張子を表示するように変更しましょう。既に変更している人は、この節を飛ばしてください。

エクスプローラーを開いて、「表示」メニューを開けましょう。メニューの右の方に、「ファイル名拡張子」というチェックボックスがあります。これをチェックしておきましょう (図 2)。



3　Python インストール

Python インストール

- インターネットから Python をダウンロード
  - Miniforge3 を利用
- インストーラを使ってインストール

Python という言語に対して、それを実行する環境には、いくつかのものがあります。

必要となるライブラリが最初からパッケージされたものがある一方で、最初は最小限だけをインストールし、あとから必要なものをインストールものもあります。前者は、様々なライブラリを一気にインストールすることができるというメリットがある一方で、そのために大きなディスク容量を必要とするという課題があります。今回の目的は、学習のための環境構築ですから、軽量なものを選択します。

今回は Miniforge3 を利用します。以下の URL から Windows 用インストーラーをダウンロードします。ダウンロードしたインストーラーをダブルクリックして起動します。Windows の場合には、自分だけにインストールすれば十分です。

https://github.com/conda-forge/miniforge

インストールが完了したら、Miniforge Prompt というアイコンがアプリ一覧にできるはずです。これを起動し、以下のコマンドを実行します。これにより、作業用の環境 myenv を作成します。以降は、この作業用環境にライブラリを追加します。

```
conda create -n myenv python=3.12
```

4 ライブラリのインストール

ライブラリのインストール

- 講義で必要な Python ライブラリをインストール
- Miniforge Prompt を起動
  - conda activate myenv で作業環境に移動
  - conda install コマンドを利用

次に、当面の学習のために必要なライブラリを導入します。Miniforge Prompt を起動します。起動後、作業用環境に移動します。

```
conda activate myenv
```

当面の学習に必要なライブラリを以下に示します。

```
jupyter
xlrd
lxml
pandas
```

matplotlib

requests

openpyxl

これらをインストールしてきます。例として jupyter をインストールする例です。複数パッケージをスペースで区切って指定することもできます。

```
conda install jupyter
```

最後に、作図に日本語を使うことができるライブラリを導入します。残念ながら上記のコマンドで実行できません。以下のコマンドで導入します。

```
pip install japanize-matplotlib
```

5 VSCode のインストール

VSCode のインストール

- プログラムを書き、実行するための環境
- Visual Studio Code (VSCode) の導入
- インストール途中の「追加タスクの選択」に注意

次は、プログラムを書いて、実行するための環境を作りましょう。Microsoft が無償で提供する Visual Studio Code (VSCode) を使いましょう。このエディターは、様々なプログラミング言語に対応した拡張機能があり、通常のテキストの編集にも便利です。

Microsoft Store からインストールせず、以下の URL からダウンロードしてインストールしてください。

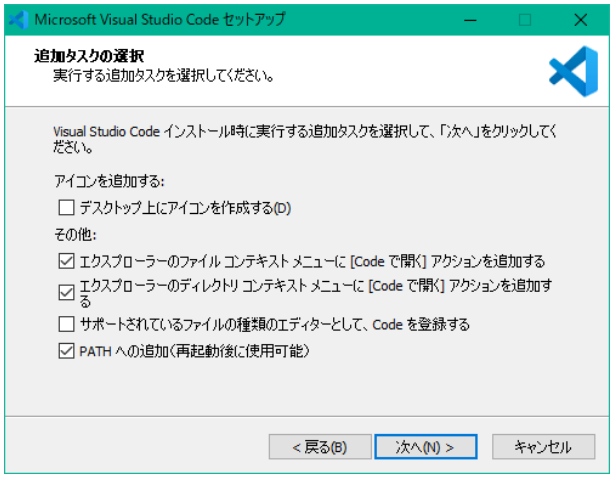
https://azure.microsoft.com/ja-jp/products/visual-studio-code/

インストールの途中で現れる「追加タスクの選択」(図 5) で、

- エクスプローラーのファイルコンテキストメニューに [Code で開く] アクションを追加する
- エクスプローラーのディレクトリコンテキストメニューに [Code で開く] アクションを追加する

の二つにチェックをしてください。上記作業を忘れた場合は、再度上書きインストールを

開始し、上記作業を行います。



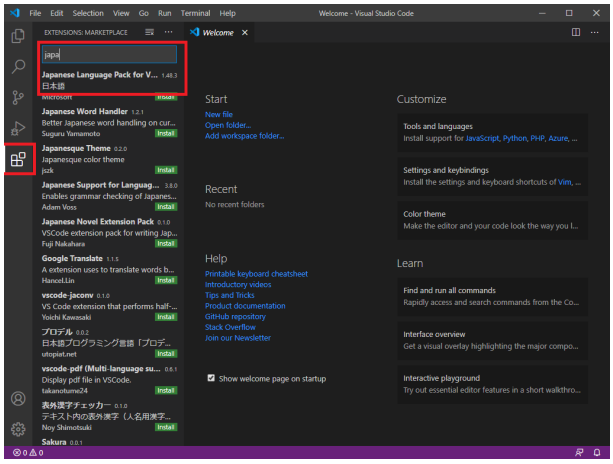
5.1 VSCode の日本語化

VSCode の日本語化

- VSCode の日本語化
- 拡張機能を利用

新しいバージョンの VSCode では、インストール直後に、日本語化パッケージのインストールが始まります。もしも、日本語化が自動で始まらない場合には、以下の手順で日本語化します。

VSCode を起動しましょう。図 5.1 の左に、小さな正方形 3 つと 1 つからなるアイコンがあります。これは、VSCode に機能を追加する際に使うアイコンです。このアイコンを押し、上のほうの検索窓に `japan` と打つと、名前に `japan` を含む拡張機能を表示します。その中から、`Japanese Language Pack` を選んでインストールします。一旦 VSCode を終了し、再度、起動すると日本語表示となります。



5.2 Python 用の拡張機能

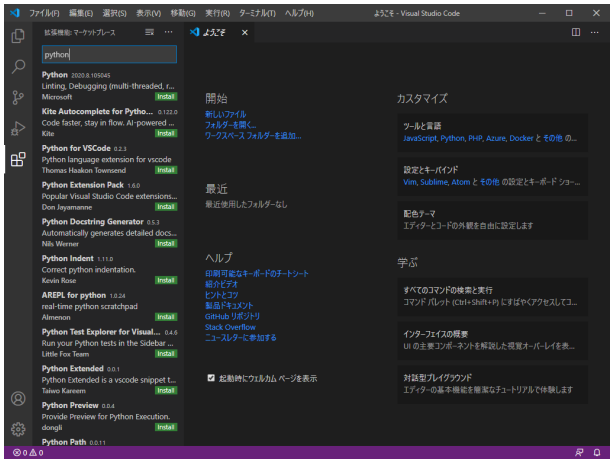
Python 用の拡張機能

- VSCode に Python 環境を導入
- 拡張機能を利用
- Python インタープリターとの対応付け

次に、Python の環境を VSCode に導入しましょう。日本語化と同様に、機能を追加します。検索窓に `python` と入力し、Microsoft が提供する

`Python`

を選んでインストールします (図 5.2)。



また、デフォルトのインタープリターを設定しておきましょう。左下の歯車のマークを押すと、設定メニューが現れます。その中の「設定」を選び、「設定の検索」窓に Python:Default と打ちましょう。Python:Default Interpreter Path に、myenv 環境へのパス

```
~\miniforge3\envs\myenv\python.exe
```

を入力します。ここで”~” は、ユーザのホームディレクトリを表しています。例えば、通常のドキュメントは~\Documents の下にあります。

もう一つ、Python Terminal と打って検索すると、

```
python>Terminal:Activate Environment
```

という設定項目があります。これのチェックを外します。この設定で、VSCode 内のターミナルが開く際に、Python 環境が自動的に起動するのを抑止することができます。

これらの設定は

```
~\AppData\Roaming\Code\User\settings.json
```

に保存されています。VSCode から編集するには、

歯車アイコン ->「設定」-> 画面右上の文書アイコン

で表示できます。関連する有用な設定をソースコード 5.1 に示します。3 番目は、基本的な文法チェックを行うためのもの、4 番目は jupyter notebook で行番号を表示するためのものです。

ソースコード 5.1　python 関連の設定

```
"python.defaultInterpreterPath":
↪  "~\\miniforge3\\envs\\myenv\\python.exe",
"python.terminal.activateEnvironment": false,
"python.analysis.typeCheckingMode": "basic",
"notebook.lineNumbers": "on",
```

もう一つ、IntelliCode という Microsoft が提供する拡張機能を入れておきましょう。コマンド等を途中まで入力すると、候補となるコマンド等を提案してくれるものです。プログラム作成が効率的になります。

6　簡単な例を作ろう

簡単な例を作ろう

- 作業用フォルダの作成
- ファイルを作成
- ファイルの編集
- 実行

課題 6.1　それでは、適当な作業をフォルダを作って、環境が動作することを確かめましょう。例えば、「ドキュメント」の下に PythonProjects というフォルダを作成しましょう。その下に更に Hello というフォルダを作りましょう。このフォルダにマウスを置き、右ボタンを押し、「Code で開く」を選びます。作成者を信頼するかを聞かれますから、「はい、作成者を信頼します」を選びます。

左の一番上にある文書のアイコンを押すとファイル操作を行うことができます。その右にエクスプローラーのメニューが表されます。文書にプラスのついたアイコンを押すとファイルを生成します。フォルダにプラスのついたアイコンは、フォルダ生成です。

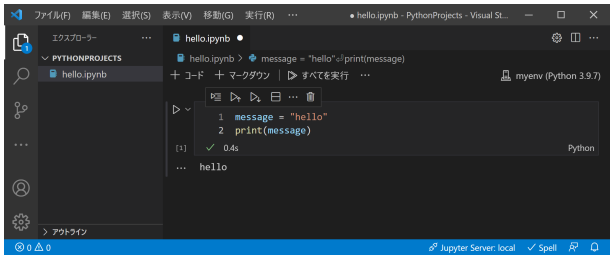
それでは、hello.ipynb というファイルを作りましょう。拡張子.ipynb は、この講義で利用する jupyter notebook を表しています。このファイルを VSCode で直接くと、編集はできますが、実行はできません。必ず、フォルダを選択してください。

図 6.1 のように、ソースコード 6.1 を入力します。入力している部分のことをセルと呼びます。セルの左にある”▷” を押すと、そのセルの部分を実行し、セルの下に結果を表示します。実行しようとすると実行するためのカーネルを聞かれることがあります。その際には、myenv を選んでください。図 6.1 では、結果として hello という文字列を表示しています。

ソースコード 6.1　簡単なプログラム

```
1 message = "hello"
2 print(message)
```

なお、右下にファイルを信用するかというメッセージが出て、実行できないことがあります。その場合には、trust ボタンを押してください。



左上にあるファイル名が着いたタブに注目してください。ファイル名に白い○がついている場合には、ファイルが保存されていないことを表しています。コントロールキーを押しながら”S” を押すと、ファイルを保存することができます。

7 追加作業

Github との連携

- 講義のサンプルプログラムを Github から配布
- Git をインストール

Github は、ソフトウェア開発のプラットフォームです。授業では、サンプルプログラムの配布に使用します。Github からプログラムをダウンロードできるようにもしましょう。

最初に行うのは、`Git` というプログラムのインストールです。以下の URL から、OS に合わせてダウンロードし、デフォルトのまま、インストールします。

`https://git-scm.com/downloads`

VSCode の設定

- いろいろ設定できます
- 例えば、配色やフォントを設定してみよう

設定アイコンの中に、「配色テーマ」というメニューがあり、背景色と文字色の組合せを変えることができます。また、「設定」というメニューから、エディターとしての様々な設定、例えばフォントの大きさなどを変更することができます。