0. 環境構築: Mac 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

- 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。

Mac では、OS のバージョンによって python のプレインストールの状況が異なります。そこで、最初に行うのは、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 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 から自分の OS に併せてダウンロードします。使用している Mac のアーキテクチャを知るには、ターミナルで `uname -m` を実行します。

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

ターミナルを開き、ダウンロードしたファイルのあるフォルダに移動します。そこで、例えば

```
chmod +x Miniforge3-MacOSX-arm64.sh
```

とるすることで、ダウンロードしたファイルを実行可能とします。その後

```
./Miniforge3-MacOSX-arm64.sh
```

として実行します。初めにライセンスが表示されるので、リターンを押していきます。同意を求められたら”yes” とします。

管理者にならずに、インストールすると、自身の~/.zshrc に設定が追記されます。そのため、ターミナルを開くと conda base という環境に直接入ってしまいます。このことを回避するために、インストール後に、インストールとは別のターミナルを開き、以下のコマンドを実行し、自動的に conda base に入らないようにします。

```
conda config --set auto_activate_base false
```

うまくいかない時には~/.zshrc ができているかを確認しましょう。ターミナルを開き

```
cd
```

でホームディレクトリへ移動します。

```
ls .zshrc
```

とした時に

```
ls: .zshrc: No such file or directory
```

とでる場合には、~/.zshrc がありません。その場合には、ターミナルで以下を実行します。

```
cd
~/miniforge3/bin/conda init zsh
```

インストールが完了したら、再びターミナルを開き、conda base と打ちます。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
```

3　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。ターミナルを開き、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
```

4　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。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://code.visualstudio.com/`

ソースコード 4.1　ワークフローの記述

```
for i in "$0"
do
    open -a 'Visual Studio Code' "$f"
done
```

zip ファイルが「ダウンロード」フォルダにダウンロードされます。zip ファイルダブルクリックすると、拡張子が app のファイルが出てきます。これをアプリケーションフォルダへ移動すると、インストール完了です。

4.1　VSCode でファイルやフォルダを開く準備

次に、コントロールキーを押しながらマウスでファイルやフォルダを選択したメニューに、VSCode でファイルやフォルダを開くメニューを追加しましょう。

はじめに、Automator というアプリを起動します。ロボットの絵のついた標準のアプリです。

- 起動して、ファイルメニューから新規を選ぶと、書類の種類を選ぶ画面となります。「クイックアクション」を選びます。
- 一番左の「ライブラリ」の列から、「ユーティリティ」を選びます。
- 二番目の「アクション」列から、「シェルスクリプトを実行」を選びます。
- 右上の「ワークフローが受け取る現在の項目」から「ファイルまたはフォルダ」を選びます。
- 右上の「検索対象」から「Finder.app」を選びます。
- 二番目の「アクション」列で選んでいる「シャルスクリプトを実行」を、中央の大きい領域にドラッグします。
- ソースコード 4.1 を記載します。”␣” は、スペースです。別の文字を入れないように注意してください。
- 「シェル」には`/bin/zsh` を、「入力の引き渡し方法」には「引数として」を指定します。
- ファイルメニューから保存します。名前は、「Visual Studio Code で開く」にしましょう。

図 1 に編集の様子を示します。

4.2　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 4.2 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

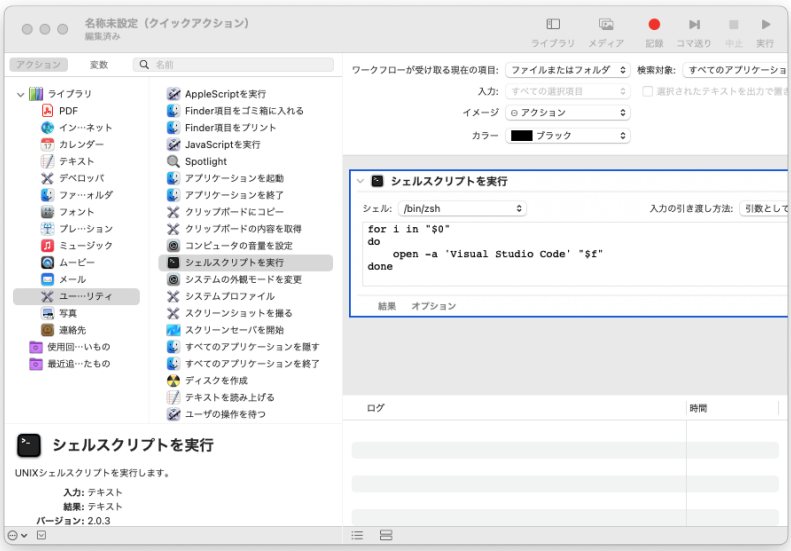
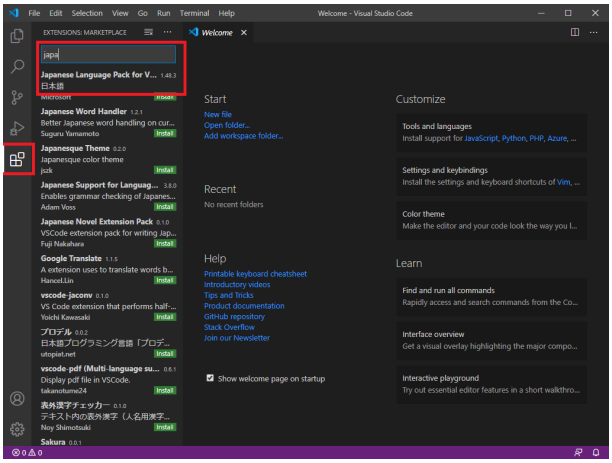


図 1　Automator での設定

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。



4.3　Python 用の拡張機能

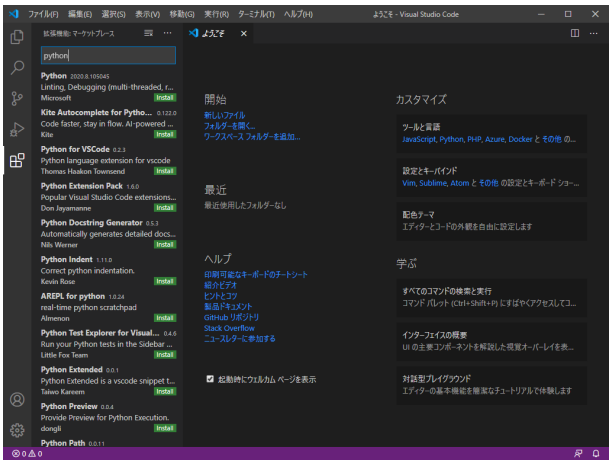
Python 用の拡張機能

- 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 に作りましょう。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 4.3)。



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

を入力しておきます。

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。

これらの設定は

~/Library/Application Support/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 4.2 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 4.2　python 関連の設定

```
"python.defaultInterpreterPath": "~/miniforge3/envs/myenv/python",
"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、候補となるモノ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。

5　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 5.1　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。

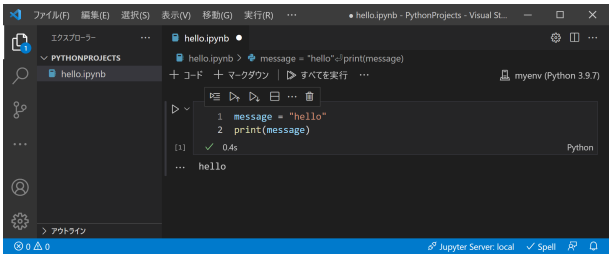
左の一番上にある文書のアイコンを押すとファイル操作を行うことができます。その右にエクスプローラーのメニューが表されます。文書にプラスのついたアイコンを押すとファイルを生成します。フォルダにプラスのついたアイコンは、フォルダ生成です。

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 5.1 のように、ソースコード 5.1 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 5.1 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 5.1　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。

6 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した。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。

7 参考: `/.zshrc`

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('~/miniforge3/bin/conda' 'shell.zsh' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "~/miniforge3/etc/profile.d/conda.sh" ]; then
        . "~/miniforge3/etc/profile.d/conda.sh"
    else
        export PATH="~/miniforge3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```