For Python

ここではPythonのプログラミング演習を行うために、個別のPCに対して行う設定等について述べます。
なお，各自で経験があるやり方を獲得している人はそちらでも良いです．

- Python：Miniconda経由で構築します
- エディタ：VsCodeを使います

1. MinicondaによるPythonのインストール

まずはMinicondaをインストールします．Minicondaとはライブラリを提供するプラットフォームであり，Anacondaという大規模プラットフォームから必要最低限な項目を選別したプラットフォームです．
パワーのあるPCを使う人はAnacondaをインストールしても良いです．

1.1 ダウンロード

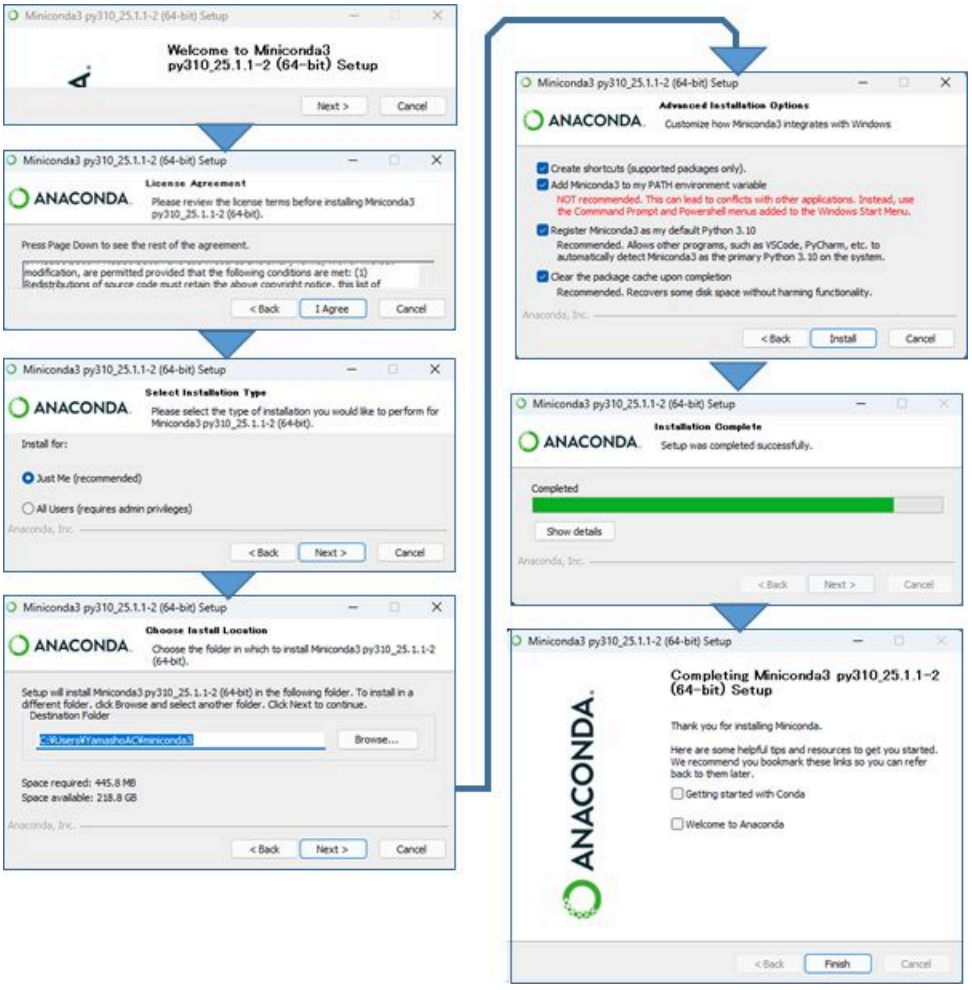
バージョンによって扱いが違いますが，2024年5月現在，安定したPython3.10を使いたいため，以下の中から該当するものをダウンロードしてください．

https://repo.anaconda.com/miniconda/

PCがWindows環境の場合，Miniconda3-py310_25.1.1-2-Windows-x86_64.exe になります．

1.2 インストール

ダウンロードしたファイルをクリックして以下のようにインストールしていきます．



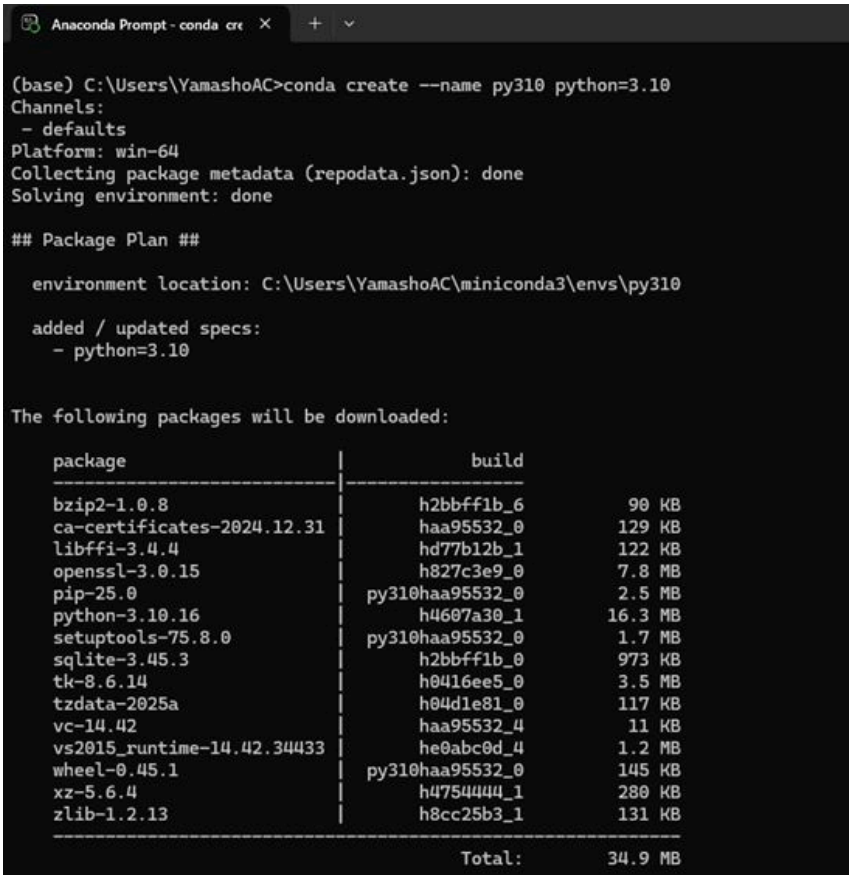
インストールが終了すると，Windowsアプリに以下のフォルダとプロンプトが生成されます．

1.3 仮想環境の構築とPythonチェック

続いて，開発を行う環境を構築していきます．Pythonは様々なアプリケーション実装が可能なため，そのプロジェクトに必要な環境を作り，独自のライブラリを獲得していきます．そのようなプロジェクトごとの設定領域を，Minicondaでは仮想環境と呼びます．

では，Anaconda Prompt(Miniconda3)を起動させてください．基本表示(base)は現在のフォルダ名を示しています．ここで，仮想環境を作るために以下の命令を書き込み，Enterを押します．Python3.10で開発していきますので，仮想環境の名前は"py310" にしましょう．

```
conda create --name py310 python=3.10
```



途中で，Proceed ([y]/n) ? が現れ，処理が中断します．ここでは新たに作成する仮想環境にインストールされるソフトウェア及びライブラリの一覧が示されています．このインストールされるパッケージに問題がなければ，"y"を入力してください．

仮想環境が整いましたら，続けて以下のように打ち込みます．ここでactivateとは，作成した仮想環境に移動することを宣言しています．

```
conda activate py310
```

プロンプトの接頭が 以下のように，(py310) に変わったら成功です．

```
(py310) C:\Users\YamashoAC>
```

では，仮想環境下でうまくPythonが動作するか，確認してみることとします．(py310)において，"Python" と打ち込み，アプリケーションを起動してみましょう．

```
python
```

```
(py310) C:\Users\YamashoAC>python
Python 3.10.16 | packaged by Anaconda, Inc. | (main, Dec 11 2024, 16:19:12)
 [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

結果として，Pythonは3.10.16がインストールされていることが確認できました．

表示の命令が完了したら，"quit()" と代入して，Pythonを終了しましょう．

```
quit()
```

1.4 必要なライブラリの導入

今回は機械学習(scikit-learn）と画像処理(Open-CV)が開発できる環境とします．インターネットでうまく動作している例が，【Python3.10.**+OpenCV 4.10+VsCode1.53.2】で開発されていましたので，まずはこの環境に近づけていきましょう．

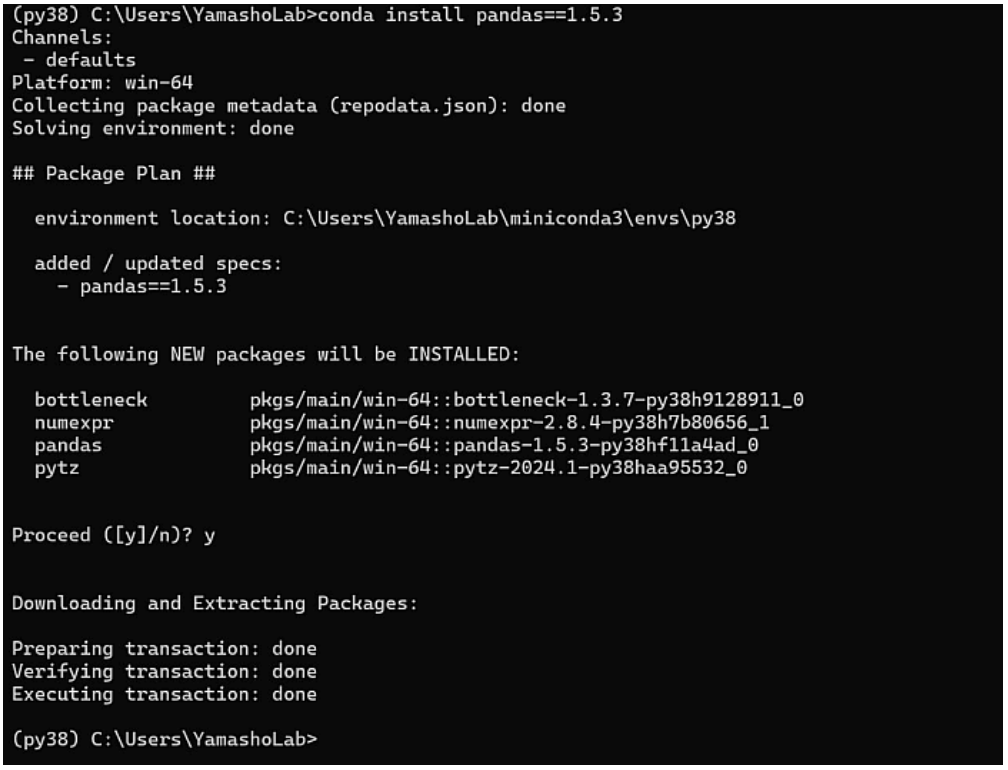
ライブラリは作成した仮想環境にインストールしていきます．ここでは**(py310)にActivate**していることを確認しながら，下記の項目を導入していきます．なお，opencvは画像処理するためのライブラリです．

- NumPy
- Pandas
- scipy
- matplotlib
- Seaborn
- scikit-learn
- opencv-python (*versionを4.10.0.84に指定）

インストールは，

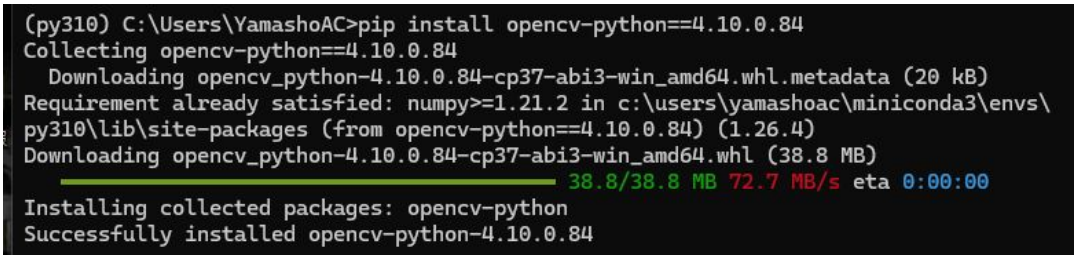
```
conda install numpy
```

で行います．" Proceed ([y]/n)? "と問われますので，versionを確認して大丈夫なら，"y" を入力してください．



また，opencv-pythonはconda環境で入れることが困難です．ですので，別の環境構築アプリである“pip” を用いて以下のように打ち込みます．

```
pip install opencv-python==4.10.0.84
```



pipもcondaもどちらもPython環境を構築することができるアプリケーションですが，目的や構造が異なるため，両立させるのはあまりお勧めしません．どちらかというとpipのほうが汎用的ですが，管理が行き届かない場合がありますので，初心者の人はcondaで環境を構築して，どうしても入れられないライブラリだけ，pipで導入することとしましょう．

最後にこれまで入れたライブラリを確認します．以下のコマンドで一覧を表示してください．

```
conda list
```

注：インストールを行う時期によって，バージョンは多少異なります

```
(py310) C:\Users\YamashoAC>conda list
# packages in environment at C:\Users\YamashoAC\miniconda3\envs\py310:
#
# Name                    Version                   Build  Channel
autopep8                  2.3.2                     pypi_0    pypi
babel                     2.17.0                    pypi_0    pypi
blas                      1.0                         mkl
blinker                   1.9.0           py310haa95532_0
bottleneck                1.4.2           py310hc99e966_0
brotli-python             1.0.9           py310h5da7b33_9
bzip2                     1.0.8                h2bbff1b_6
ca-certificates           2024.12.31           haa95532_0
certifi                   2025.1.31                 pypi_0    pypi
charset-normalizer        3.4.1                     pypi_0    pypi
click                     8.1.7           py310haa95532_0
colorama                  0.4.6           py310haa95532_0
colour                    0.1.5           py310haa95532_0
contourpy                 1.3.1           py310h214f63a_0
cycler                    0.11.0             pyhd3eb1b0_0
flask                     3.1.0           py310haa95532_0
fonttools                 4.55.3          py310h827c3e9_0
freetype                  2.12.1               ha860e81_0
geos                      3.10.6               he74ecf9_0
icc_rt                    2022.1.0             h6049295_2
icu                       73.1                 h6c2663c_0
idna                      3.10                      pypi_0    pypi
intel-openmp              2023.1.0          h59b6b97_46320
itsdangerous              2.2.0           py310haa95532_0
jinja2                    3.1.5           py310haa95532_0
joblib                    1.4.2           py310haa95532_0
jpeg                      9e                   h827c3e9_3
jpype1                    1.4.1           py310h59b6b97_0
kiwisolver                1.4.8           py310h5da7b33_0
krb5                      1.20.1               h5b6d351_0
lcms2                     2.16                 hb4a4139_0
lerc                      4.0.0                h5da7b33_0
libclang                  14.0.6          default_hb5a9fac_2
libclang13                14.0.6          default_h8e68704_2
libdeflate                1.22                 h5bf469e_0
libffi                    3.4.4                hd77b12b_1
libpng                    1.6.39               h8cc25b3_0
libpq                     17.2                 h70ee33d_0
libtiff                   4.5.1                h44ae7cf_1
libwebp-base              1.3.2                h3d04722_1
line_profiler             4.2.0           py310h214f63a_0
lz4-c                     1.9.4                h2bbff1b_1
markupsafe                3.0.2           py310h827c3e9_0
matplotlib                3.10.0          py310haa95532_0
matplotlib-base           3.10.0          py310he19b0ae_0
mkl                       2023.1.0          h6b88ed4_46358
mkl-service               2.4.0           py310h827c3e9_2
mkl_fft                   1.3.11          py310h827c3e9_0
mkl_random                1.2.8           py310hc64d2fc_0
numexpr                   2.10.1          py310h4cd664f_0
numpy                     1.26.4          py310h055cbcc_0
numpy-base                1.26.4          py310h65a83cf_0
opencv-python             4.10.0.84                 pypi_0    pypi
openjpeg                  2.5.2                hae555c5_0
openssl                   3.0.15               h827c3e9_0
packaging                 24.2            py310haa95532_0
pandas                    1.5.3           py310h4ed8f06_0
pillow                    11.1.0          py310h096bfcc_0
pip                       25.0            py310haa95532_0
ply                       3.11            py310haa95532_0
py5                       0.10.1a1                  pypi_0    pypi
pycodestyle               2.12.1                    pypi_0    pypi
pyparsing                 3.2.0           py310haa95532_0
pyqt                      5.15.10         py310h5da7b33_1
pyqt5-sip                 12.13.0         py310h827c3e9_1
python                    3.10.16              h4607a30_1
python-dateutil           2.9.0post0      py310haa95532_2
pytz                      2024.1          py310haa95532_0
qt-main                   5.15.2             h19c9488_12
requests                  2.32.3                    pypi_0    pypi
scikit-learn              1.6.1           py310h585ebfc_0
scipy                     1.15.1          py310h86d0f81_0
seaborn                   0.13.2          py310haa95532_1
setuptools                75.8.0          py310haa95532_0
shapely                   2.0.6           py310hba19cfc_0
sip                       6.7.12          py310h5da7b33_1
six                       1.16.0             pyhd3eb1b0_1
sqlite                    3.45.3               h2bbff1b_0
stackprinter              0.2.12                    pypi_0    pypi
tbb                       2021.8.0             h59b6b97_0
threadpoolctl             3.5.0           py310h9909e9c_0
tk                        8.6.14               h0416ee5_0
tkcalendar                1.6.1                     pypi_0    pypi
tomli                     2.0.1           py310haa95532_0
tornado                   6.4.2           py310h827c3e9_0
trimesh                   4.6.3                     pypi_0    pypi
tzdata                    2025a                h04d1e81_0
unicodedata2              15.1.0          py310h827c3e9_1
urllib3                   2.3.0                     pypi_0    pypi
vc                        14.42                haa95532_4
vs2015_runtime            14.42.34433          he0abc0d_4
werkzeug                  3.1.3           py310haa95532_0
wheel                     0.45.1          py310haa95532_0
xz                        5.6.4                h4754444_1
zlib                      1.2.13               h8cc25b3_1
zstd                      1.5.6                h8880b57_0
```

2. VsCodeによる環境設定

Vscodeは様々なプログラミングに対応したエディタソフトである．各言語でのプログラミングを実行させるためには様々な設定をしなければならないが，メンテナンスされた優れたエディタの一つである．今回はVsCodeからPythonを実行する環境を設定していく．

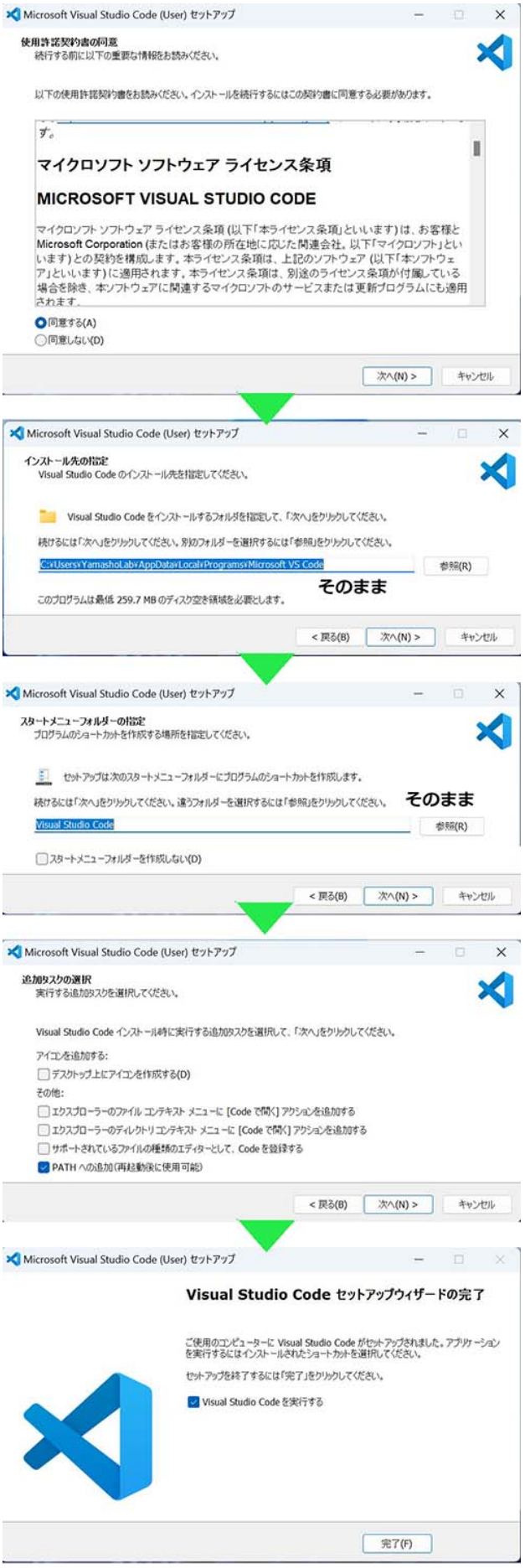
2.1 ダウンロード

VsCodeはApril 2021 (version 1.56)の更新で大幅に変更されましたので、現時点(2024/9)では1.91を導入していきます。以下のURLから ”Windows: x64” を選択してダウンロードしてください．

https://code.visualstudio.com/updates/v1_91

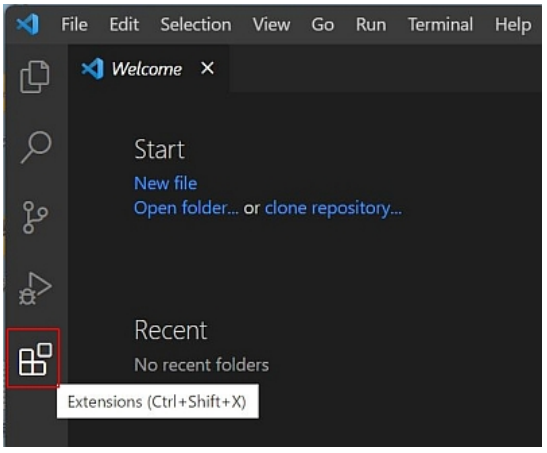
2.2 インストール

インストールはほとんど指示のままに従って実施します．

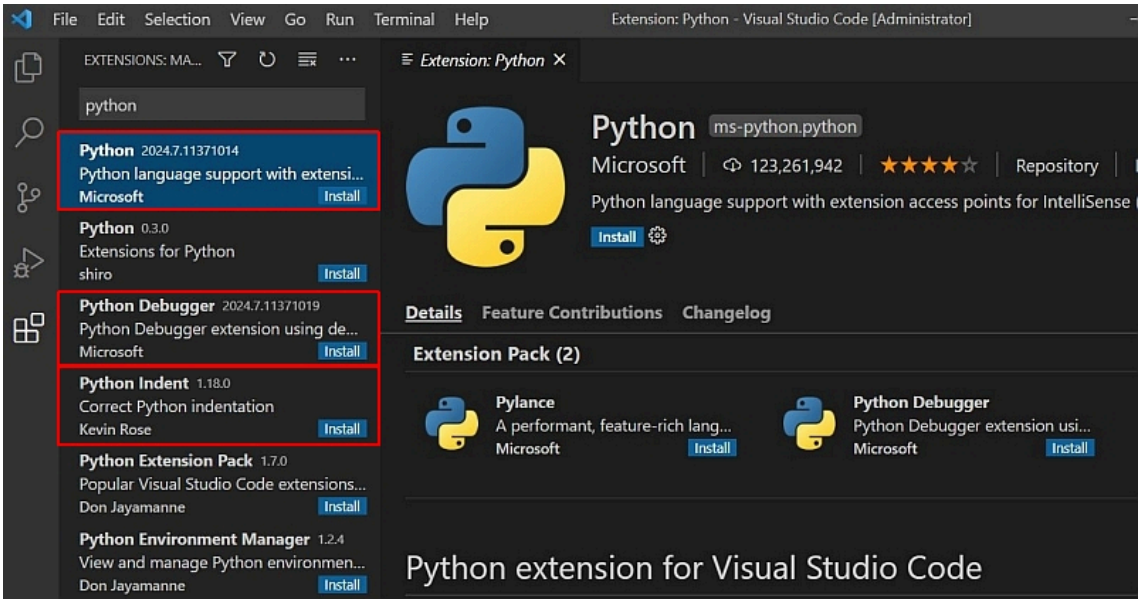

Microsoft Visual Studio Code (User) セットアップ
使用許諾契約書の同意
続行する前に以下の重要な情報をお読みください。
以下の使用許諾契約書をお読みください。インストールを続行するにはこの契約書に同意する必要があります。
マイクロソフト ソフトウェア ライセンス条項
MICROSOFT VISUAL STUDIO CODE
同意する(A)
同意しない(D)
次へ(N) >
キャンセル
Microsoft Visual Studio Code (User) セットアップ
インストール先の指定
Visual Studio Code のインストール先を指定してください。
Visual Studio Code をインストールするフォルダを指定して、「次へ」をクリックしてください。
続けるには「次へ」をクリックしてください。別のフォルダーを選択するには「参照」をクリックしてください。
C:¥Users¥YamashoLab¥AppData¥Local¥Programs¥Microsoft VS Code
参照(R)
そのまま
このプログラムは最低 259.7 MB のディスク空き領域を必要とします。
< 戻る(B)
次へ(N) >
キャンセル
Microsoft Visual Studio Code (User) セットアップ
スタートメニューフォルダーの指定
プログラムのショートカットを作成する場所を指定してください。
セットアップは次のスタートメニューフォルダーにプログラムのショートカットを作成します。
続けるには「次へ」をクリックしてください。違うフォルダーを選択するには「参照」をクリックしてください。
そのまま
Visual Studio Code
参照(R)
スタートメニューフォルダーを作成しない(D)
< 戻る(B)
次へ(N) >
キャンセル
Microsoft Visual Studio Code (User) セットアップ
追加タスクの選択
実行する追加タスクを選択してください。
Visual Studio Code インストール時に実行する追加タスクを選択して、「次へ」をクリックしてください。
アイコンを追加する:
デスクトップ上にアイコンを作成する(D)
その他:
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
サポートされているファイルの種類のエディターとして、Code を登録する
PATH への追加(再起動後に使用可能)
< 戻る(B)
次へ(N) >
キャンセル
Microsoft Visual Studio Code (User) セットアップ
Visual Studio Code セットアップウィザードの完了
ご使用のコンピューターに Visual Studio Code がセットアップされました。アプリケーションを実行するにはインストールされたショートカットを選択してください。
セットアップを終了するには「完了」をクリックしてください。
Visual Studio Code を実行する
完了(F)

2.3 拡張機能の導入

インストールが終了したら，そのままVisual Studio Codeを起動します．まず，いろいろな拡張機能を入れていくため，初期画面において　"Extensions" をクリックします．

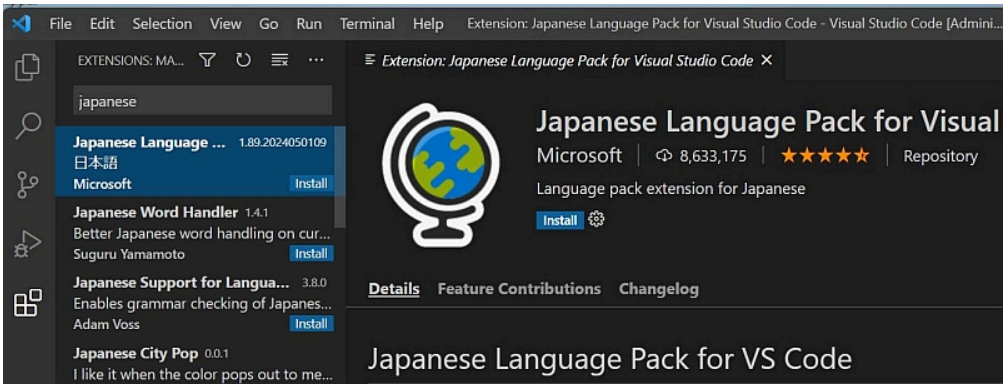


Extention画面の検索窓に" Python "と入力して，まずは　Python extension for Visual Studio Codeをインストールします．



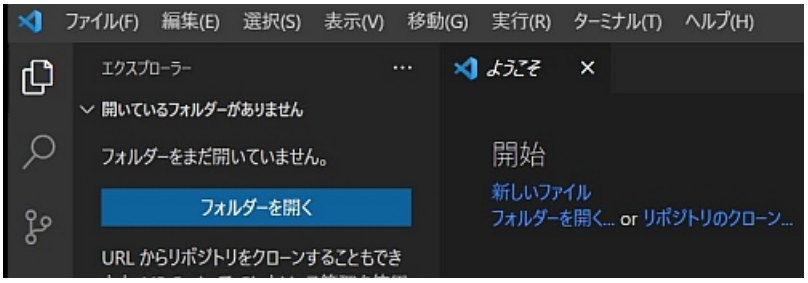
似たようなExtensionがあるので，間違えないように．特にPythonも同時に入ってしまうものもあるので，そちらは選択しないように注意してください．更に追加として，Python Debugger と Python Indent をインストールすると便利になります．

続いて，日本語をインストールします．英語のままでも良いですが，Webサイトなどでは日本語で命令を説明しているページもあるため，間違えが少なくなります．このExtensionは検索窓に "Japanese" と代入するとすぐ見つかります．

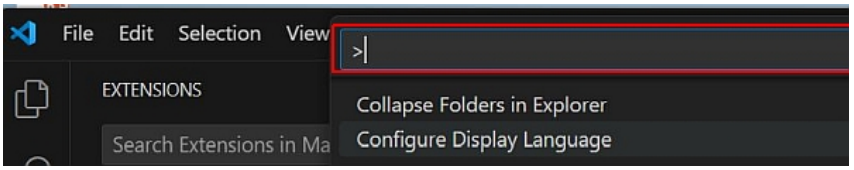


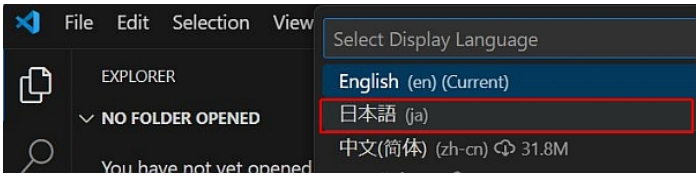
日本語パックを入れると，VsCodeの再起動を要求されるので，従って実行します．

無事日本語がインストールされれば，再起動後，以下のような画面が現れます．



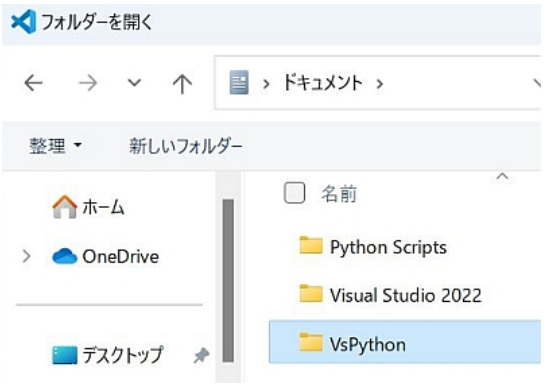
なお，PC環境によっては日本語にならない場合があります．そのときには，「View」-「Command Palete」から，"Configure Display Language" を選択します．すると，日本語を選択可能です．なお，この設定を行うと，再び再起動が促されます．



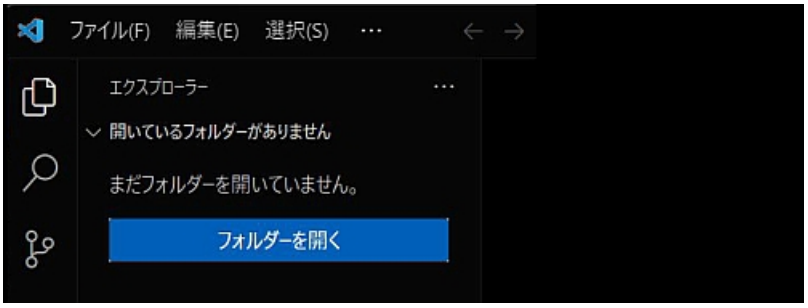


2.4 Pythonプログラミングの初期設定

続いて，プログラミングの作業設定を行っていきます．最初に，Exploreを起動させ，My Documents に作業用フォルダを事前作成しておきます．ここでは ”VsPython” とします．

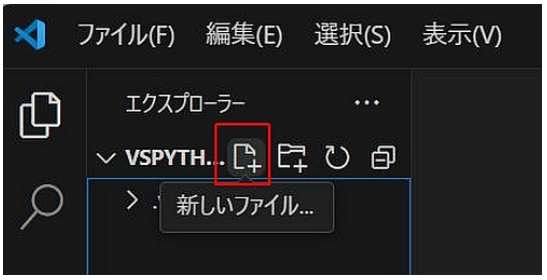


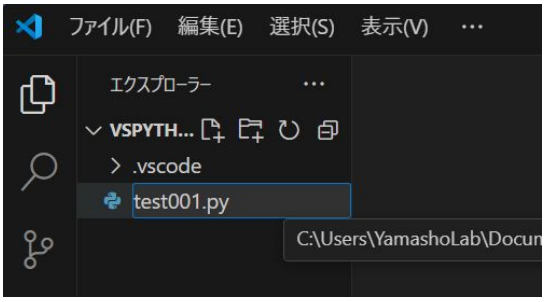
そして，VsCodeの起動画面から作成したフォルダを選択します．



このとき，セキュリティの画面が出るが，特に問題はないので，信頼しましょう(笑)．

続いて、サンプルプログラムを作成します。エクスプローラ枠の上部にある新しいファイルを作成するタグをクリックします。ファイル名はtest.pyとしましょう。

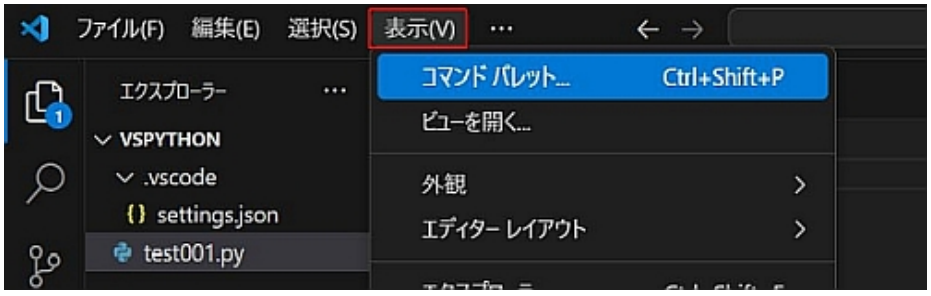


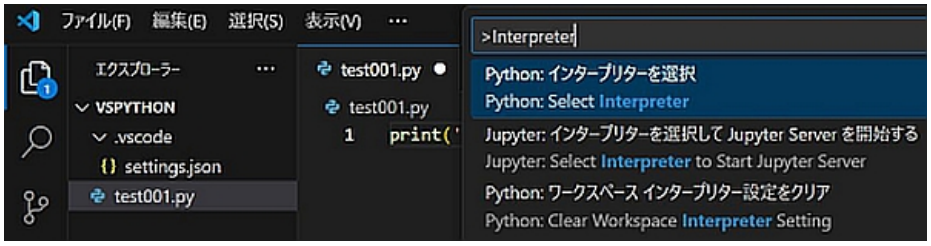


test001.pyの書き込みフィールドに，以下のような一文を記入しましょう．

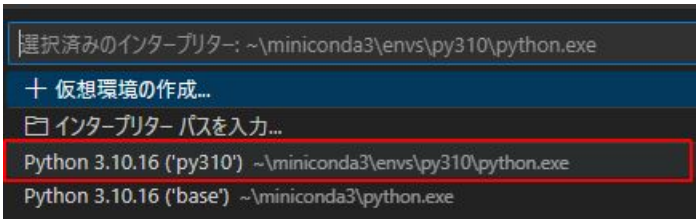
```
print('Hello Python World')
```

そして、次にコンパイルする環境を選択します。再び「表示」-「コマンドパレット」を選択して，検索窓に,"Interpreter " と代入します．





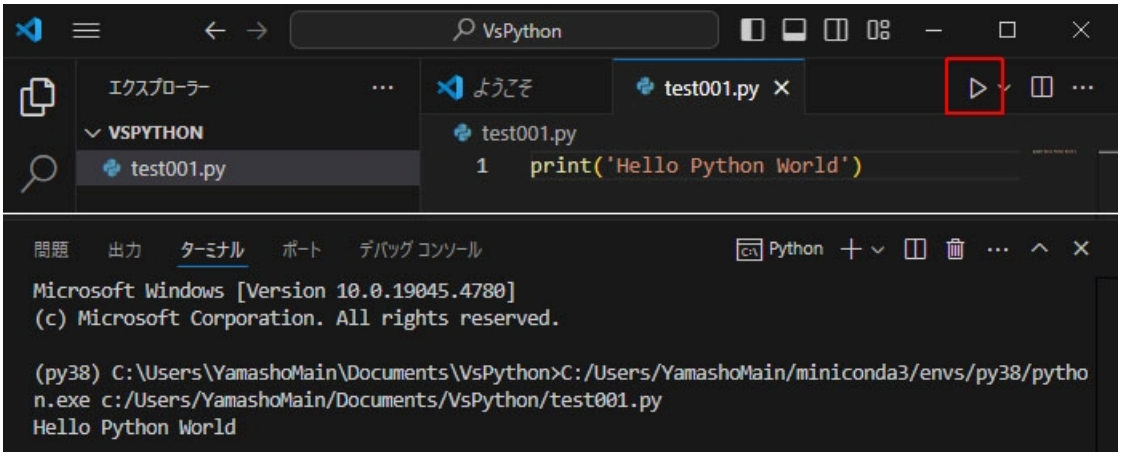
「Python: インタープリター選択」を選ぶと，仮想環境も含めて，Python.exeがインストールされている複数の個所が現れますので，今回，1章で作成した仮想環境「py310」を選択します．



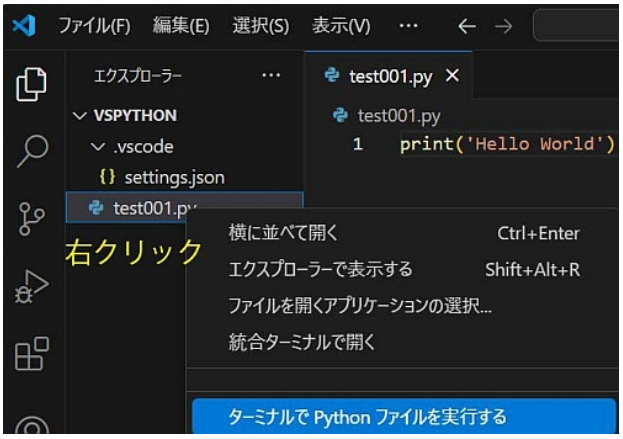
これでプログラムをコンパイル・動作させるまでの設定が終了しました．

2.5 プログラムの動作及びデバッグ環境の確認

では早速，プログラムを動作させていきます。まずは画面右側にある実行マーク▷をクリックします．すると，ターミナル画面が現れ，Hello Python World と表示されます．



VsCodeではもう一つプログラムを実行する手段があります．それはプログラム名を直接，右クリックして，「ターミナルでPythonファイルを実行する」 を選択する方法です．この方法ではデバッガーは機能しませんが，直接，Pythonを動かしています．



どちらの方法でも，実行結果がターミナルに表示されています．

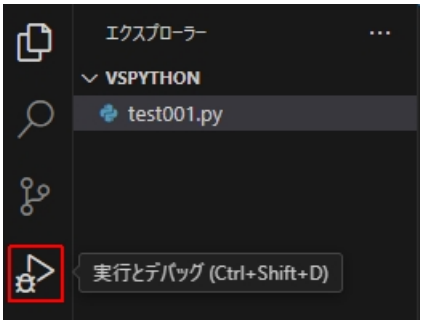
ここまではあくまでもVsCode上でプログラムを記述して，Python.exeで実行しているだけでした．実際のプログラミングではデバッグしながら開発を進めますので，その環境を設定します．

VsCodeでPythonプログラミングを実行するための設定ファイルは以下の通りです．

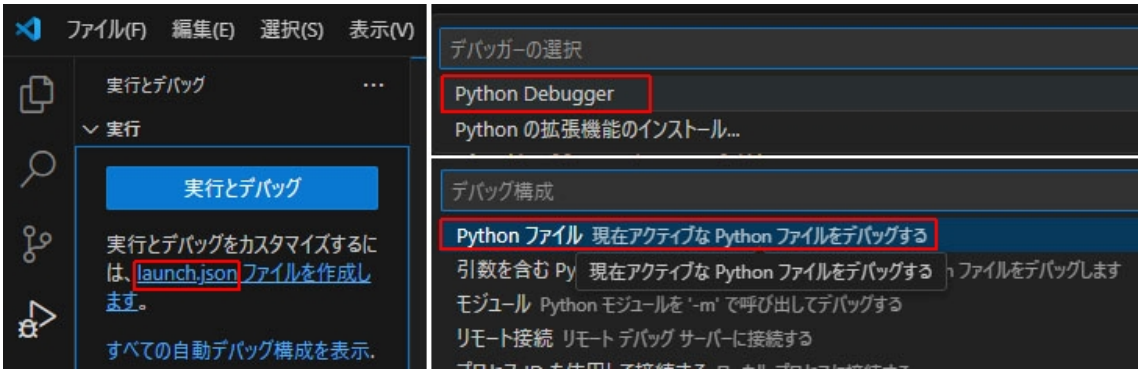
- settings.json：VSCodeの設定を記述するファイルです
- tasks.json：一通りのビルドに関する設定を指定します
- launch.json：デバッグに関する設定を指定します

なお、VSCodeにはユーザ設定とワークスペース設定がありますので注意が必要です。
・ユーザ設定：どのワークスペースでも共通の設定
・ワークスペース設定：特定のワークスペースのみでの設定

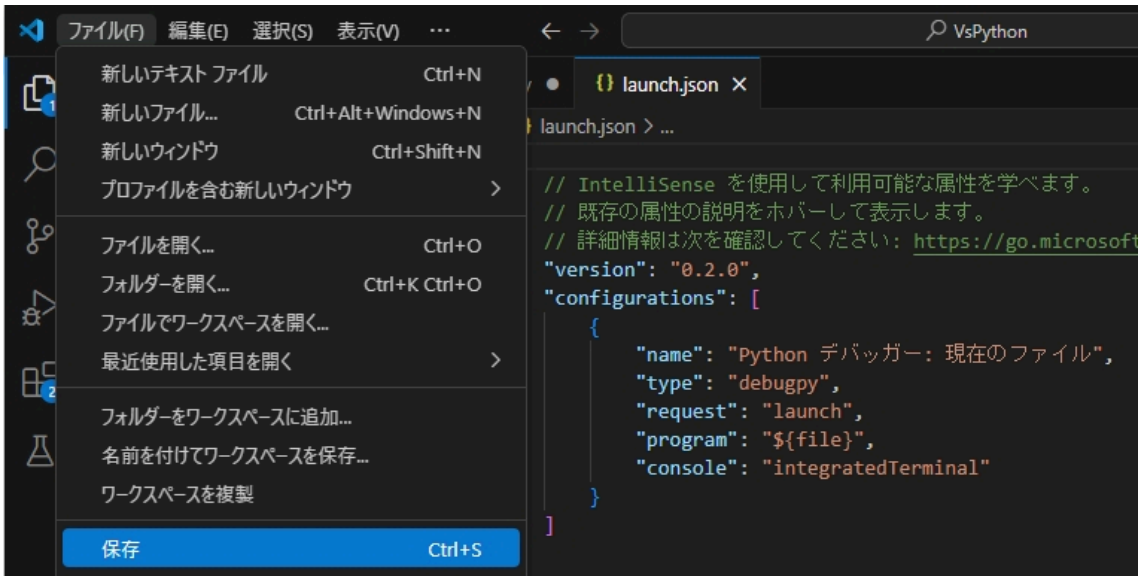
では，test001.pyが選択されている状態で，左側にあるデバッグ開始ボタンを押してみましょう．



すると，「実行とデバッグ」というウィンドウが開きますので，その下の，”launch.json” をクリックします．デバッガーの選択画面で， “Python Debugger” を選択して，次の画面で， “Python ファイル” を選択します．

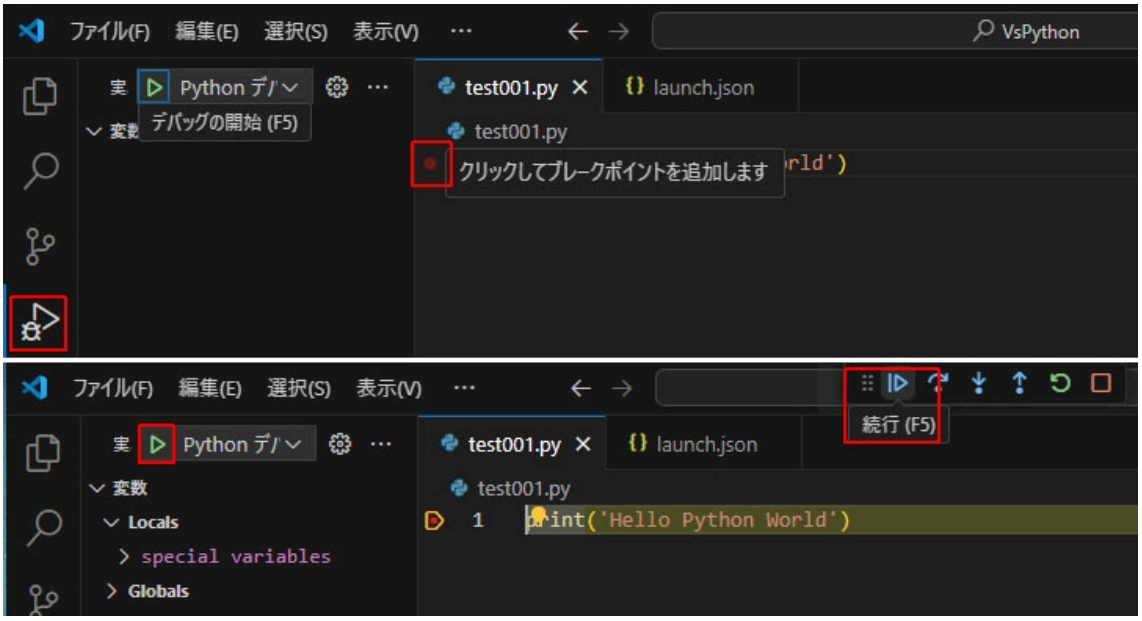


すると，launch.jsonが自動的に生成されました．ここでは特に中身はカスタマイズしません．そのまま保存しましょう．



それではもう一度，左側にあるデバッグ開始ボタンを押しましょう．すると先ほどとは違った画面構成になりました．それではデバッグの動作を確認するために，test001.pyのprint文の頭をクリックしてブレークポイントを設定します．

そして，緑の矢印をクリックすると，デバッガーが動作して，print文のところまで実行してくれます．このようにデバッグは特定の位置でプログラムを中断して，内部変数の変化やメモリの状態を確認していく作業となります．ブレークポイントを超えて続行するには水色の矢印をクリックしていきます．



ここまでがデバッガの設定と動作確認です．なお，デバックしながら実行するときは，「実行」-「デバッグの実行」を選択しても実施できます．また，F5ボタンでも実行できます．

但し，デバッガーを起動するとプログラム自体は重くなりますので，不明なエラーが出ない限り，通常のコンパイル/実行（画面右側にある実行マーク▷又は右クリック＋ターミナルでPythonファイルを実行する）だけでOKです．

2.6 応用プログラミングの動作確認

この先，ゼミ授業でデータサイエンスと画像処理を勉強してきます．そのために1章ではScikit-learnとOpenCVのライブラリを導入してきました．その外部ライブラリが動作しているかをここでは確認していきましょう．

まず，Scikit-learnについては以下のプログラムを書き込み，実行してみましょう．

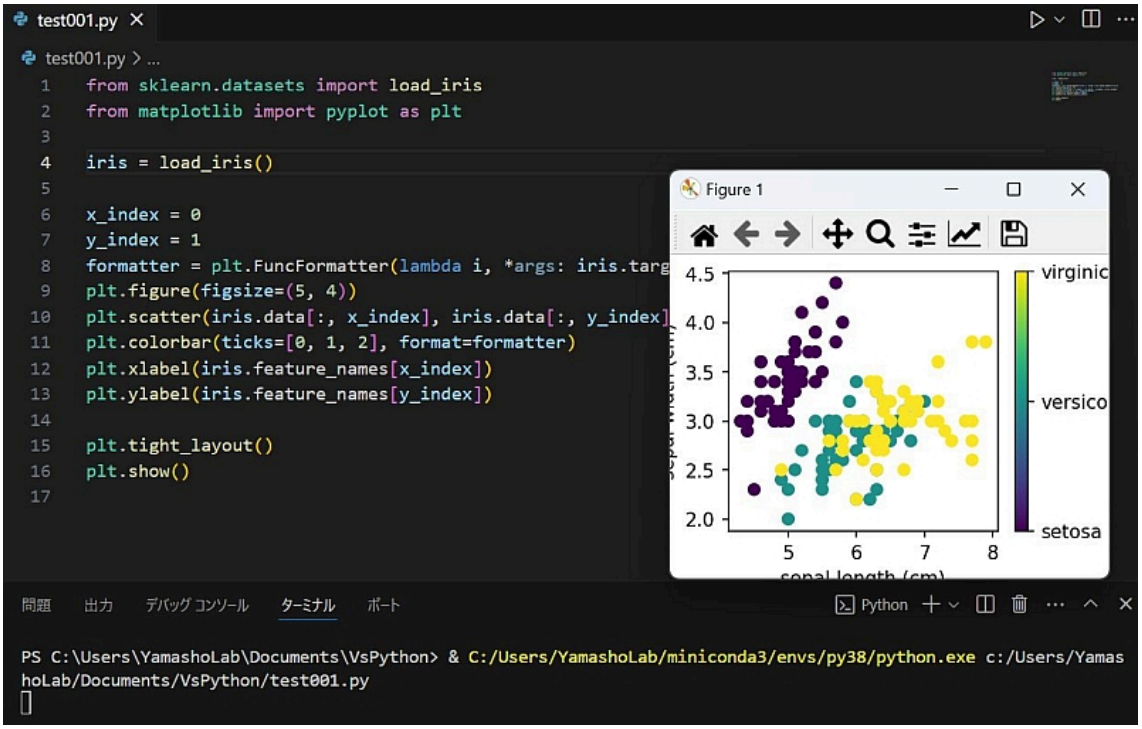
```
from sklearn.datasets import load_iris
from matplotlib import pyplot as plt

iris = load_iris()

x_index = 0
y_index = 1
```

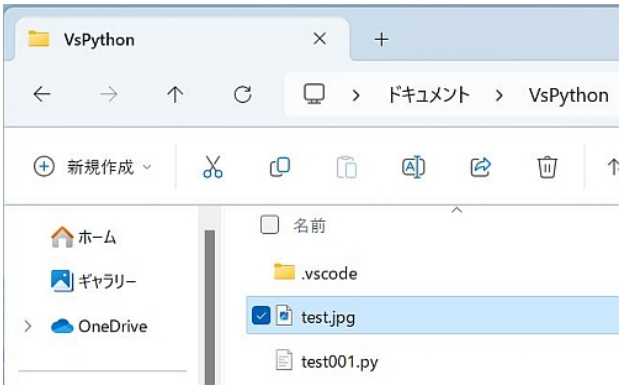
```
formatter = plt.FuncFormatter(lambda i, *args: iris.target_names[int(i)])
plt.figure(figsize=(5, 4))
plt.scatter(iris.data[:, x_index], iris.data[:, y_index], c=iris.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel(iris.feature_names[x_index])
plt.ylabel(iris.feature_names[y_index])

plt.tight_layout()
plt.show()
```



無事，グラフまで表示できたなら，ライブラリのインストールはうまくいっています．

続いて，OpenCVの動作確認ですが，先に何でもよいので，test.jpg ファイルを作成して，プログラムteest001.pyと同じフォルダに入れておきましょう．



そして，以下のプログラムを実行してみます．

```
import cv2

img = cv2.imread('./test.jpg',1)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```