Raspberry Pi勉強会

*SBS*情報システムのアリフ・マウラナ

内容

- Raspberry Pi紹介
- AI画像認証紹介
- python準備
- Python virtual environment
- Visual Studio Code 準備
- OpenCV準備とWeb Cameraテスト
- 顔認識

ノート：

エラー❗＝エラーが発生した場合

■　ターミナルコマンド

●　Pythonコード

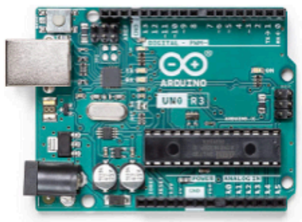
Raspberry Pi (1/3)

- Raspberry Pi は、クレジット カード サイズの低コストのコンピュータで、コンピュータ モニタまたはテレビに接続し、標準のキーボードとマウスを使用します。これは、あらゆる年齢層の人々がコンピューティングを探索し、Scratch や Python などの言語でプログラミングする方法を学ぶことができる、有能な小型デバイスです。インターネットの閲覧や高解像度ビデオの再生から、スプレッドシートの作成、ワードプロセッサ、ゲームに至るまで、デスクトップ コンピュータに期待されることはすべて実行できます。

Raspberry Pi (2/3)

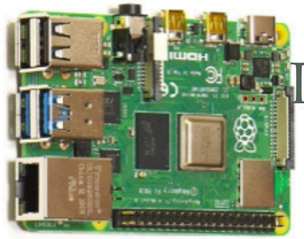
- パーソナル コンピューターとの主な違いは、Raspberry Pi には、外界を制御し、通信するためのオープン GPIO ポートがあり、音楽マシンや親検出器から気象観測所や赤外線カメラを備えたさえずり巣箱に至るまで、さまざまなデジタル メーカー プロジェクトで使用されてきたことです。
- オープン GPIO ポートを持つ他のデバイスと比較すると、Raspberry Pi はミドルクラスのデバイスです。

2 千–7千円
CPU 8-32bit



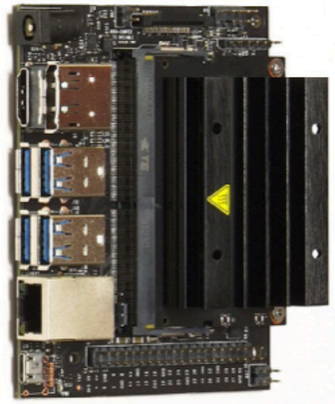
Arduino

1万-1万7千円
CPU 32-64bit



Raspberry Pi

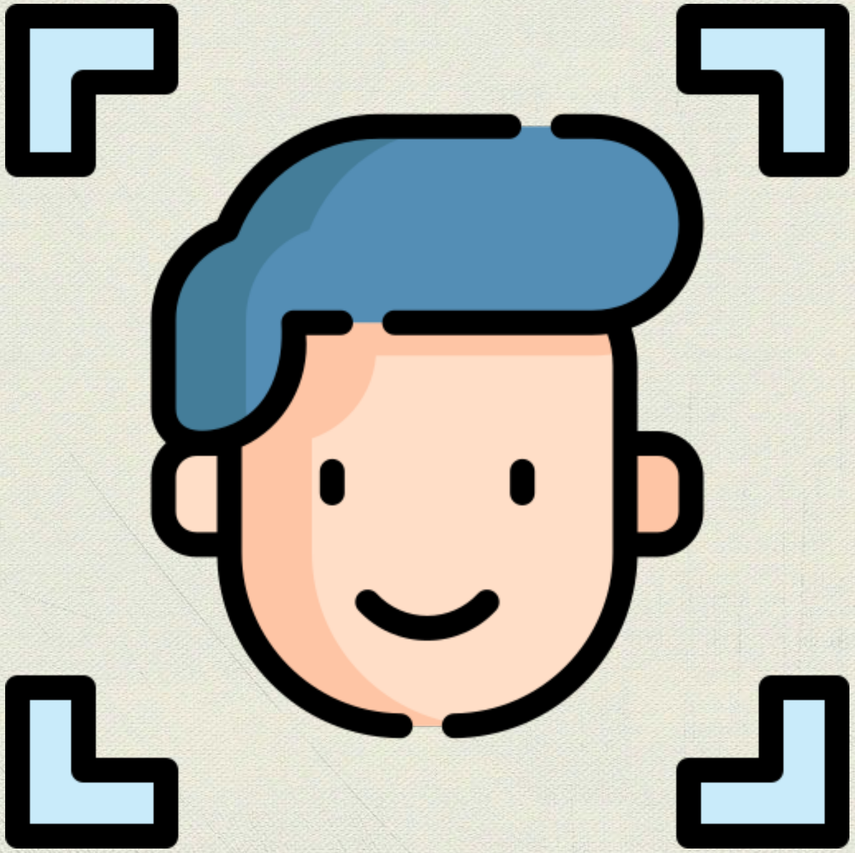
2万4千-3万円
CPU 64bit + GPU



Jetson Nano

Raspberry Pi (3/3)

- 今回は授業の始まりなので、Raspberry Pi について紹介したいと思います。 Raspberry Piではどのようなプログラム環境が動作するのでしょうか？このクラスは、Raspberry Pi と同様の環境をコンピューター内に作成します。
- 顔認識ライブラリを使用して顔を認識しますが、これはやりがいがあって楽しいものです。



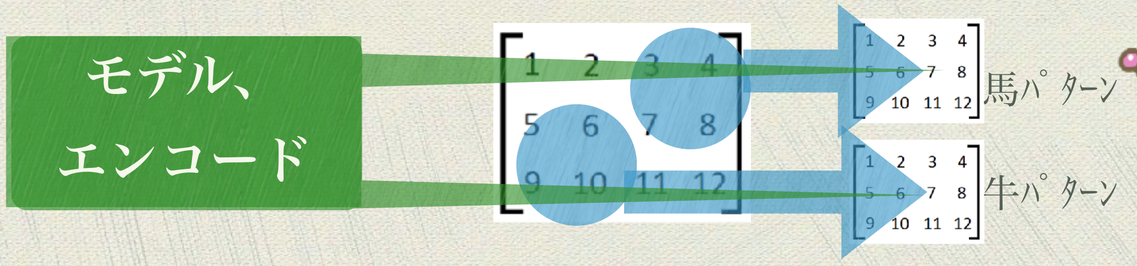
AIの基礎

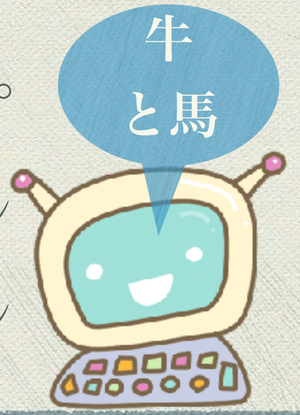
- パソコンは画像ではなくて行列形です、牛と馬が見えないですが、

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- 読み取りが速く、正確で、忘れることがないです。

- 認識方法は、パターンを認識し、人間が理解できる単語や文章に変換する数学的プロセスです。



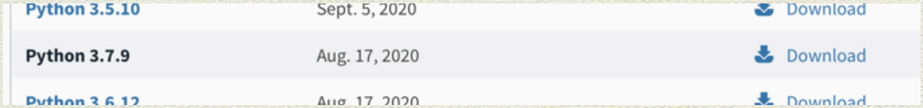


Python v3.7.9準備 (1/3)

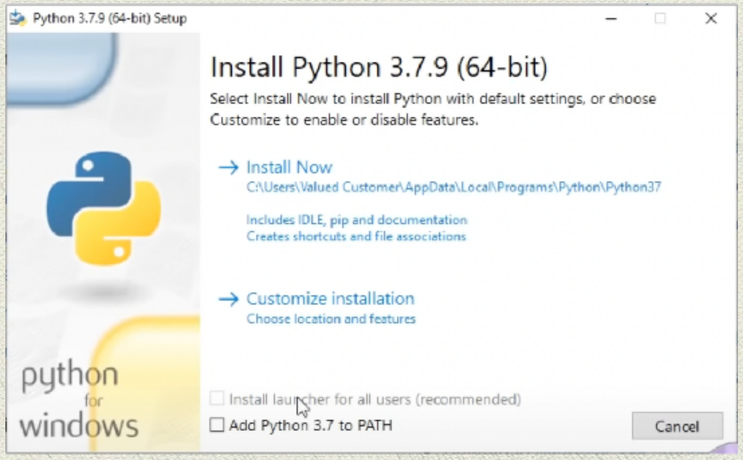
- エンジンでPython Downloadを実行してください

https://www.python.org/downloads/

- Pythonをダウンロードします。

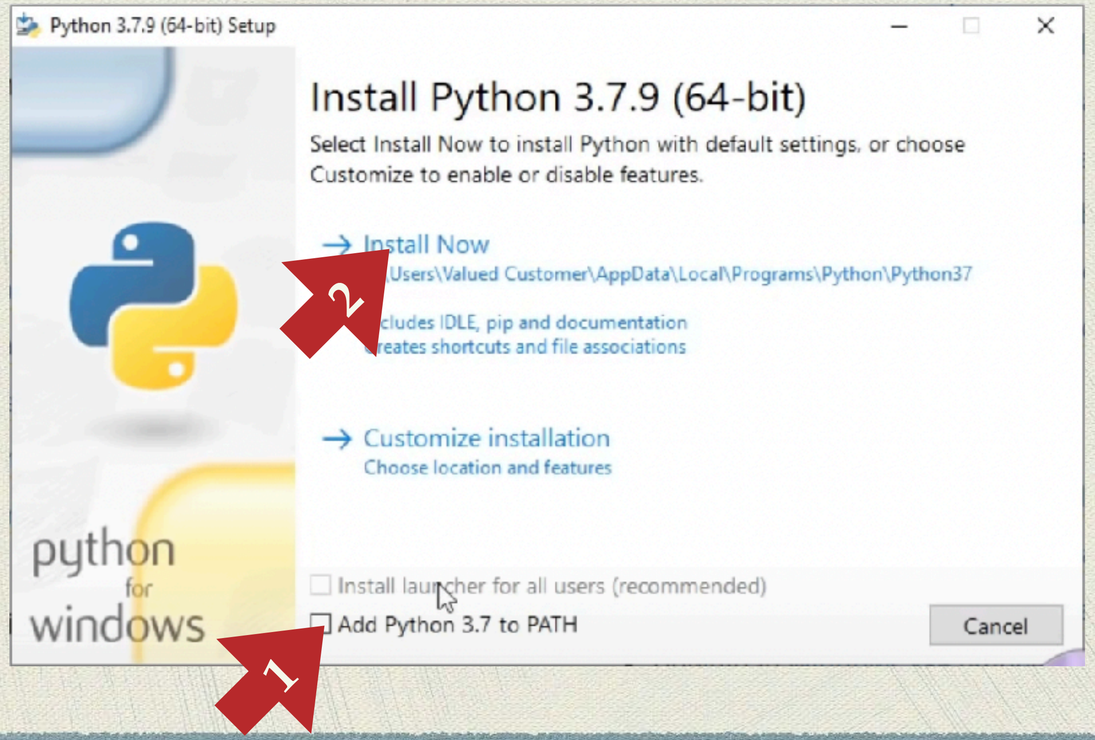


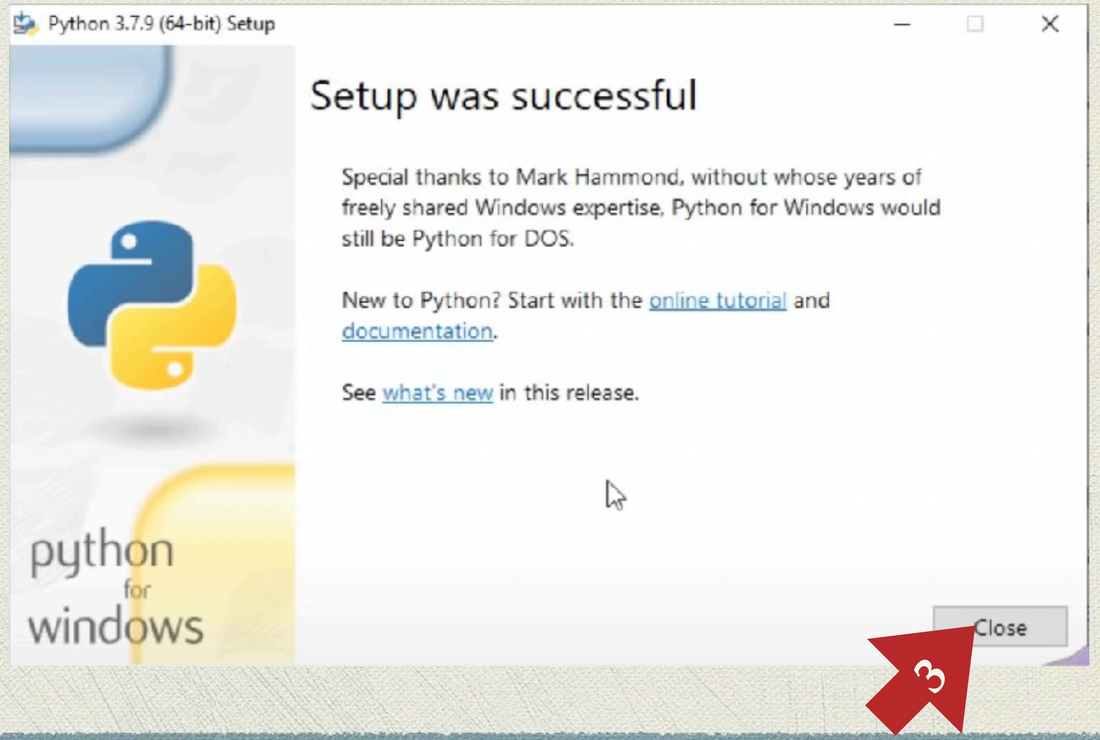
- Pythonをインストール



Python v3.7.9準備 (2/3)

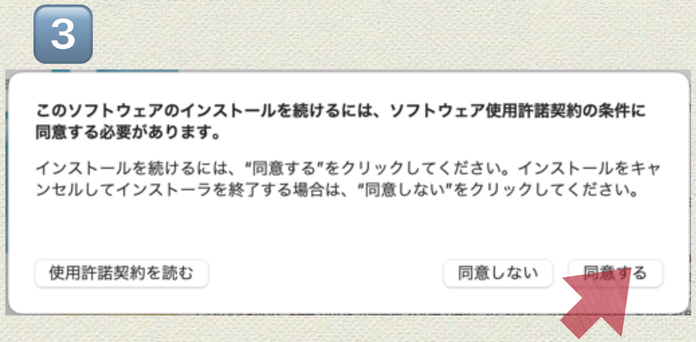
- Add Python 3.7 to Path をチェックしてインストールしてください。 ☑ Add Python 3.7 to PATH

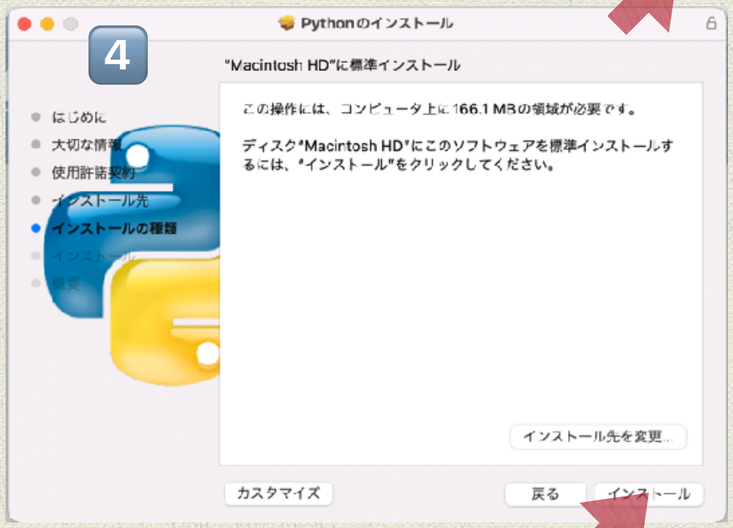




Python v3.7.9準備 (2/3) (Mac)

自分のパソコンで実行希望の方





Python 準備 3/3

- WindowsのPowerShellで確認

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Valued Customer> python
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- 複数pythonがインストールしたら、pyで最新pythonバーションを実行する

```
PS C:\Users\Valued Customer> py
Python 3.9.6 (tags/v3.9.6:db3ff76,
Type "help", "copyright", "credits"
>>> quit()
```

`quit()`を書いてPythonを終了

Python Virtual Environment (1/2)

- 作業フォルダを準備、ターミナルで
- ■mkdir python　（こちらは作業フォルダーとなります）
- ■cd python
- ■dir
  空であることを確認してください。
  ディレクトリの別の名前と場所を使用して、必要に応じて仮想環境を作成できます。

Python Virtual Environment (2/2)

- 環境フォルダーを作成して設定します。
  - ■ `python -m venv pyrasppi`
- To activate (有効化)

  Raspberry PI / Windows : ■ `pyrasppi/Scripts/activate`
  Mac: ■ `source pyrasppi/bin/activate`
- Error ❗ : Windows: ■ `set-executionpolicy remotesigned -Scope CurrentUser`
  “Yes to All”を選択します

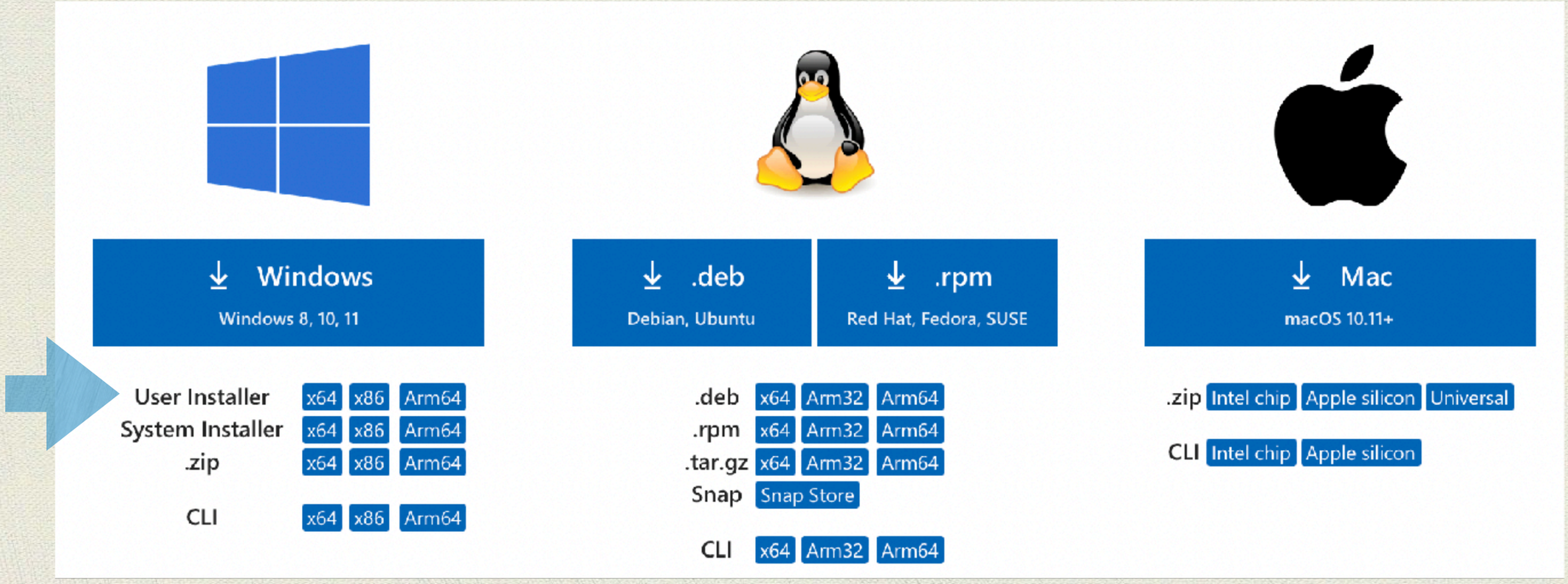
  もし前の状況を戻したい場合：
  - ■ `set-executionpolicy restricted -Scope CurrentUser`
- To deactivated （無効化）
  - ■ `deactivate`

Visual Studio Codeを準備(1/7)

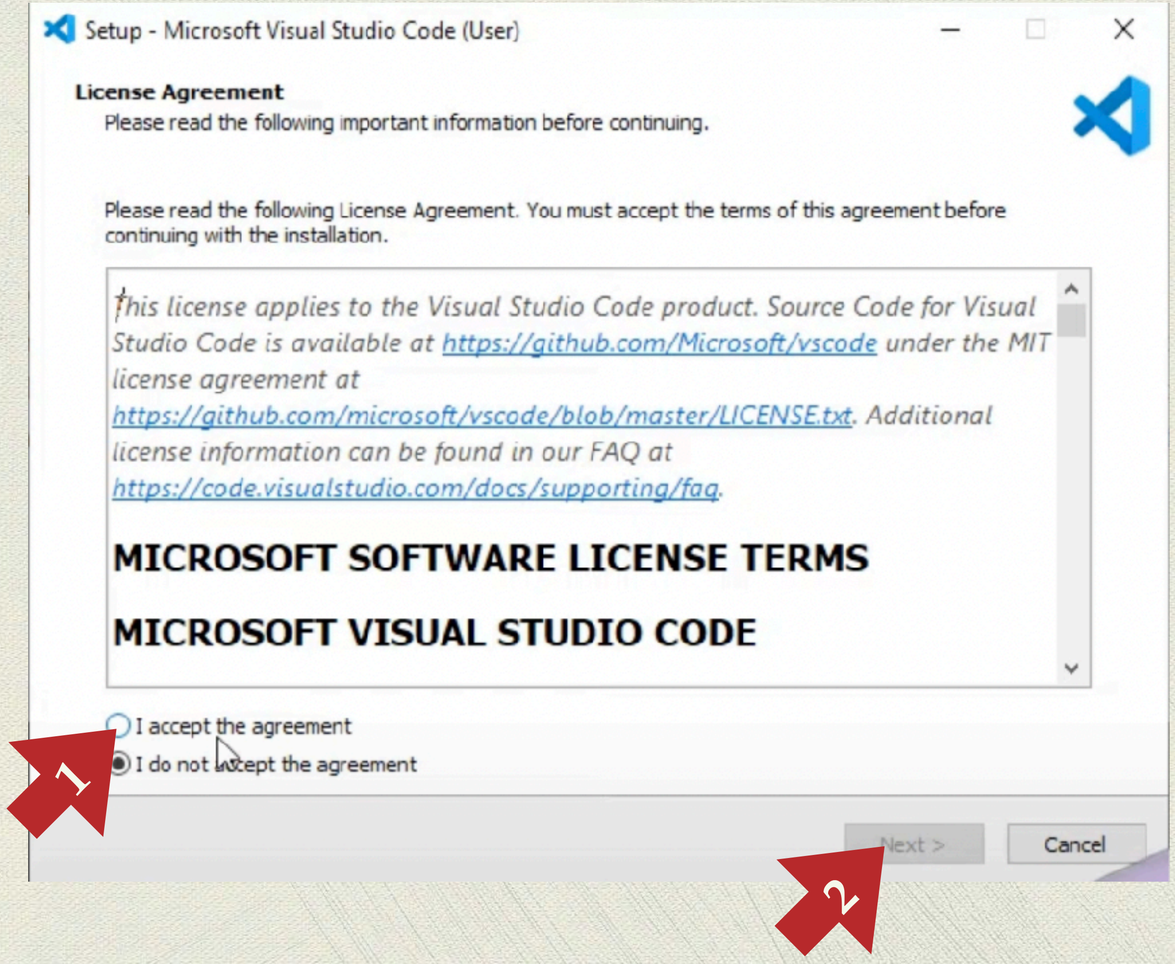
- インターネットのサーチエンジンでDownload visual studio codeで検索してください

Visual Studio Codeを準備(2/7)

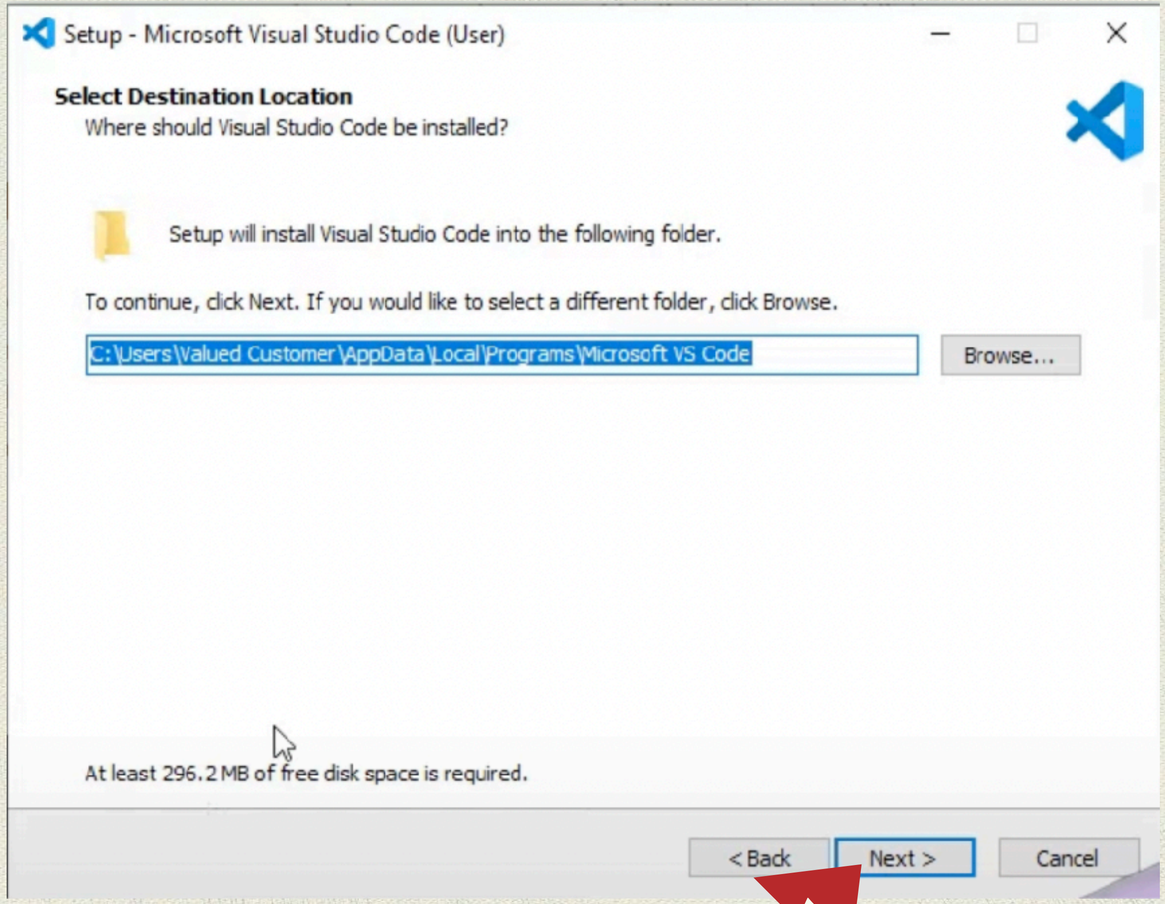
- WindowsならUser Installerバーションでダウンロードしてください。

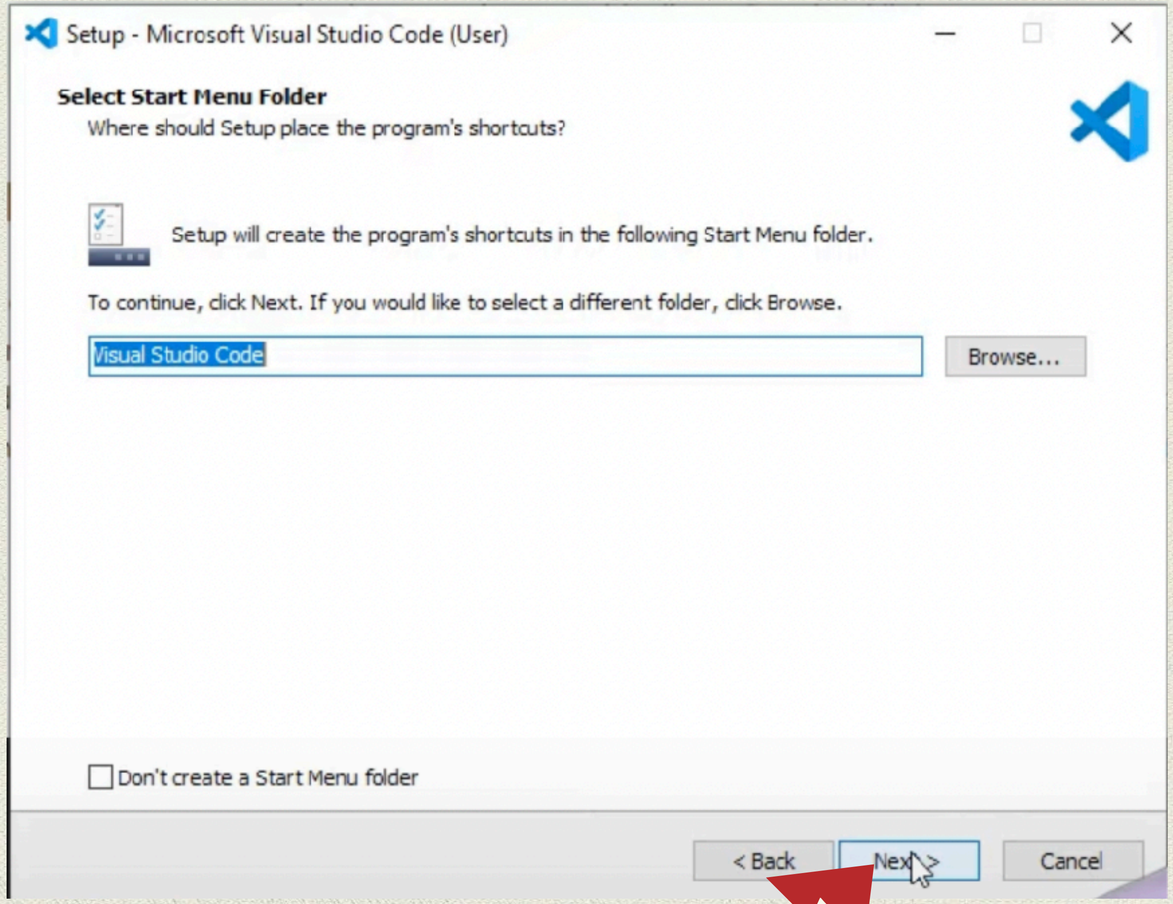


Visual Studio Codeを準備(3/7)

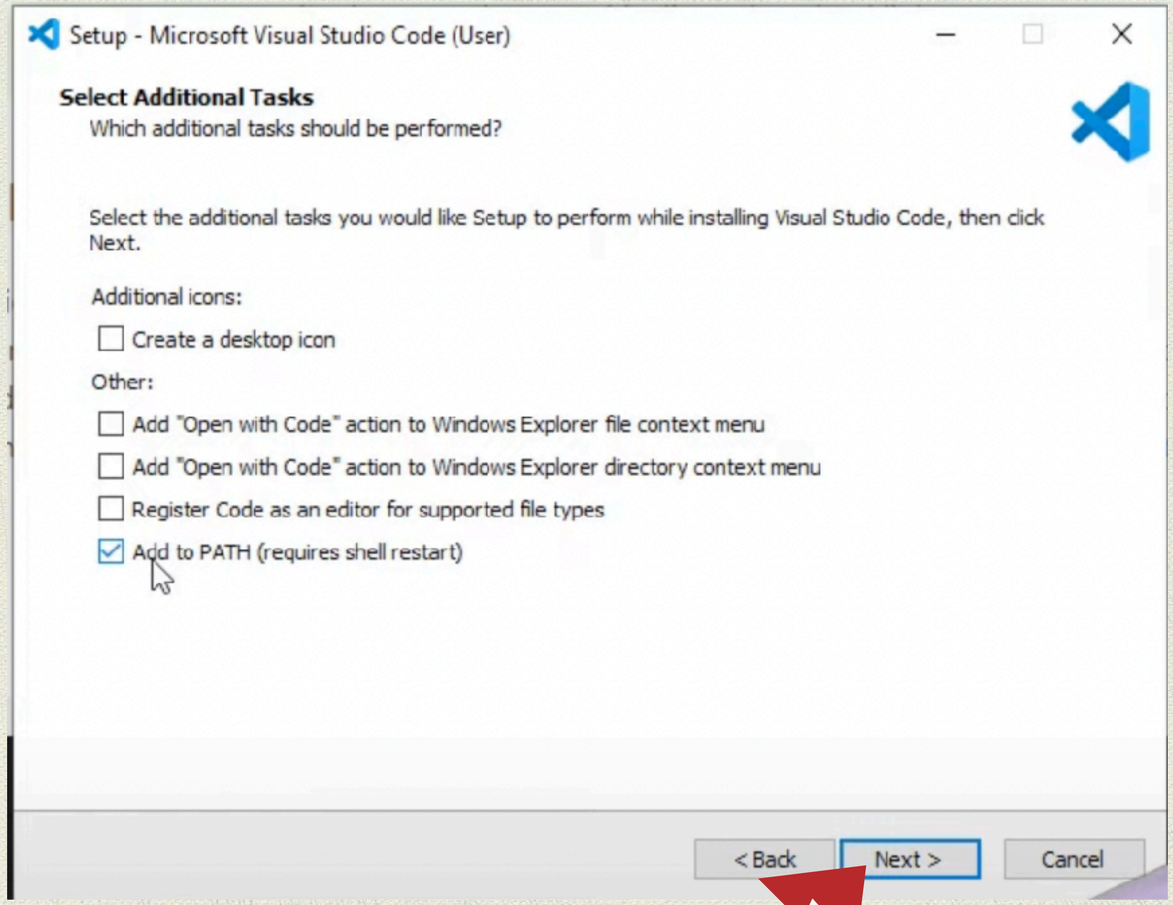


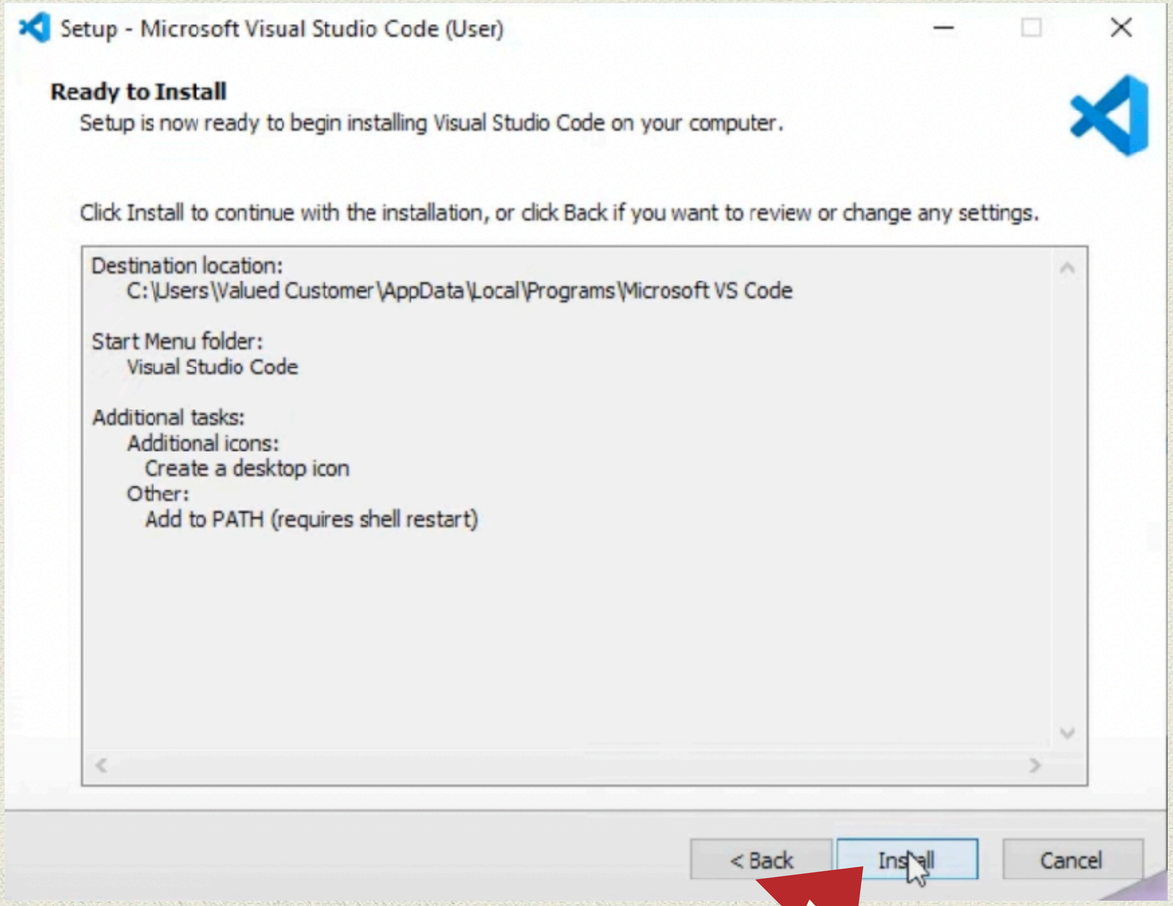
Visual Studio Codeを準備(4/7)



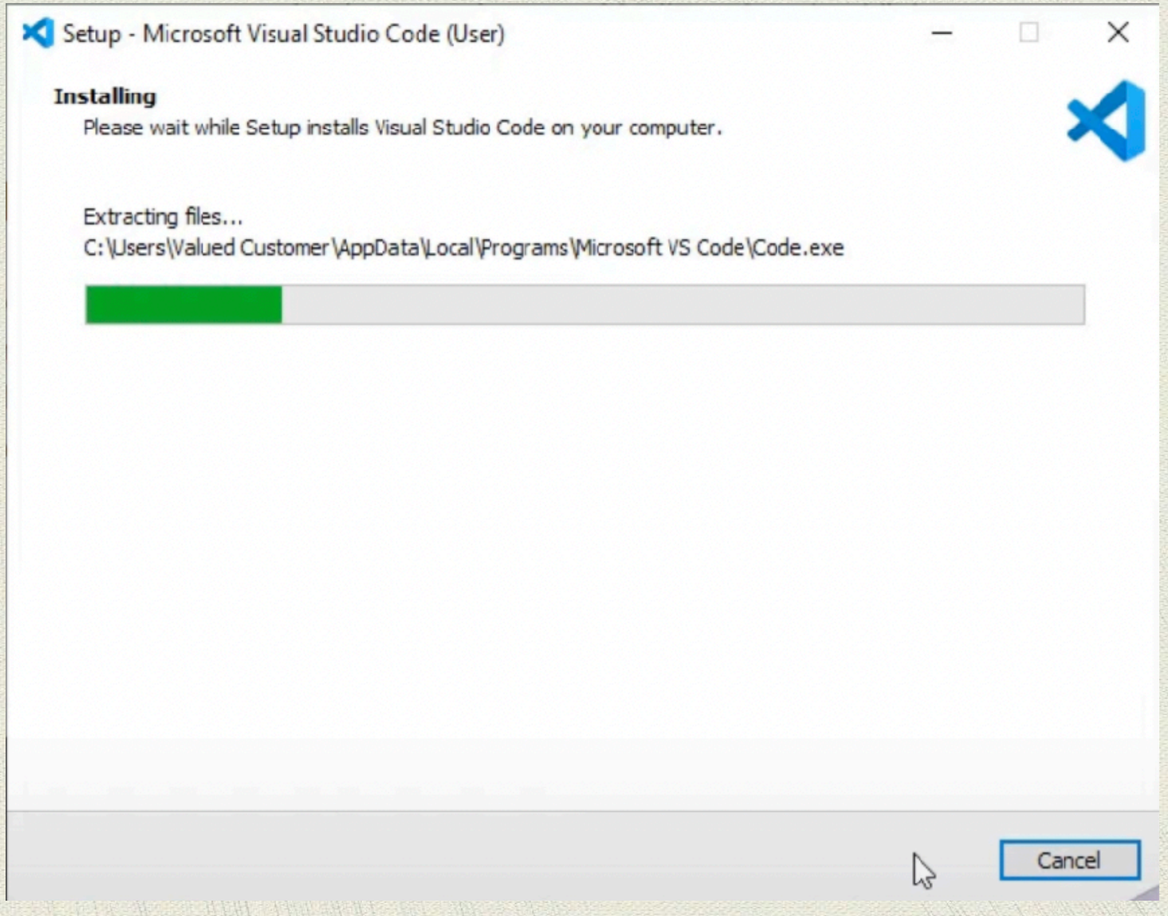


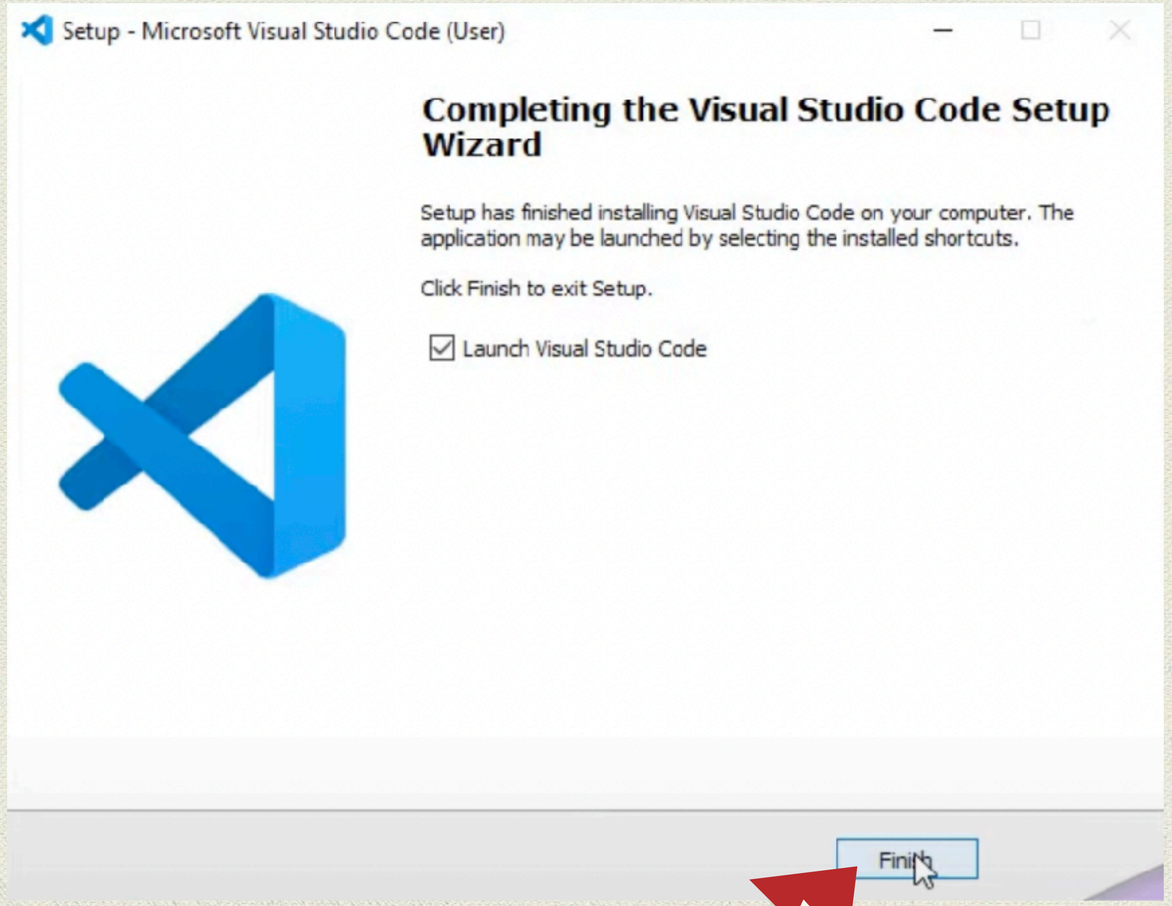
Visual Studio Codeを準備(5/7)



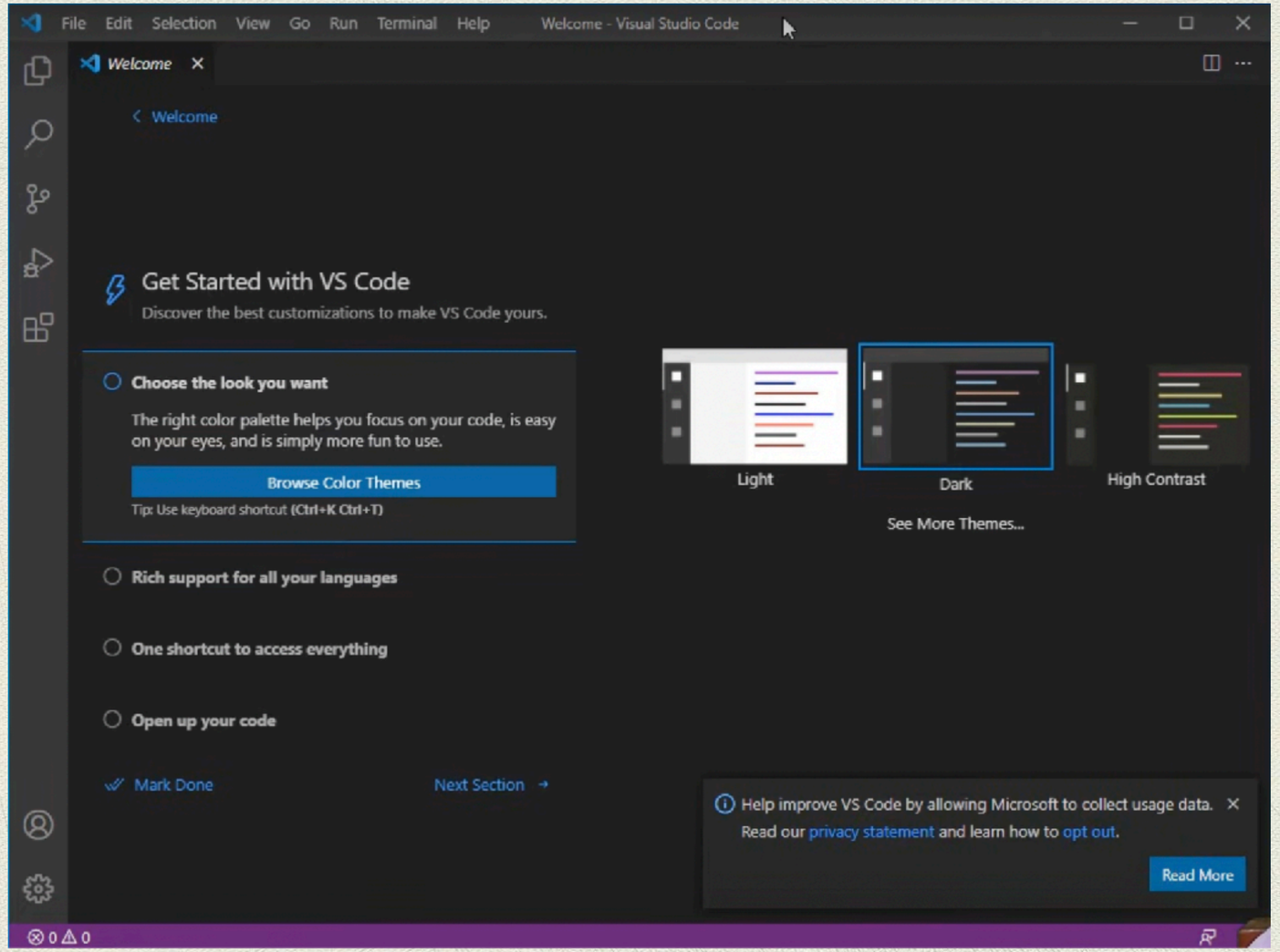


Visual Studio Codeを準備(6/7)





Visual Studio Codeを準備(7/7)



準備完了

OpenCV準備とWebcamテスト(1/6)

- 作業フォルダーにpython3.7.9を有効化します。

- OpenCVをインストールします：
  - `pip install opencv-python`
- Pythonターミナルで OpenCVを確認する：

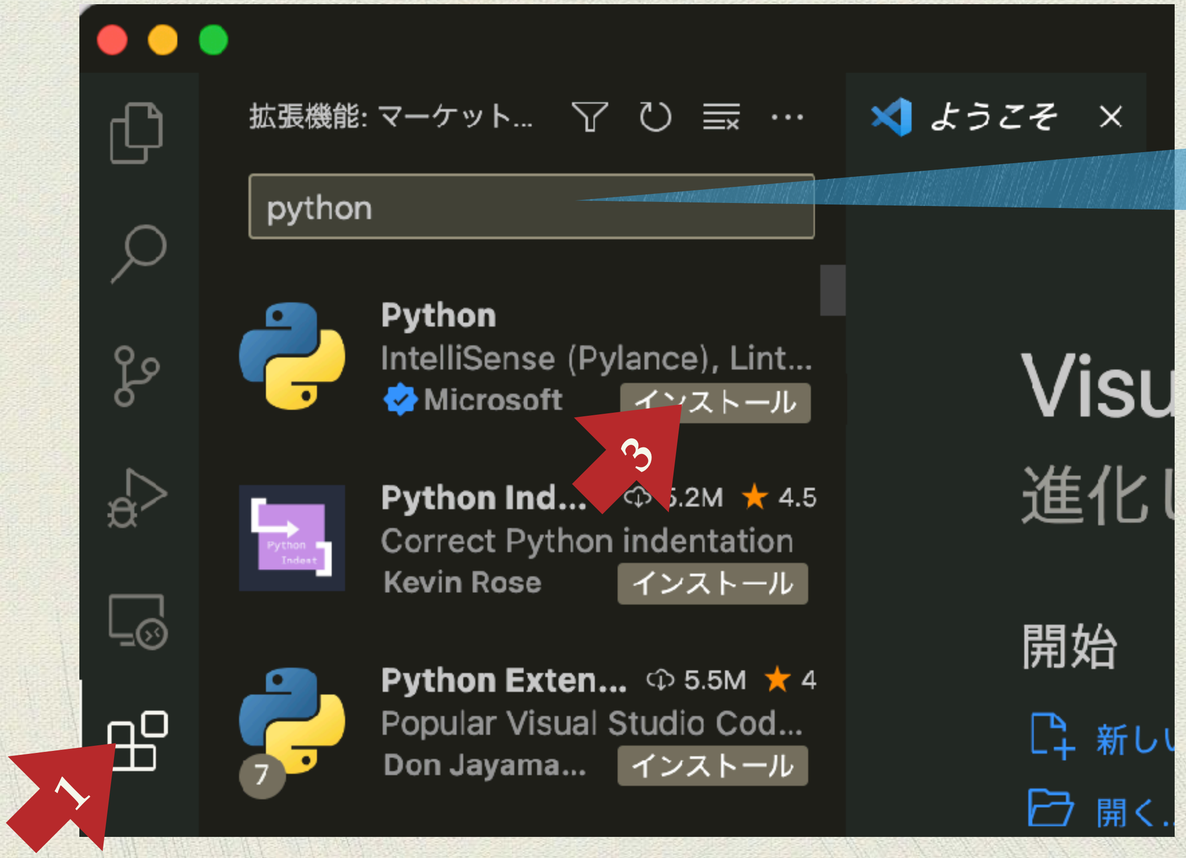
  `python` 実行
  - `import cv2`
- CV2バーションを確認：
  - `print(cv2.__version__)`

```
(pyrasppi) (base) s118301@Neutrino python % python
Python 3.7.9 (v3.7.9:13c94747c7, Aug 15 2020, 01:31:08)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>>
```

`quit()` を書いてPythonを終了

OpenCV準備とWebcamテスト(2/6)

- Visual studio codeにpython interpreterをインストールします



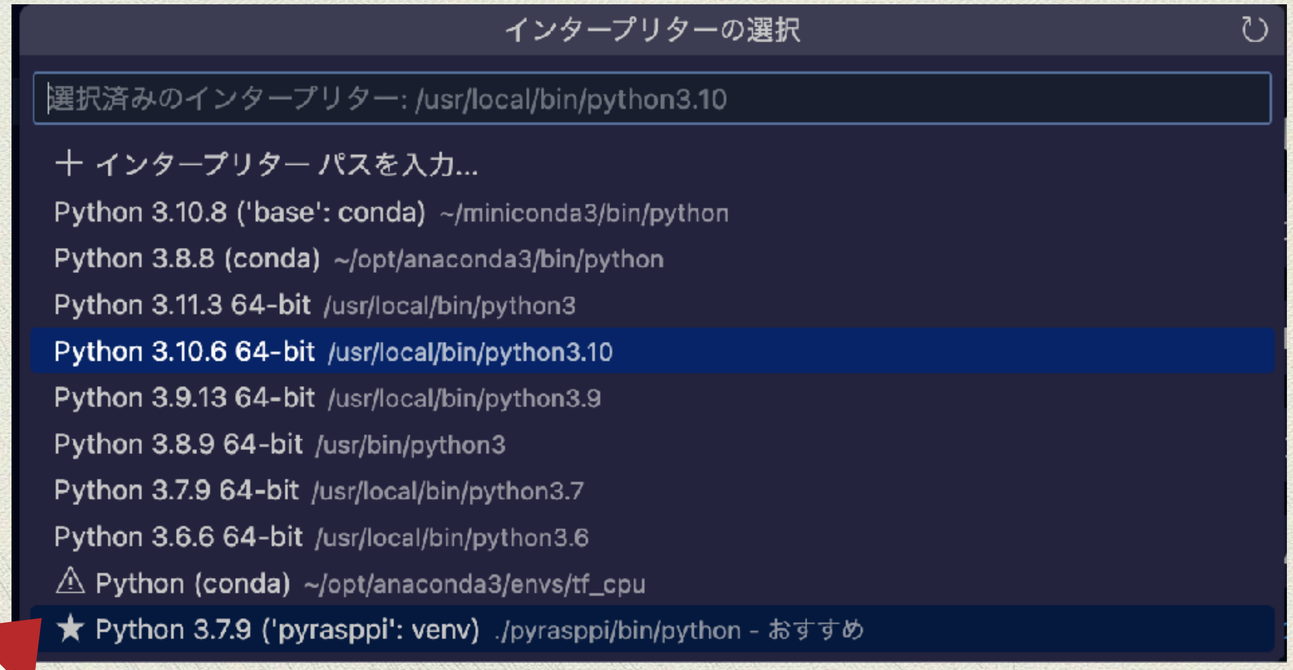
2. ここにpythonを書いてください

OpenCV準備とWebcamテスト(3/6)

- Visual Studio CodeにVirtual pythonを設定

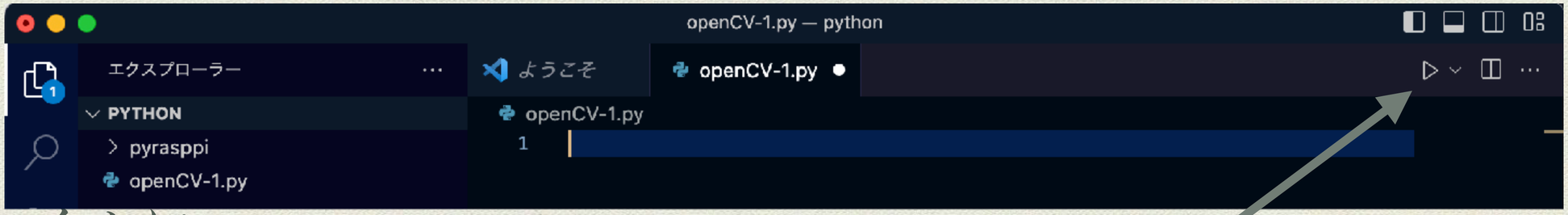
  Visual Studio Codeを開けて、作業フォルダに行って

  Ctrl+Shift+P
  Python: Select Interpreter



OpenCV準備とWebcamテスト(4/6)

- 新規ファイルを追加してください（openCV-1.py）



- ファイルに：
  - import cv2
  - print(cv2.__version__)
- 実行（確認するため）右上にある▶️アイコンをクリックしてください

OpenCV準備とWebcamテスト(5/6)

- カメラを設定、こちらのコマンドを追加してください

```
cam=cv2.VideoCapture(0)
```

- 無限ループを作成し、フレームを取得して表示します。

```
while True:
  ignore, frame = cam.read()
  cv2.imshow(‘my WEBcam’, frame)

  if cv2.waitKey(1) & 0xff == ord(‘q’):
    break

cam.release()
```

- コードを実行してください

```
import cv2
print(cv2.__version__)
cam=cv2.VideoCapture(0)
while True:
    ignore, frame = cam.read()
    cv2.imshow('my WEBCam', frame)
    if cv2.waitKey(1) & 0xff == ord('q'):
        break
cam.release()
```

OpenCV準備とWebcamテスト(6/6)

- AI プロセスをどこに配置するか?

```
1   import cv2
2   print(cv2.__version__)
3   cam=cv2.VideoCapture(0)
4   while True:
5       ignore, frame = cam.read()
6
7
8
9       cv2.imshow('my WEBCam', frame)
10      if cv2.waitKey(1) & 0xff == ord('q'):
11          break
12  cam.release()
```

画像を撮る

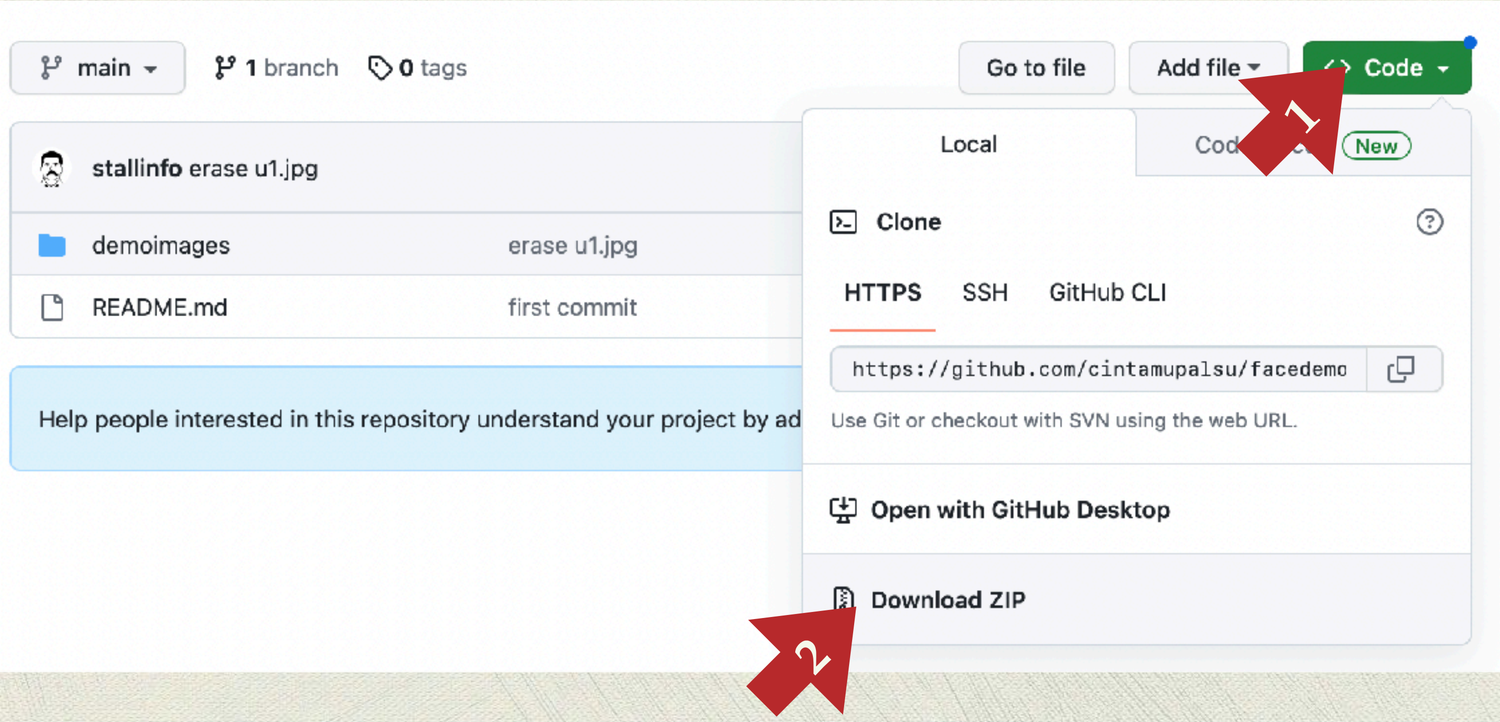
この間にAIプロセス
を追加します

画像を表示する

顔認証 (1 / 15)

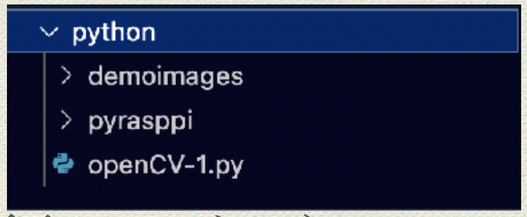
- こちらのURLからフォルダーをダウンロードしてください。

https://github.com/cintamupalsu/facedemoimages



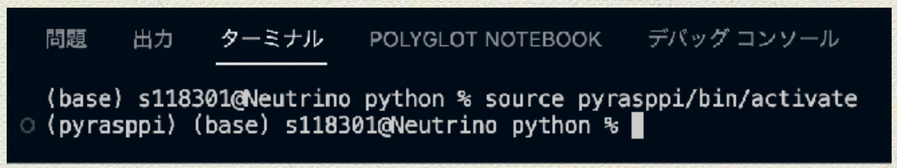
顔認証 (2/15)

- Demoimagesフォルダーをワーキングフォルダーにコピーしてください



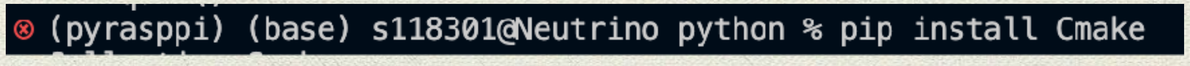
- Python virtual environmentがアクティブであることを確認してください。

  Windows: ■`pyrasppi/Scripts/activate`
  Mac: source ■`pyrasppi/bin/activate`



- Cmakeをインスールしてください。

  pip install Cmake
  エラー❗: sudo pip install Cmake



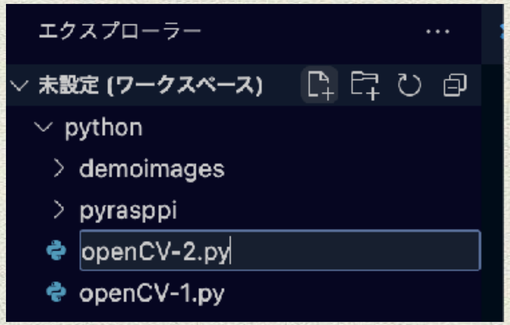
顔認証 (3/15)

- 顔認識ライブラリーをインストールしてください。

  ■`pip install face_recognition==1.2.3`

- エラー❗:■`sudo pip install  face_recognition==1.2.3`

- 新規Pythonファイルを追加してコードを書いて始めましょう



- それでもエラーが発生する場合は、このサイトにアクセスしてアドオンをダウンロードしてください。
- https://visualstudio.microsoft.com/visual-cpp-build-tools/

顔認証 (4/15)

- OpenCVとFace Recognitionライブラリーをインポートしてください。
  - `import cv2`
  - `import face_recognition as FR`

face_recognitionと
いう単語が長いのでFRで短くして使用
する方です

- 人の名前を表示するためFontスタイルを設定してください
  - `font=cv2.FONT_HERSHEY_SIMPLEX`

```
import cv2
import face_recognition as FR
font=cv2.FONT_HERSHEY_SIMPLEX
```

- 一回実行してエラー無しを確認してください。

```
(pyrasppi) (base) s118301@Neutrino python % /Users/s118301/Documents/python/pyra
sppi/bin/python /Users/s118301/Documents/python/openCV-2.py
(pyrasppi) (base) s118301@Neutrino python %
```

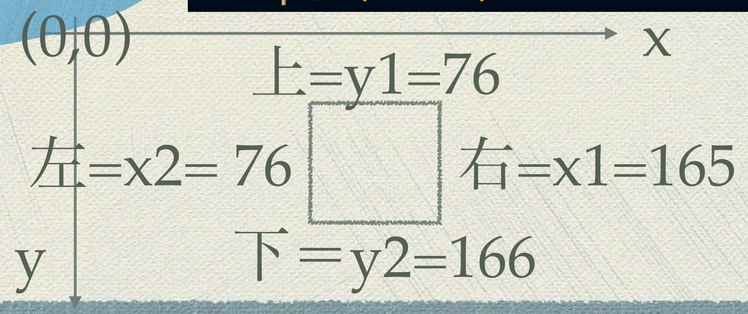
顔認証 (5/15)

- サンプル顔画像をロードしましょう。（今回は岡村さんのです）
  - `okamuraKao=FR.load_image_file('<file_path>')`
- ロードした画像の中に顔が有るかどうか確認するためこちらのコマンドを追加してください。
  - `faceLoc=FR.face_locations(okamuraKao)`
  - `print(faceLoc)`

見つけた顔の直交座標を表示する

```
import cv2
import face_recognition as FR
font=cv2.FONT_HERSHEY_SIMPLEX
okamuraKao=FR.load_image_file('/Users/s118301/Documents/python/demoimages/known/okamura_hisanori.jpg')
faceLoc=FR.face_locations(okamuraKao)
print(faceLoc)
```

- 実行してください。




```
(pyrasppi) (base) s118301@Neutri
/Documents/python/openCV-2.py
[(76, 165, 166, 76)]
```

([上、右、下、左])

顔認証 (6/15)

- 画像の中に複数顔がある可能性があるので、データは複数の値を保持できるArray型となります。
  今回は最初の最初に見つけた顔を表示するため、

  line 5: `faceLoc=FR.face_locations(okamuraKao)[0]`

  [0]を追加してください

- 画像で確認しましょうこちらのコマンドを追加してください
  - `cv2.imshow('my Window', okamuraKao)`
  - `cv2.waitKey(5000)`

  5秒間画像を表示して消える

  その後実行してください

```
1  import cv2
2  import face_recognition as FR
3  font=cv2.FONT_HERSHEY_SIMPLEX
4  okamuraKao=FR.load_image_file('/Users/s1183
5  faceLoc=FR.face_locations(okamuraKao)[0]
6  print(faceLoc)
7  cv2.imshow('my Window',okamuraKao)
8  cv2.waitKey(5000)
```

- これを見ると色がちょっと変ですね。

顔認証 (7/15)

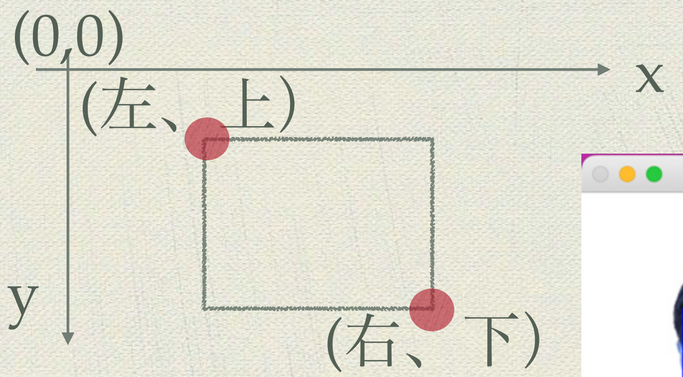
- 顔の周りに線を描きましょう。line 6: `print(faceLoc)` の下にこちらのコマンドを追加しましょう。

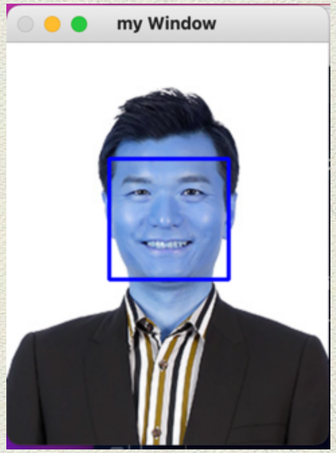
色(青,緑,赤)

- `top, right, bottom, left=faceLoc`
- `cv2.rectangle(okamuraKao,(left, top),(right, bottom),(255,0,0),2)`

(左、上)　(右、下)

線の厚さ





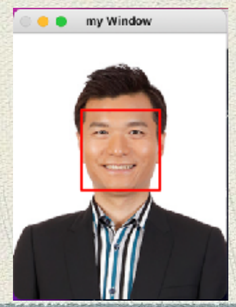

```
1   import cv2
2   import face_recognition as FR
3   font=cv2.FONT_HERSHEY_SIMPLEX
4   okamuraKao=FR.load_image_file('/Users/s118301/Documents/python/de
5   faceLoc=FR.face_locations(okamuraKao)[0]
6   print(faceLoc)
7   top,right,bottom,left=faceLoc
8   cv2.rectangle(okamuraKao,(left,top),(right,bottom),(255,0,0),2)
9   cv2.imshow('my Window',okamuraKao)
10  cv2.waitKey(5000)
11
```

実行してください

顔認証 (8 / 15)

- ほとんどの画像ファイルは RGB 形式で動作しますが、OpenCV は BGR 形式で動作します。OpenCV を使用して画像を表示する前に、RGB から BGR に変換する必要があります。こちらのコマンドをline 8: `cv2.rectangle(okamuraKao,(left,top),(right,bottom),(255,0,0),2)`の下に追加してください。
  - `okamuraKaoBGR=cv2.cvtColor(okamuraKao,cv2.COLOR_RGB2BGR)`
- そして、次のコマンとに表示するのはokamuraKaoではなくてokamuraKaoBGRを変更してください。
  - `cv2.imshow('my Window',okamuraKaoBGR)`

その後、実行して確認してください。



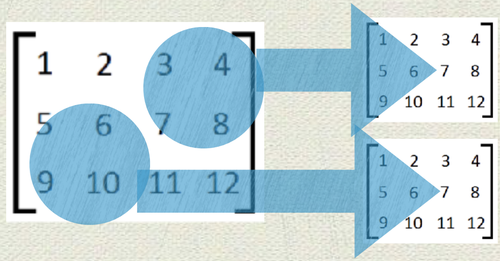

```
8   cv2.rectangle(okamuraKao,(left,top),(right,bottom),(255,0,0),2)
9   okamuraKaoBGR=cv2.cvtColor(okamuraKao,cv2.COLOR_RGB2BGR)
10  cv2.imshow('my Window',okamuraKaoBGR)
11  cv2.waitKey(5000)
```

RGB➡BGR

BGR変数

顔認証 (9/15)

- これで、画像内の顔を見つける方法がわかったので、コマンドを line 6から line 10まで消してください。




```
6  print(faceLoc)
7  top,right,bottom,left=faceLoc
8  cv2.rectangle(okamuraKao,(left,top),(right,bottom),(255,0,0),2)
9  okamuraKaoBGR=cv2.cvtColor(okamuraKao,cv2.COLOR_RGB2BGR)
```

次のステップでは、確立された顔をエンコードする必要があります。 このコマンドをline 5: `faceLoc=FR.face_locations(okamuraKao)[0]`下に追加してください。

- `okamuraKaoEncode=FR.face_encodings(okamuraKao)[0]`

- 別の顔(重長さんの顔）を読み込んでエンコードするので、line 4からline 6までのコマンドをコピーして6行目の後に入れてください。その後 line 7からline 9までokamuraをshigenagaに編集してください。

Shigenagaに編集

```
4   okamuraKao=FR.load_image_file('/Users/s118301/Documents/python/demoimages/known/okamura_hisanori.jpg')
5   faceLoc=FR.face_locations(okamuraKao)[0]
6   print(faceLoc)
7   okamuraKaoEncode=FR.face_encodings(okamuraKao)[0]
8   shigenagaKao=FR.load_image_file('/Users/s118301/Documents/python/demoimages/known/shigenaga_tomoko.jpg')
9   faceLoc=FR.face_locations(shigenagaKao)[0]
10  shigenagaKaoEncode=FR.face_encodings(shigenagaKao)[0]
```

顔認証 (10/15)

- これで、2つのエンコードされた顔が得られました。 それを配列に保存してみます。 line 9:shigenagaKaoEncode=FR.face_encodings(shigenagaKao)[0]の下にこちらのコマンドを追加してください。
  - knownEncodings=[okamuraKaoEncode,shigenagaKaoEncode]
- また、それらの名前を別の配列に保存する必要があります。こちらのコマンドを先のコマンドの下に追加してください。
  - names=[‘Okamura Masanori’,'Shigenaga Tomoko']

```
10    knownEncodings=[okamuraKaoEncode,shigenagaKaoEncode]
11    names=['Okamura Hisanori','Shigenaga Tomoko']
```

顔認証 (11 / 15)

- 次に、未知の顔画像を使用して、これらのエンコードされた顔をテストする必要があります。 そのうちの 1 つをロードしましょう。line 11: `names=['Makino Katsuhiko','Ohmi Yuka']`の下にこちらのコマンドを追加してください。

- `unknownKao=FR.load_image_file('<u1.jpgファイルパス>')` — U6.jpg(未知顔画像）ファイルをロードする
- `unknownKaoBGR=cv2.cvtColor(unknownKao,cv2.COLOR_RGB2BGR)` — 未知画像RGBからBGRに変更する
- `faceLocations=FR.face_locations(unknownKao)` — 未知画像にある顔の場所をArrayに保存する
- `unknownEncodings=FR.face_encodings(unknownKao,faceLocations)` — 未知画像にある顔をエンコードしてArrayに保存する

```
12  unknownKao=FR.load_image_file('/Users/s118301/Documents/python/demoimages/unknown/u1.jpg')
13  unknownKaoBGR=cv2.cvtColor(unknownKao,cv2.COLOR_RGB2BGR)
14  faceLocations=FR.face_locations(unknownKao)
15  unknownEncodings=FR.face_encodings(unknownKao,faceLocations)
```

顔認証 (12/15)

knownEncodings=[$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$, $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$]

unknownEncodings=[$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$, $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$]

これで、エンコードされた顔の配列が 2 つできました。 知っている顔と知らない顔。 次のステップでは、未知のエンコードされた顔をチェックして、既知のエンコードされた顔と一致するかどうかを確認し、名前を付けます。

顔認証 (13/15)

- for faceLocation, unknownEncoding in zip(faceLocations, unknownEncodings):
  - top,right,bottom,left=faceLocation
  - cv2.rectangle(unknownKaoBGR,(left,top),(right,bottom),(255,0,0),2)
  - name='Unknown Person'
  - matches=FR.compare_faces(knownEncodings,unknownEncoding)
  - print(matches)

画像上の顔の位置とそのエンコーディングを１つずつ取得します。

顔の周りに長方形を作ります。

未登録の顔に「未知の人」というラベルを付けます。

ここで、エンコードされた顔が既知のエンコードされた顔と一致することを確認します。一致する場合は True を返します。

```
for faceLocation, unknownEncoding in zip(faceLocations, unknownEncodings):
    top,right,bottom,left=faceLocation
    cv2.rectangle(unknownKaoBGR,(left,top),(right,bottom),(255,0,0),2)
    name='Unknown Person'
    matches=FR.compare_faces(knownEncodings,unknownEncoding)
    print(matches)
```

実行してください →

```
(pyrasppi) (base) s118301@Neut
s/python/openCV-2.py
[True, False]
[False, True]
```

顔認証 (14/15)

- if True in matches:

エンコードされた既知の顔と一致するかどうかを確認しま

- matchIndex=matches.index(True)

一致する場合は、既知のエンコードされた顔が一致する番号を取得します。

- name=names[matchIndex]

次に、エンコードされた既知の顔から名前を取得して表示します。

- cv2.putText(unknownKaoBGR,name,(left,top),font,.5,(0,0,255),2)

```
for faceLocation, unknownEncoding in zip(faceLocations, unknownEncodings):
    top,right,bottom,left=faceLocation
    cv2.rectangle(unknownKaoBGR,(left,top),(right,bottom),(255,0,0),2)
    name='Unknown Person'
    matches=FR.compare_faces(knownEncodings,unknownEncoding)
    print(matches)
    if True in matches:
        matchIndex=matches.index(True)
        name=names[matchIndex]
    cv2.putText(unknownKaoBGR,name,(left,top),font,.5,(0,0,255),2)
cv2.imshow('my Window',unknownKaoBGR)
```

- cv2.imshow('my Window',unknownKaoBGR)

顔認証 (15/15)

その後、プログラムを実行してください。
他のunknownの画像を確認して結果を確認することもできます。 一部の人物は、その人物の顔をエンコードしていないため、不明に見える場合があります。

これで、ファイルから画像を取得し、顔があるかどうかを確認して認識する方法がわかりました。 前回は、Web カメラから画像を取得する方法を説明しました。これは宿題です。カメラを使用して画像を取得し、カメラで捉えられた顔を認識してpythonコードを作ってください。

まずは自分でやってみて、出来なかったらhomeworkというフォルダに答えを書きました。 それを見て、なぜ完了できないのかを見つけてください。