雰囲気でPythonを使っている人のための Python再入門

2021/09/04

GTUGGirls ハンズオン

めも：

- `>>>` 部分は不要なところは削除
- 一緒に写経してもらいたいところを考える
  - モジュールのところは一緒にするかどうか？
- 会の後の質問場所 > メンションか、DM ください
- windows store でインストールしたひとのシェル、ターミナルという時はそれを立ち上げる

自己紹介

- しんせいたろう
- 雰囲気でpython書いてる(10年以上)
- しごと
  - 米国株とれーだー
  - 時系列データ分析 (株、暗号資産、為替) 、投資ストラテジー作成
  - python の個別指導
- コミュニティ
  - GTUGGirls (スタッフ)
  - Fin-py (准管理者)
  - 月刊Fintalk (主宰)
  - モグモグDjango (主宰)

今日の予定

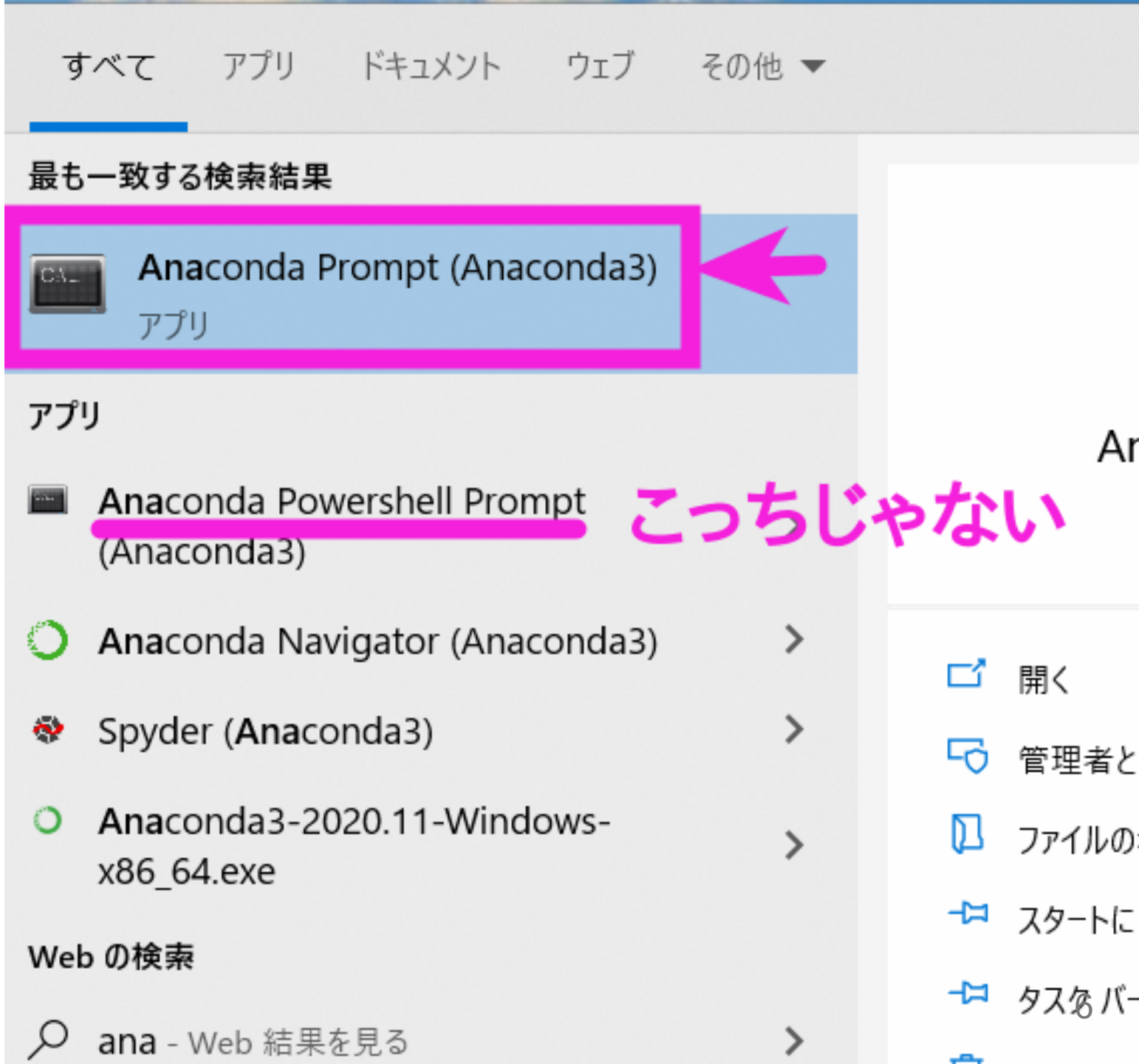
1. 仮想環境作成
2. Python完全に理解した、からちょっと脱却して、チョットデキルひとになる最初の一歩

今日伝えたいことはたったひとつ

とりあえずPythonの仮想環境作ってください。
心からのお願いです。

Windows Anaconda

- Anaconda Prompt を使う
- Anaconda PowerShell Prompt ではない



by @しんせいたろう

Windows Store

- Windows Store で python をインストールした場合

Python を確認

- terminal / command prompt 立ち上げ
- windows:

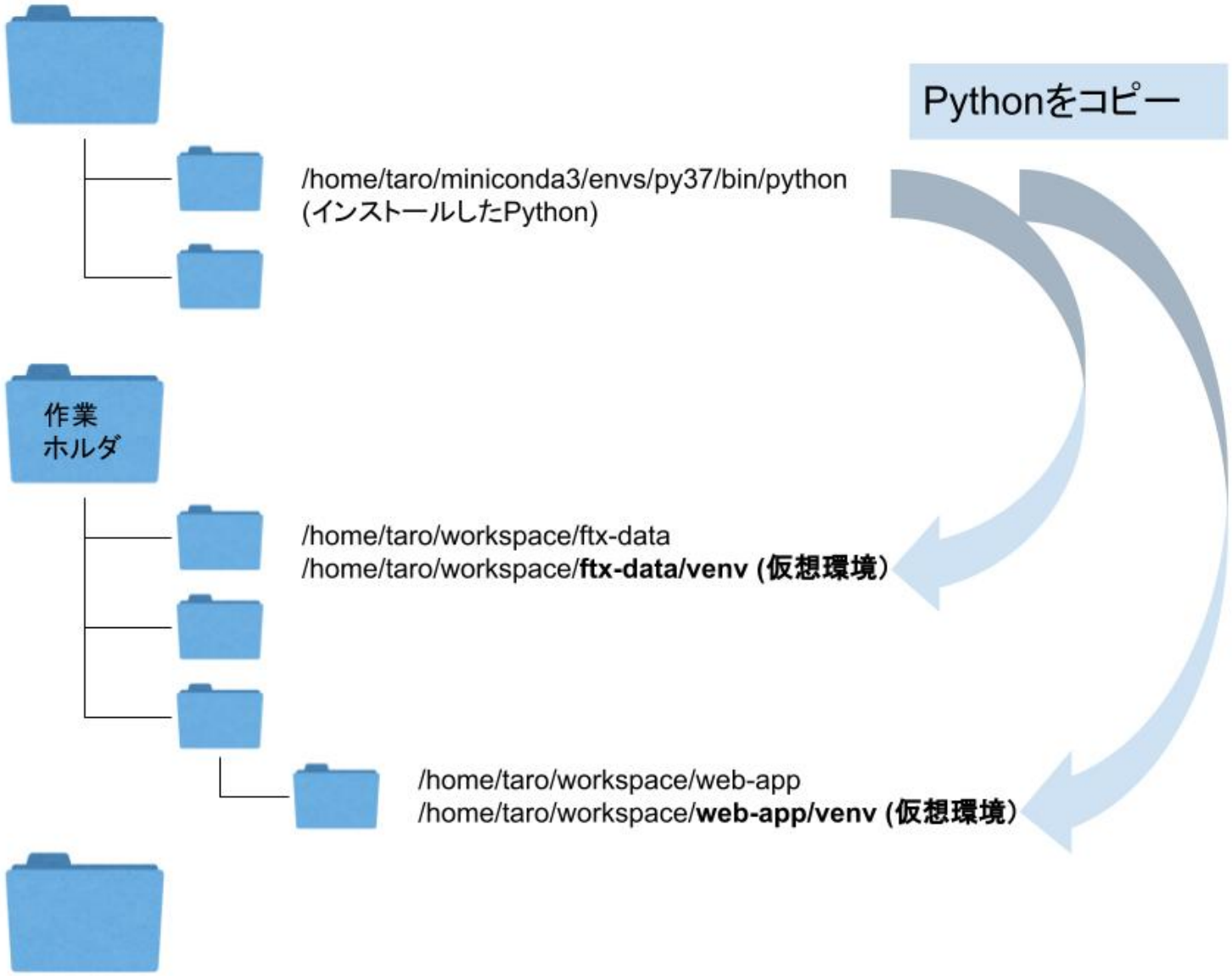
```
where python
```

  - 一番上にでてきた python へのパスが有効
- mac / linux

```
which python
```

  - 有効な python パスのみ出てくる

仮想環境構築


Pythonをコピー
/home/taro/miniconda3/envs/py37/bin/python
(インストールしたPython)
作業
ホルダ
/home/taro/workspace/ftx-data
/home/taro/workspace/ftx-data/venv (仮想環境)
/home/taro/workspace/web-app
/home/taro/workspace/web-app/venv (仮想環境)

仮想環境とは

- ただのホルダ
- 仮想環境を作成とは
  - インストールしたPythonだけを作業ディレクトリ（プロジェクトディレクトリ）にコピーすること
- 仮想環境に入るとは
  - つくったコピー環境を使うこと
- なぜ仮想環境を作る？
  - 作成、削除、改変、なんでも出来る
  - 何かしらの理由で実行できなくなったとしても仮想環境ディレクトリを消して再作成すればいい
  - サードパーティライブラリのコンフリクトを防ぐ
  - 「自分のPythonではinstall出来ません」という情弱行動を防げる 👈 😆

仮想環境構築手順

- 例： `myproject` という作業ディレクトリ配下で仮想環境を作る
  - お作法
    - デスクトップに作らない
    - root ディレクトリ直下には作らない
    - 半角英数字のみで作成(日本語は使わんで)
- 作業ディレクトリ = レポジトリの感覚
- レポジトリがわからないひとにはただのホルダでオッケ

仮想環境構築手順 Mac / Linux

1. 作業ディレクトリ作成

```
mkdir myproject
cd myproject
```

2. 現在のPythonを確認する（任意）

```
# linux / mac
which python
```

3. 仮想環境構築

```
# python -m venv <仮想環境ディレクトリの名前>
python -m venv .venv
```

   - ディレクトリ名でよく使われるもの： `.venv` `venv` `myenv` `myvenv`

4. 仮想環境に入って、確認

```
source .venv/bin/activate
which python
```

仮想環境構築手順 Windows

あとで書く

実行

1. インタラクティブシェル

```
python
```

2. ファイル実行

```
python src/hoge.py
```

以後、Python を実行するターミナルでは必ず仮想環境にはいる

- (linux / mac)

```
source .venv/bin/activate
which python
```

- (win)

```
venv\Script\activate
where python
```

ライブラリインストール

pip とは

- Pythonで書かれたソフトウェアをインストール、管理するためのパッケージ管理システム

文法

```
pip install ライブラリ名
```

【コラム】むやみに、pip install して不幸になるひとの行動パターン

> 1. 仮想環境作らずに、いきなり `pip install ライブラリ名` するひと
> 2. jupyter notebook 上で `!pip install ライブラリ名` するひと
>
> - pip は ライブラリ同士の依存関係を管理しない。よってコンフリクトを起こして「自分のPythonでは install出来ません」という情弱行動になる（二回目）👈😆

ライブラリインストール

1. まずは `which python` / `where python` して仮想環境に入っているか確認
2. `pip install ライブラリ名` するのは、pip じたいをUpdateするときだけ

```
pip install -U pip
```

3. 上記以外のインストールは、作業ディレクトリ直下に `requirements.txt` にインストールしたいライブラリ名を書いてインストールする
    i. `requirements.txt` 新規作成(慣習的にこの名前だが違ってもOK)

```
touch requirements.txt
```

    ii. `requirements.txt` にインストールしたいライブラリ名を書く

```
# 一行に一つ
beautifulsoup4
# バージョン指定も可
requests-html == 0.10.0
```

    iii. `-r` オプションでインストール

```
pip install -r requirements.txt
```

ライブラリインストール 確認

```
pip freeze
```

.venv を覗く 👀

.venv を削除してもう一回インストール

よくある質問

- どこそこに仮想環境を作ることになって、容量食いませんか？
  - 消せばいいよ(.venv だけ)
  - 気になるひとは → pipxのススメ - podhmo's diary https://pod.hatenablog.com/entry/2021/06/23/221537

演習

1. ディレクトリ `gtugtest` 作成
2. その中に python の仮想環境を作成
3. 以下のライブラリを仮想環境下にインストール
   > djangorestframework
   > Django
4. `gtugtest` 削除

「Python完全に理解した」からちょっと脱却して「チョットデキル」ひとになる最初の一歩

Python は jupyter notebook ではない

- jupyter を python と思ってるひとが多数
- jupyter でしか python を実行出来ないひと多数
- ヤメて

エラーは読んで

```
>>> 10/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

- エラーメッセージは下から上に読んでいく
- 上記の場合 `ZeroDivisionError: division by zero` がエラーの根幹
- 質問する時は、エラーメッセージを全部コピペして、問題ない範囲でコードを渡して再現してもらおう

コピペはしないで

- 記者に「プログラミングのスキル」って必要なの？ちなみにNHKニュースの画像生成も記者がコードを書いてます｜NHK取材ノート｜note
  > こうした手を動かした学習は確実に取材に役立ち、大きなシステムトラブルが発生した時に、**原因が推測**できたりするようになりました。

  - ほんとコレ
  - ほんとコレ
  - ほんとコレ

python の概念,キーワード,知らないとツライ単語

- オブジェクト
- メソッド
- 関数
- クラス
- インスタンス
- スコープ
- モジュール

Pythonista がこのむ python らしい書き方

- 内包表記
- アンパック代入
- f-リテラル
- 高階関数 (lambda, map, filter)

test

python 概念と pythonist らしいコードの例

構造

- 私が「キレイ」だと思ったpythonのfile構造
- モジュールの説明も兼ねる
- main.py を実行ファイルとして、あとのファイルには関数群を作る

python はオブジェクト

- beautifulsoup でスクレイピングしたデータを
- pandas に流し込んで
- 可視化

VSCode の設定

```
{
    // PythonのPATHをワークスペースの仮想環境にする
    "python.pythonPath": "${workspaceFolder}/venv/bin/python",
    // 仮想環境にインストールしたファイルは監視対象から除外する
    "files.watcherExclude": {
        "**/venv/**": true
    },
    // リンタでPyLintは使わない
    "python.linting.pylintEnabled": false,
    // リンタでFlake8を使う
    "python.linting.flake8Enabled": true,
    // コードフォーマッタでBlackを使う
    "python.formatting.provider": "black",
    // Blackは貼り付け時の整形に対応していないので無効にする
    "editor.formatOnPaste": false,
    // 1行の文字数を88文字とする
    "python.linting.flake8Args": ["--max-line-length", "88"],
    // languageServerにPylanceを使う
    "python.languageServer": "Pylance",
    // Pylanceの型チェックをbasicにする
    "python.analysis.typeCheckingMode": "basic",
    // Pylanceの括弧補完を有効にする
    "python.analysis.completeFunctionParens": true,
}
```

VSCode の設定

```
{
    // PythonのPATHをワークスペースの仮想環境にする
    "python.pythonPath": "${workspaceFolder}\\venv\\Scripts\\python.exe",
    // 仮想環境にインストールしたファイルは監視対象から除外する
    "files.watcherExclude": {
        "**/venv/**": true
    },
    // リンタでPyLintは使わない
    "python.linting.pylintEnabled": false,
    // リンタでFlake8を使う
    "python.linting.flake8Enabled": true,
    // コードフォーマッタでBlackを使う
    "python.formatting.provider": "black",
    // Blackは貼り付け時の整形に対応していないので無効にする
    "editor.formatOnPaste": false,
    // 1行の文字数を88文字とする
    "python.linting.flake8Args": ["--max-line-length", "88"],
    // languageServerにPylanceを使う
    "python.languageServer": "Pylance",
    // Pylanceの型チェックをbasicにする
    "python.analysis.typeCheckingMode": "basic",
    // Pylanceの括弧補完を有効にする
    "python.analysis.completeFunctionParens": true
}
```

tips:

- github の private repo 機能便利