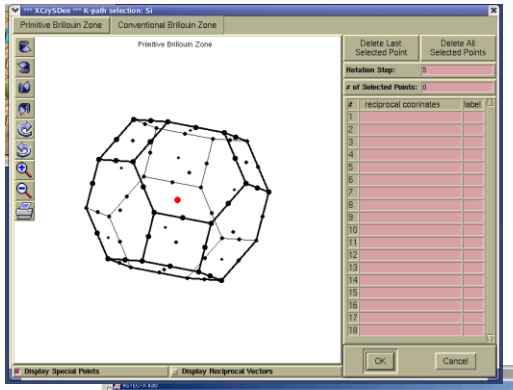
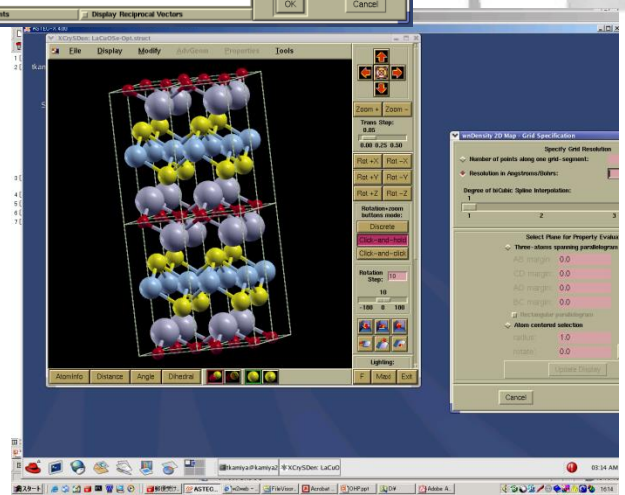
講義web: http://d2mate.mdxes.iir.isct.ac.jp/Lecture/FundamentalsCMS/

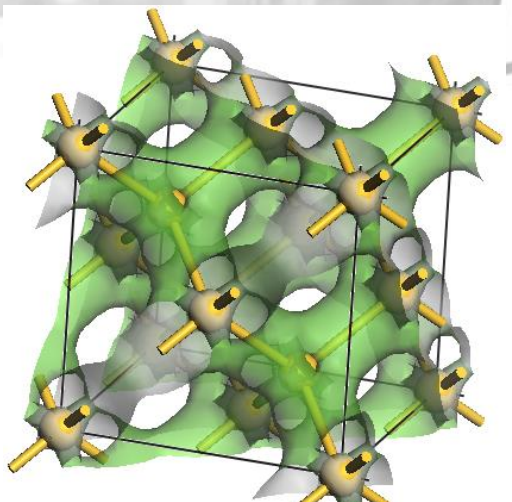
材料計算科学基礎

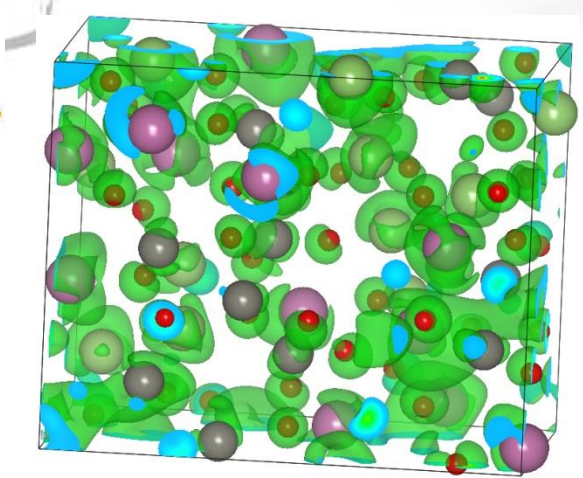
物質理工学院材料系
総合研究院 元素戦略MDX研究センター
神谷利夫

TA: 樋口龍生
南島悠人









授業日程

#01 2024/12/06 (金)～#08 2025/1/7 (火)　山口先生担当

分子軌道計算の基礎･演習
機械学習の基礎･演習
・回帰 (教師あり)　・分類 (教師なし)　・強化学習　・次元 (変数) 削減

#09　2025/1/10 (金)　神谷担当

#10　2025/1/14(火)

#11　2025/1/21(火)

- プログラミングの基礎 (python)
- 例題: 回帰
  - 線形最小二乗法
  - 非線形最小二乗法
  - 生成AI (ChatGPT、Microsoft 365 copilot, github copilot) を使ったプログラミング

#12　1/24～#14　1/31　講義

評価

講義毎の課題で評価

提出方法: T2SCHOLARで提出
提出期限: 1/12(日) 23:59

課題: 次のいずれかを選択して、成果物等を提出
- 成果物の例:
  - 授業で作成したプログラム (.py、.ipynb)
  - 授業に関連して作成したプログラム
  - 授業に関連して生成AIに質問し、その結果理解したこと
    質問のプロンプトと回答、回答に関しての自分の感想・理解など
- 講義に関する質問、要望

講義の目的

研究の解析・整理に必要な python プログラムを理解・作成できるようになる

実際に使えるプログラム (最小二乗法) を完成させる

講義Web

http://d2mate.mdxes.iir.isct.ac.jp/Lecture/FundamentalsCMS

D²MatE Top

original HTML
display HTML

2024年度Q4 材料計算科学基礎

python
プログラム例と文法

講義予定
#01 ～ #08: ・機械学習の基礎・演習
#09: ・Visual Studio Codeによる環境構築
・pythonプログラム実習: polyfit()を用いた多項式最小二乗プログラムの作成
#10:
・より複雑な最小二乗法
線形最小二乗法
非線形最小二乗法
#11:
・生成AIを利用したプログラム作成
#12～#14:講義

original HTML | display HTML

2024年度Q4 材料計算科学基礎

Fundamentals of Computational Materials Science 2024 Q4

News:

- 2025/1/7　14:42　1/10講義資料アップロード
- 2025/1/6　12:59　本ページオープン

講義資料

#09 2025/1/10 (金) 神谷担当

pythonプログラミング
講義資料: 20250108slides.pdf

- ▶ 01/data.csv (download/show)
- ▶ 01/test.ipynb (download/show)
- ▶ 01/test.py (download/show)
- ▶ 01/read.ipynb (download/show)
- ▶ 01/data2.csv (download/show)
- ▶ 01/lsq.ipynb (download/show)

最新講義資料・プログラム等

講義Web

http://d2mate.mdxes.iir.isct.ac.jp/Lecture/FundamentalsCMS

D²MatE Top

original HTML
display HTML

2024年度Q4 材料計算科学基礎

python

プログラム例と文法

講義予定

#01 ～ #08: ・機械学習の基礎・演習

#09: ・Visual Studio Codeによる環境構築
・pythonプログラム実習: polyfit()を用いた多項式最小二乗プログラムの作成

#10:
・より複雑な最小二乗法
線形最小二乗法
非線形最小二乗法

#11:
・生成AIを利用したプログラム作成

#12～#14:講義

original HTML display HTML

簡単なプログラム例

- ▶ キーボード入力とコンソール出力 (download/show)
- ▶ list型とforループ (基本型) (download/show)
- ▶ list型とforループ (list変数をiteratorとして使う) (download/show)
- ▶ list型とforループ (enumerate) (download/show)
- ▶ リスト内包表記を使ったforループ (download/show)
- ▶ if ~ elif ~ else (download/show)
- ▶ CSVファイルの読み込み (基本形) (download/show)
- ▶ CSVファイルの読み込み (with構文) (download/show)
- ▼ CSVファイルの読み込み (CSVライブラリ) (download/show)

Clipboardへコピー

copy

Exampleプログラムの内容

```
import csv

infile = "data.csv"

with open(infile, "r") as fp:       # input.csvファイルを開いてファイルハンドルを取得
    csv_reader = csv.reader(fp)         # CSVファイルを読み込むため、csv.reader() でreaderオブジェクトを取得
    labels = next(csv_reader)           # labelを読み込む
    data_list = []

    for row in csv_reader:                  # 1行ずつ読み込み、list型配列をrowに入れる
        row = [float(d) for d in row]   # rowの要素（文字列型）を実数型（float）に変換してリストを作る
        data_list.append(row)

print("labels:", labels)
print("data_list:", data_list)
```

関連Web: 最小二乗法の理論

http://conf.mdxes.iir.isct.ac.jp/D2MatE/D2MatE_programs.html?page=cms

top python webプログラミング 小物プログラム optimize flex SILVACO ATLAS Tutorial2022 数値解析 結晶工学スクール

2024年度Q2 計算材料学特論 (資料: 英語＋日本語版)

Computational Materials Science 2024 Q2

数値解析に関する講義資料・pythonプログラム (神谷担当分)

Lecture materials for numerical analyses (by Kamiya)

講義で使うプレゼン資料は共通して使うpythonプログラム」の下にあります

Lecture presentation slides are found after the "Common python programs" section below.

Update News:

- July 05, 8:24 Lecture materials on July 05 have been uploaded (20240705FFT-Matrixpdf.zip)
- June 28, 10:42 Lecture materials on June 28 have been updated (20240628FT_Matrix3.zip)
- June 27, 9:25 Lecture materials on June 25 have been updated (20240627LSQEquationOptimize.zip)
- June 23, 9:32 Lecture materials on June 21 have been updated (20240621InterporlateSmoothing.zip)
- June 23, 9:32 Lecture materials on June 21 have been updated (20240621InterporlateSmoothing.zip)
- June 14, 13:07 Lecture materials on June 14 have been updated (20240614DifferentialIntegration2.zip)
- June 12, 14:13 Lecture materials on June 11 have been updated (20240612ComputerAndErorrSouorces.zip)

▶ 詳細履歴

python notes and tips

Other related programs

- D2MatE拠点開発プログラム (Japanese)
  機械学習、数値解析のpythonプログラムをGUIインターフェースで配布。

機械学習関連プログラム

http://conf.mdxes.iir.isct.ac.jp/D2MatE/D2MatE_programs.html?page=top

top | python | webプログラミング | 小物プログラム | optimize flex | SILVACO ATLAS | Tutorial2022 | 数値解析 | 結晶工学スクール

original HTML
display HTML

- Top page
- 共通ファイル・単位など
  - 神谷・片瀬研共通単位表
- 小物プログラム集
- 汎用最適化フレームワーク (optimize_flex)
- Webプログラミング
- Electronによるデスクトップアプリ開発
- pythonでGPIB制御メモ
- Excel=>python移植支援
- SILVACO ATLAS tips
- ▶ python Tips
- ▶ python 高度な内容
- ▶ プログラミングメモ
- ▶ Linuxサーバ設定
- ▶ 神谷・片瀬研メモ
- ▶ 講義資料
- D2MatE拠点限定ページ
- 更新履歴

original HTML | display HTML

智慧とデータが拓くエレクトロニクス新材料開発拠点　公開・非公開プログラム情報
(Data Driven Materials Research Institute for Electronics)

質問、要望、バグ報告などの連絡先: 神谷 利夫 tkamiya@msl.titech.ac.jp
東京科学大学 総合研究院　先駆研究センター 元素戦略MDX研究センター 教授

News (ユーザ)

- **New!** 2025/1/1 plotly-Dash-Flaskプログラム ページに、Flask/Dash/Plotlyの参考プログラムを掲載しました。FlaskとDash
  と選択基準を近いうちにまとめます
- **New!** 2024/12/29 WSGIアプリのデプロイ ページに、AlmaLinux9 + httpd + python3.11 + mod_wsgi のインストール方
  しました。
- **New!** 2024/12/28 本Webサイトを http://conf.msl.titech.ac.jp/D2MatE/D2MatE_programs.html から
  http://conf.mdxes.iir.isct.ac.jp/D2MatE/D2MatE_programs.html に移行しました。
- Webプログラミング で開発しているJavascriptライブラリを使い、Topページを含む一部ページをupdateしています。
- 2024/12/08 Webプログラミング で開発しているJavascriptライブラリを使い、Topページを含む一部ページをupdateしていま
- 2024/11/28 Electronによるデスクトップアプリ開発 を作り始めました。
- 2024/11/22 Web/HTML/Javascript/CSS/framework関係を Webプログラミング ページにまとめました。
- 2024/11/19 汎用最適化フレームワーク optimize_flex のページを作り始めました。
  関連プログラムを含めたパッケージは近いうちに公開します。

機械学習回帰のチュートリアルコース 2022年

http://conf.mdxes.iir.isct.ac.jp/D2MatE/2022Tutorial/tutorial2022.html

材料計算科学・データ解析チュートリアルコース 2022年度

2022年度チュートリアルコースは終了いたしました。

チュートリアル資料の更新は、下記の4/2 14:04更新と録画公開で終了しました。
配布プログラムの更新は、Topページにて行います。

- **2023/4/2 14:04 第5回チュートリアル (3月22日) 講義資料 最終更新版: 20230322Tutorial.zip**
  2023/4/1 注: Arrhenius plot機能について、活性化エネルギーの計算に間違いがあったものを修正しました。
- **2023/3/22 第1～5回チュートリアル録画 2023年3月いっぱいの予定で公開: 2022年度チュートリアル録画**

チュートリアル資料 再構成版: チュートリアルの資料をテーマごとにまとめなおしました。
ファイル名が日本語になっています。ダウンロードができない場合、tkamiya@msl.titech.ac.jp までご連絡ください。

- 01-注意.pdf
- 02-Launcher.pdf
- 02-python-tips.pdf
- 03-強化学習プログラムbayes_gp_gui.pyの使い方.pdf
- 04-線形最適化・最小二乗法(回帰).pdf
- 05-機械学習(回帰).pdf
- 06-非線形最適化・最小二乗法.pdf

チュートリアル: 学生と教員のためのpythonとChatGPT活用法

http://conf.mdxes.iir.isct.ac.jp/D2MatE/2023Tutorial/tutorial2023-python-ChatGPT.html

2023年度 チュートリアル

主催: D^{2}MatE

共催:
元素戦略MDX研究センター　講演会
第172回フロンティア材料研究所学術講演会
出島コンソーシアム・チュートリアル講座
物質・情報卓越教育院講演会

チュートリアル題目: チュートリアル： 学生と教員のためのpythonとChatGPT活用法

開催日時: 2024年1月24日(水) 15:00～16:30 (この時間後も個別の質問を受け付けます)

開催方法: Zoom Webinar

参加対象: 制限はありません

参加費: 無料

事前登録 (修了しました) :

プログラミング・コンピュータ環境:
講義中には時間がありませんので、実習は行いません。
興味のある方は、python (numpy, scipy, openpyxl, pandas), Visual Studio Code (python plug-in推奨),
JupyterおよびChatGPTが使える環境があれば、講義中にも確認できると思います。

講義資料: python-tutorial2023-V5.zip (2024/1/24 17:12)

なぜpythonか

長所:

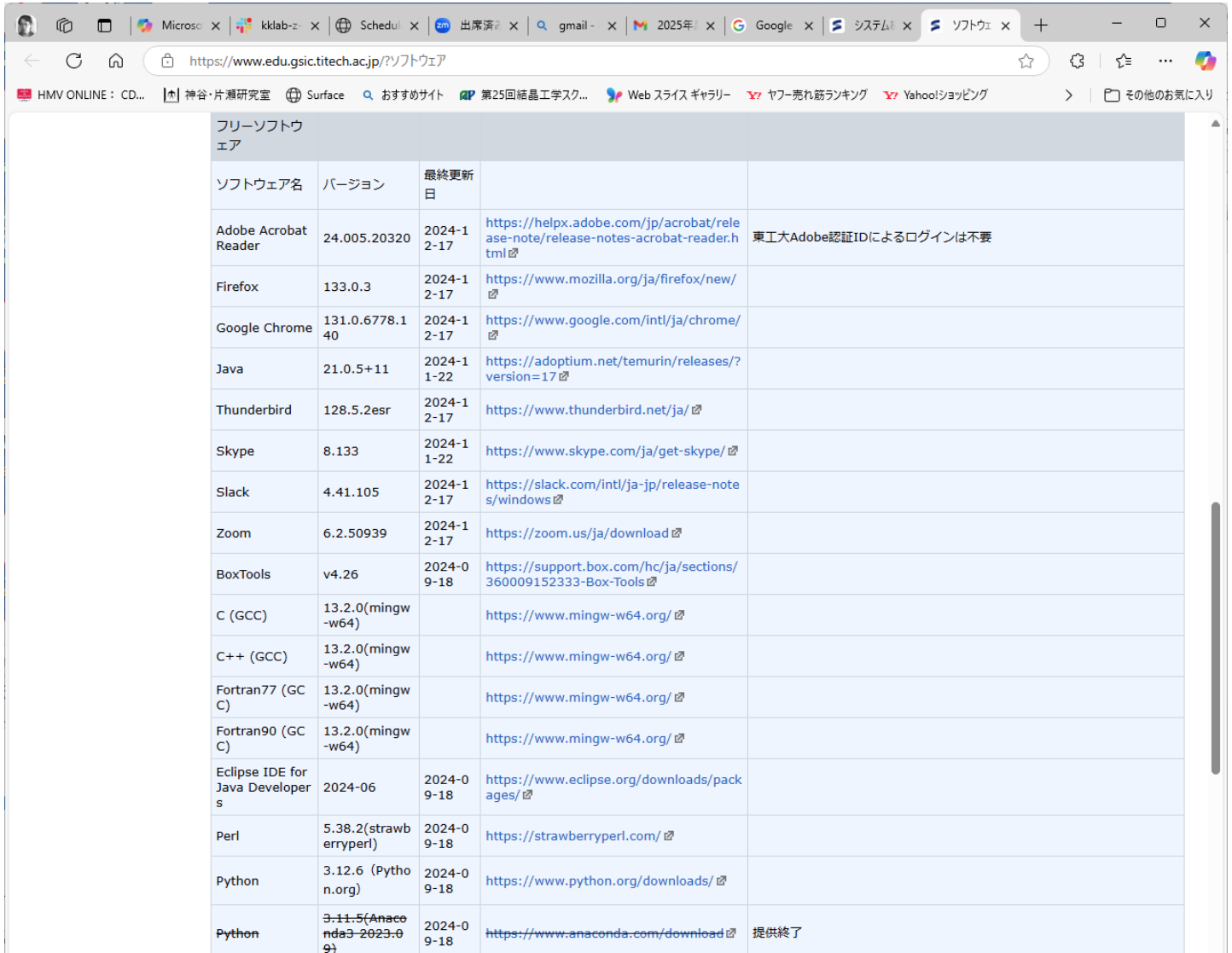
- 基本的に覚えることが少なく、学習コストが低い
- 科学計算、機械学習、可視化(グラフ化)ライブラリが充実
- 自分で作ったプログラムを、そのままライブラリとして再利用できる
- (プログラム作成 => 実行 => 修正) サイクルを回して
  使える状態まで完成させる時間が短い
- 読みやすい (誰が書いても似たプログラムになる)
- 多言語 (C, C++, php, javascript etc) へ展開しやすい (C言語系の文法)

短所:

- 実行速度が遅い
- python独特の文法がある (インデントブロック、for文 など)

教育用電子計算機システム2024の環境

https://www.edu.gsic.titech.ac.jp/?ソフトウェア
https://www.edu.gsic.titech.ac.jp/?%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2



フリーソフトウェア

ソフトウェア名	バージョン	最終更新日		
Adobe Acrobat Reader	24.005.20320	2024-12-17	https://helpx.adobe.com/jp/acrobat/release-note/release-notes-acrobat-reader.html	東工大Adobe認証IDによるログインは不要
Firefox	133.0.3	2024-12-17	https://www.mozilla.org/ja/firefox/new/	
Google Chrome	131.0.6778.140	2024-12-17	https://www.google.com/intl/ja/chrome/	
Java	21.0.5+11	2024-11-22	https://adoptium.net/temurin/releases/?version=17	
Thunderbird	128.5.2esr	2024-12-17	https://www.thunderbird.net/ja/	
Skype	8.133	2024-11-22	https://www.skype.com/ja/get-skype/	
Slack	4.41.105	2024-12-17	https://slack.com/intl/ja-jp/release-notes/windows	
Zoom	6.2.50939	2024-12-17	https://zoom.us/ja/download	
BoxTools	v4.26	2024-09-18	https://support.box.com/hc/ja/sections/360009152333-Box-Tools	
C (GCC)	13.2.0(mingw-w64)		https://www.mingw-w64.org/	
C++ (GCC)	13.2.0(mingw-w64)		https://www.mingw-w64.org/	
Fortran77 (GCC)	13.2.0(mingw-w64)		https://www.mingw-w64.org/	
Fortran90 (GCC)	13.2.0(mingw-w64)		https://www.mingw-w64.org/	
Eclipse IDE for Java Developers	2024-06	2024-09-18	https://www.eclipse.org/downloads/packages/	
Perl	5.38.2(strawberryperl)	2024-09-18	https://strawberryperl.com/	
Python	3.12.6（Python.org）	2024-09-18	https://www.python.org/downloads/	
~~Python~~	~~3.11.5(Anaconda3-2023.09)~~	2024-09-18	~~https://www.anaconda.com/download~~	提供終了

使用する環境

Python: version 3.12.6　https://www.python.org/
　モジュール: numpy, scipy, openpyxl, pandas, jupyter などが使える
　実行方法: Windowsのタスクバー “検索”ボックスに “cmd” と入力して実行
　実行方法: Windowsのタスクバー “検索”ボックスに “code” と入力して実行
　　=> “python” を実行

エディタ: **Visual Studio Code 1.92.2 + python, CSV拡張機能**
　実行方法: Windowsのタスクバー “検索”ボックスに “code” と入力して実行

Microsoft 365 copilot
　実行方法: Windowsのタスクバーの “検索”ボックスの右のアイコンをクリックし、
　　検索ウインドウ右上のアイコンをクリック

ChatGPT
　Sign up: https://platform.openai.com/signup

ファイルの共有

講義Web: http://conf.mdxes.iir.isct.ac.jp/Lecture/FundamentalsCMS/

original HTML | display HTML

2024年度Q4 材料計算科学基礎

Fundamentals of Computational Materials Science 2024 Q4

News:

- 2025/1/6　12:59　本ページオープン

講義資料

#09 2025/1/10 (金) 神谷担当

pythonプログラミング
講義資料: 20250106slides.pdf

- ▶ 01/data.csv (download/show)
- ▶ 01/test.ipynb (download/show)
- ▶ 01/test.py (download/show)
- ▶ 01/read.ipynb (download/show)
- ▶ 01/data2.csv (download/show)
- ▶ 01/lsq.ipynb (download/show)

教育用電子計算機システム

Y:¥2024¥kamiya-t-aa　　学生は読み込みのみ
Y:¥2024¥kamiya-t-aa¥rwx　　学生も書き込み可

環境構築

Visual Code Studioに拡張機能をインストール

・日本語拡張パック

・python拡張機能

(・CSV拡張機能)

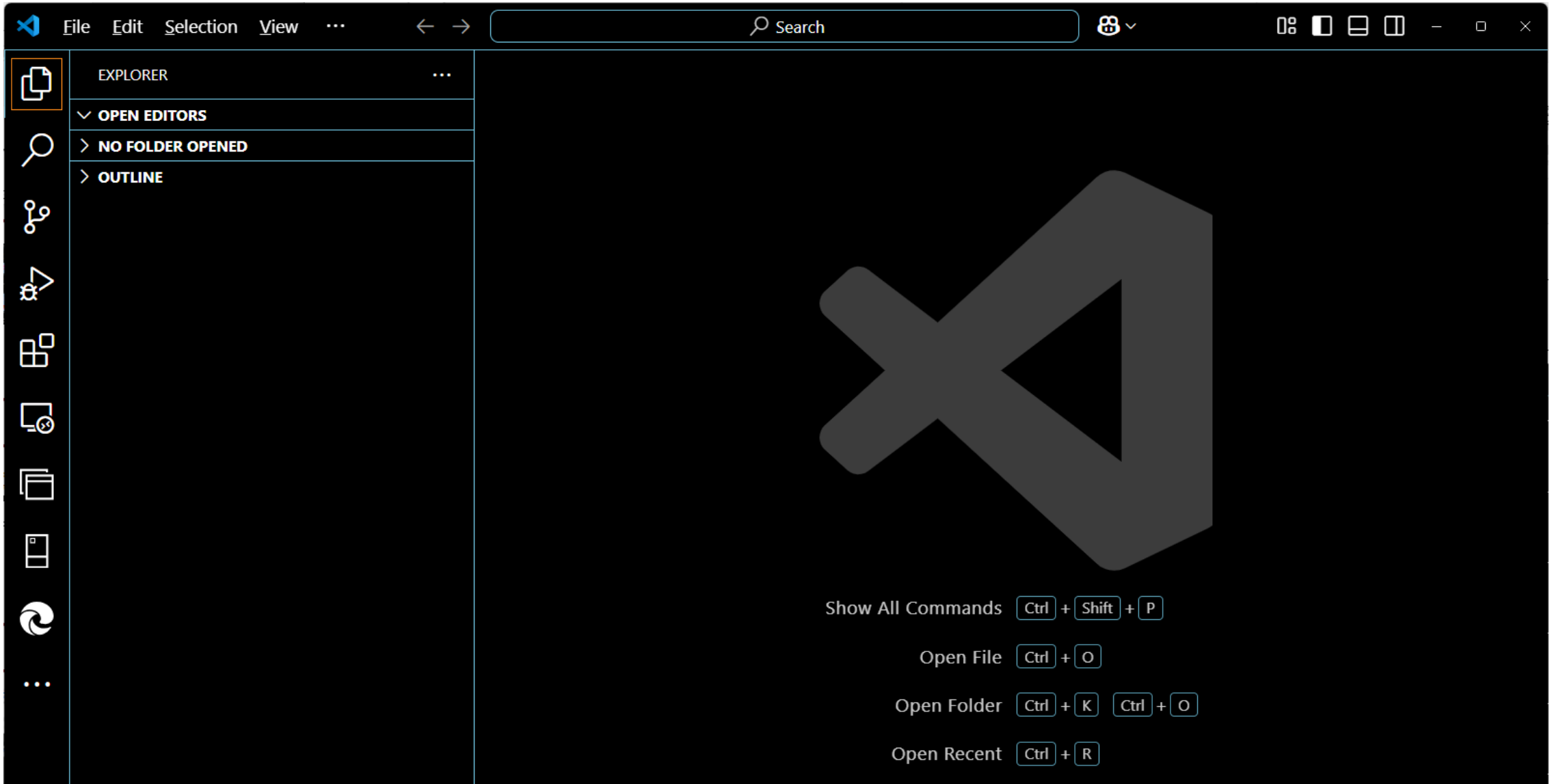
テキストエディタ: Visual Studio Code (VSCode)

https://code.visualstudio.com/

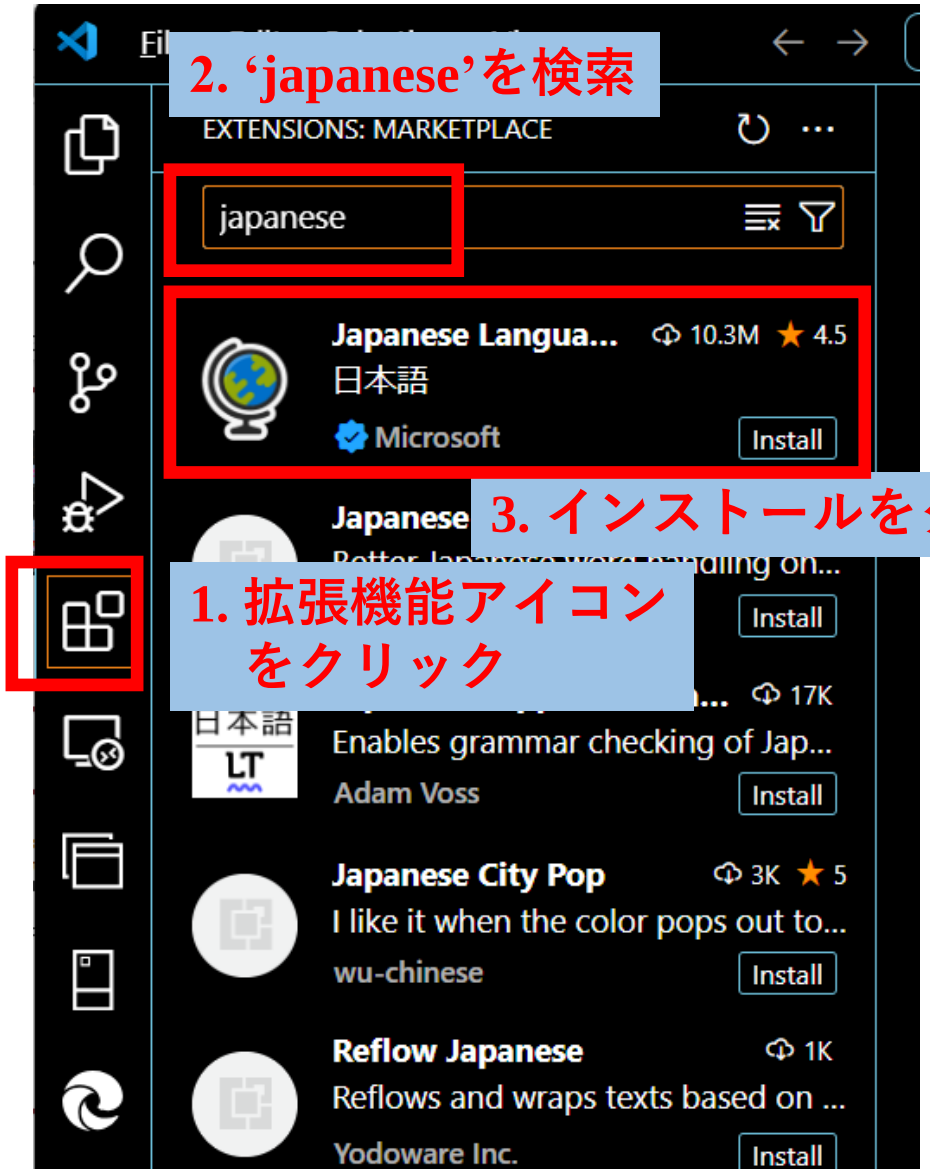
- 現在最も広く使われている
- 無料
- 多言語対応、マルチプラットフォーム (Win, macOS, Linux)
- 拡張機能 (python, CSV, ChatGPT, github copilot、その他たくさん)

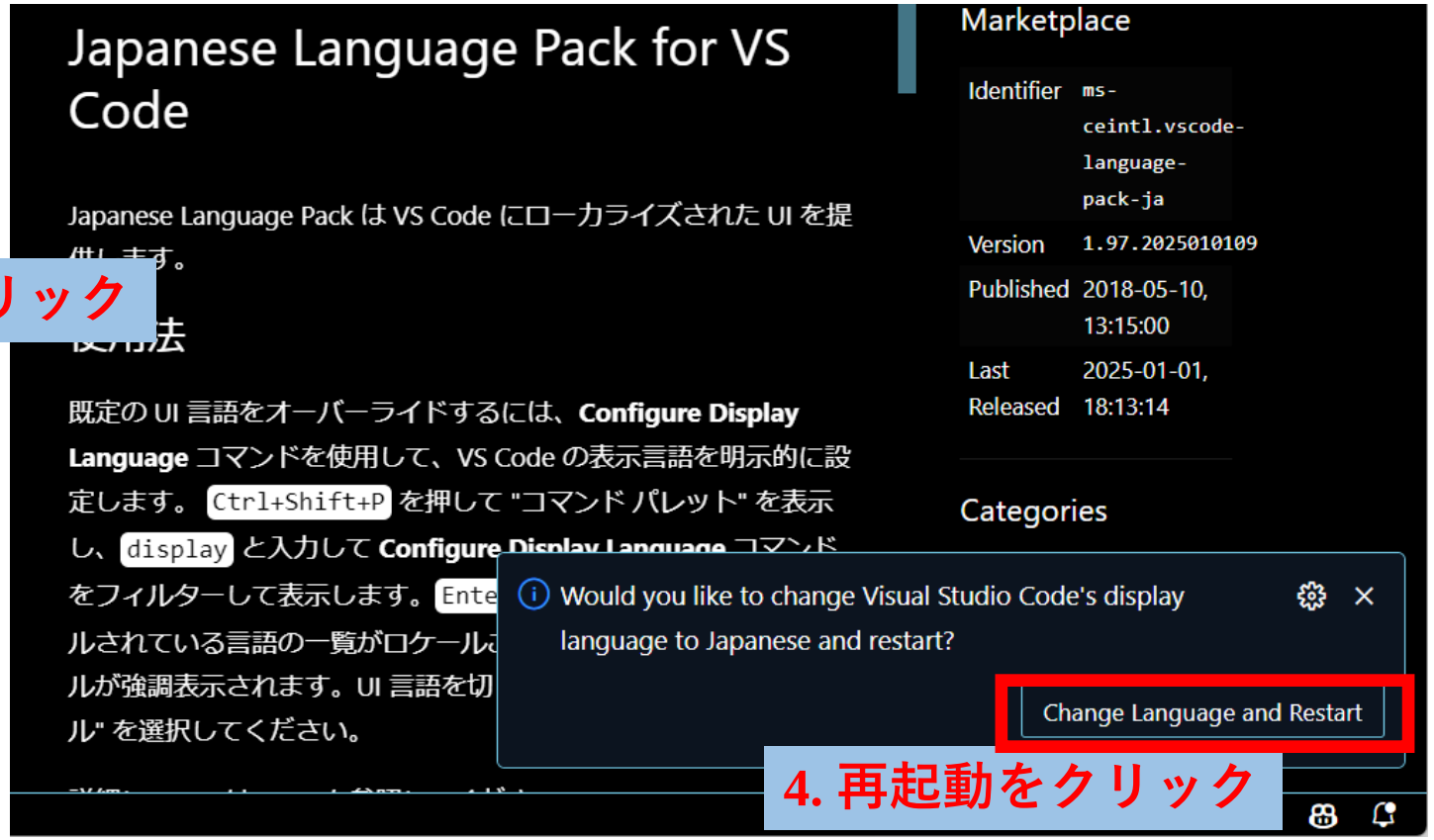
Visual Code Studioの起動

HP: https://code.visualstudio.com/
実行方法: Windowsのタスクバー “検索”ボックスに “code” と入力して実行

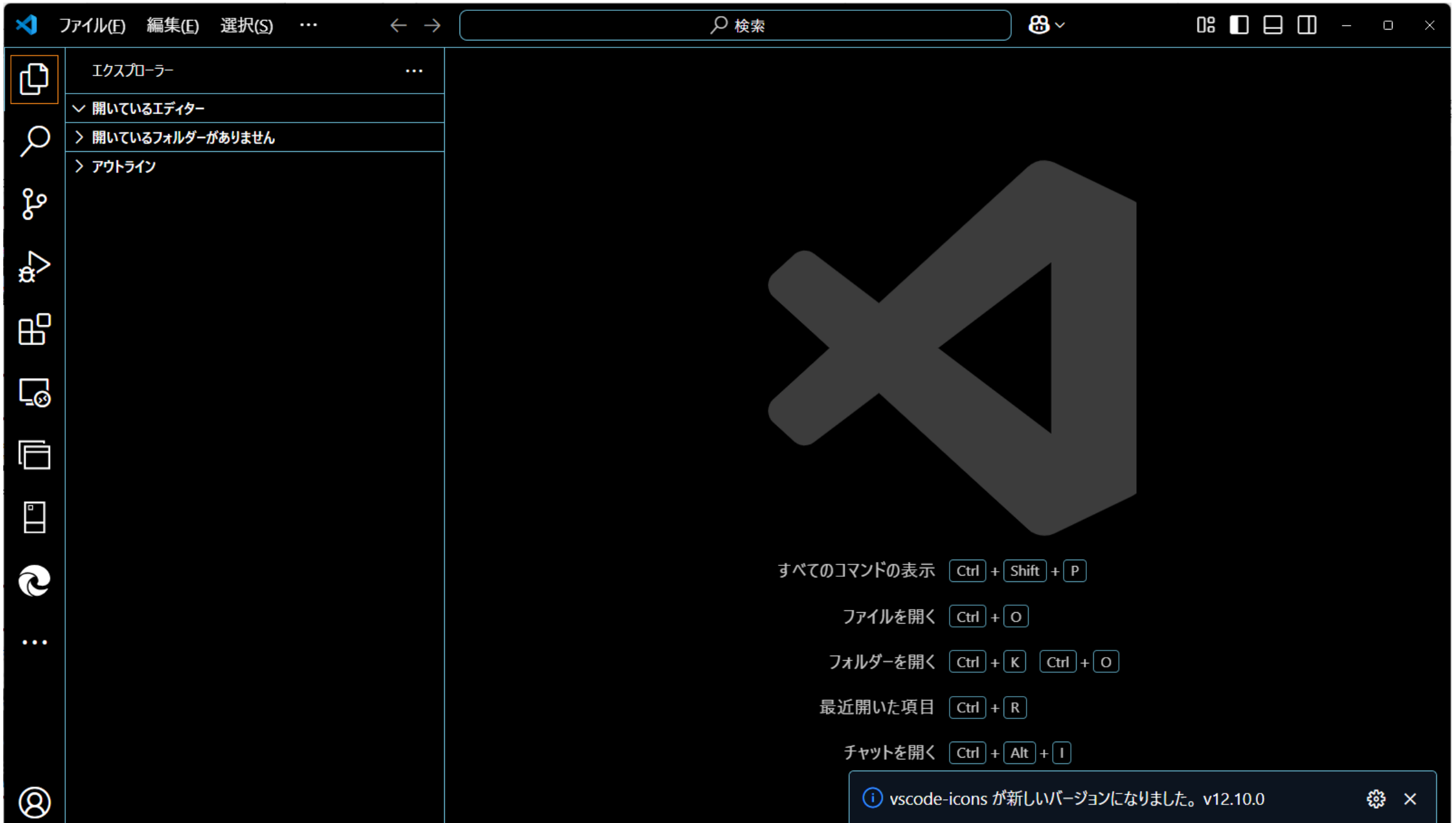


python 拡張機能: 日本語拡張パック

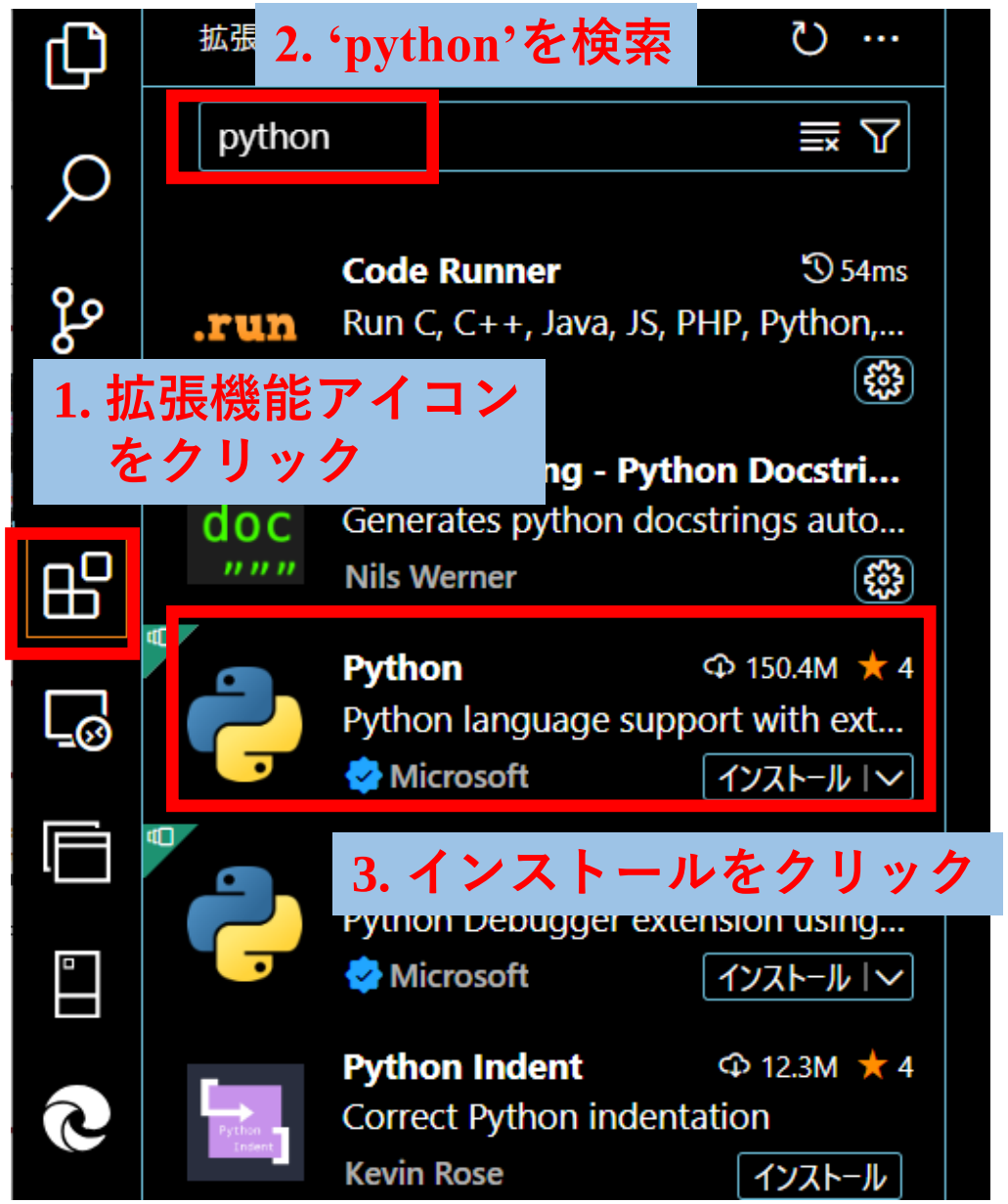


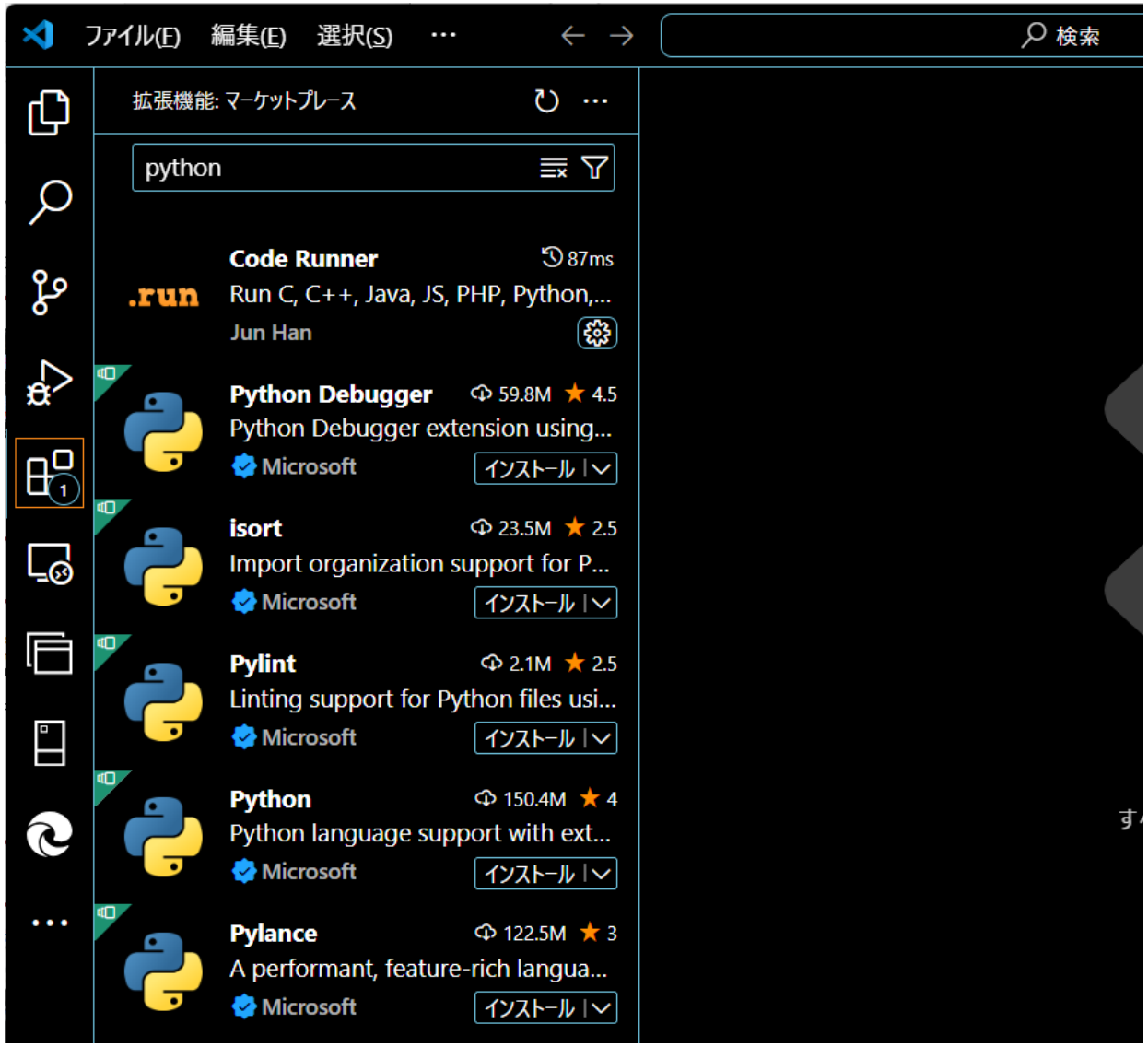


日本語Visual Code Studio

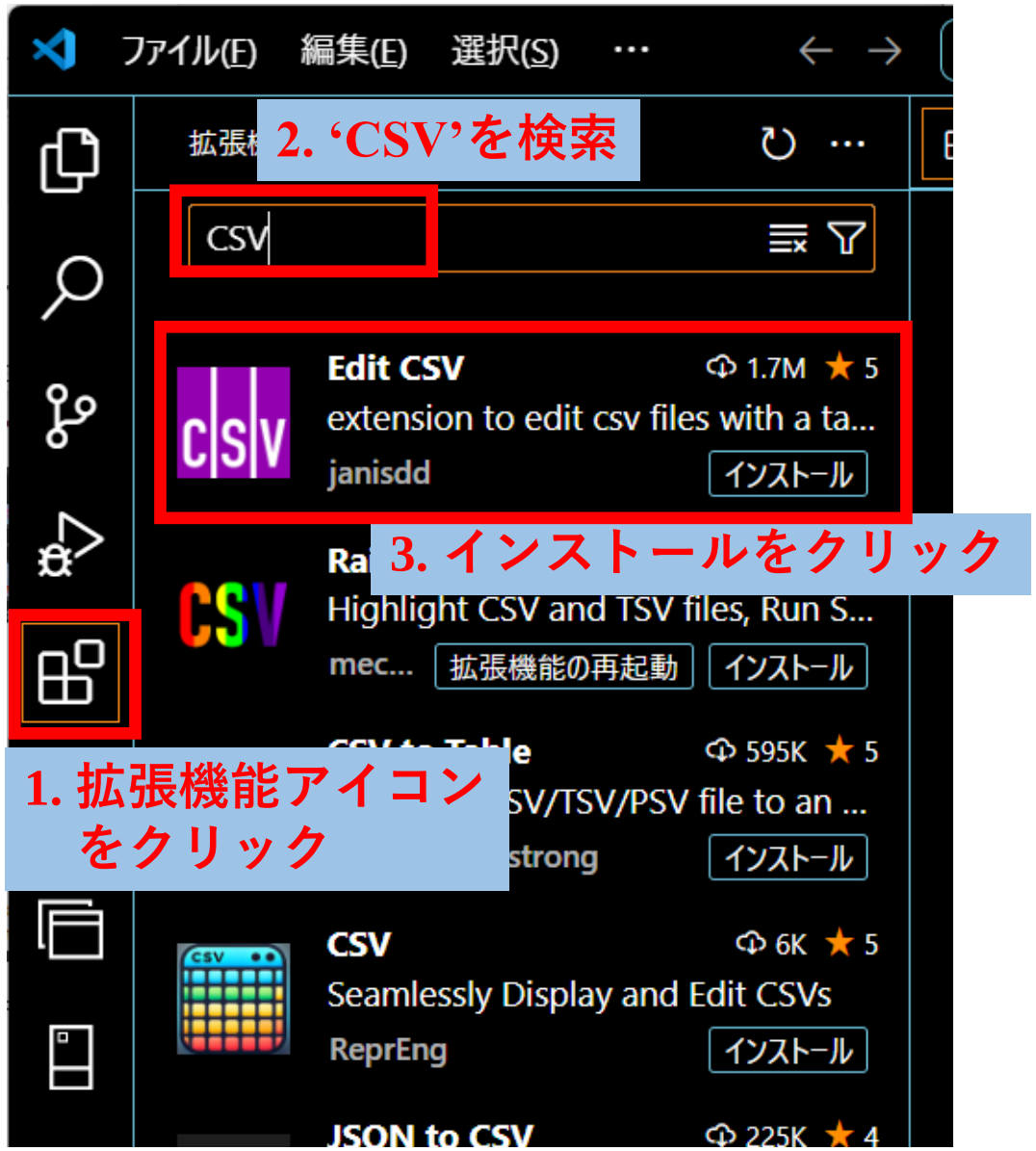


python 拡張機能

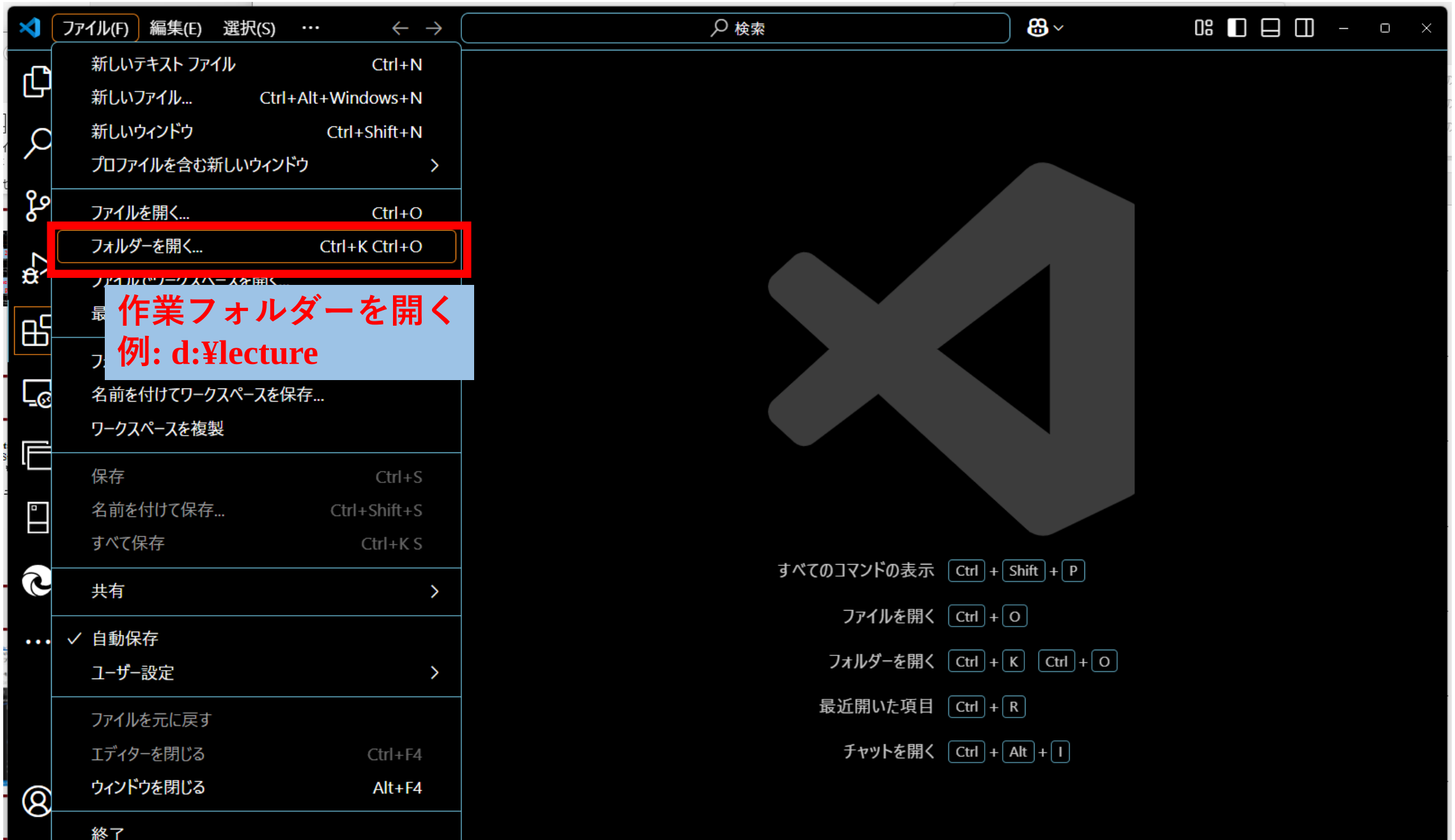

2. ‘python’を検索
python
Code Runner
54ms
Run C, C++, Java, JS, PHP, Python,...
1. 拡張機能アイコンをクリック
Generates python docstrings auto...
Nils Werner
Python
150.4M
4
Python language support with ext...
Microsoft
インストール
3. インストールをクリック
Microsoft
インストール
Python Indent
12.3M
4
Correct Python indentation
Kevin Rose
インストール


ファイル(F)
編集(E)
選択(S)
検索
拡張機能: マーケットプレース
python
Code Runner
87ms
Run C, C++, Java, JS, PHP, Python,...
Jun Han
Python Debugger
59.8M
4.5
Python Debugger extension using...
Microsoft
インストール
isort
23.5M
2.5
Import organization support for P...
Microsoft
インストール
Pylint
2.1M
2.5
Linting support for Python files usi...
Microsoft
インストール
Python
150.4M
4
Python language support with ext...
Microsoft
インストール
Pylance
122.5M
3
A performant, feature-rich langua...
Microsoft
インストール

CSV 拡張機能


ファイル(F)
編集(E)
選択(S)
2. 'CSV'を検索
CSV
Edit CSV
1.7M
5
extension to edit csv files with a ta...
janisdd
インストール
3. インストールをクリック
Highlight CSV and TSV files, Run S...
mec...
拡張機能の再起動
インストール
595K
5
SV/TSV/PSV file to an ...
strong
インストール
1. 拡張機能アイコンをクリック
CSV
6K
5
Seamlessly Display and Edit CSVs
ReprEng
インストール
JSON to CSV
225K
4

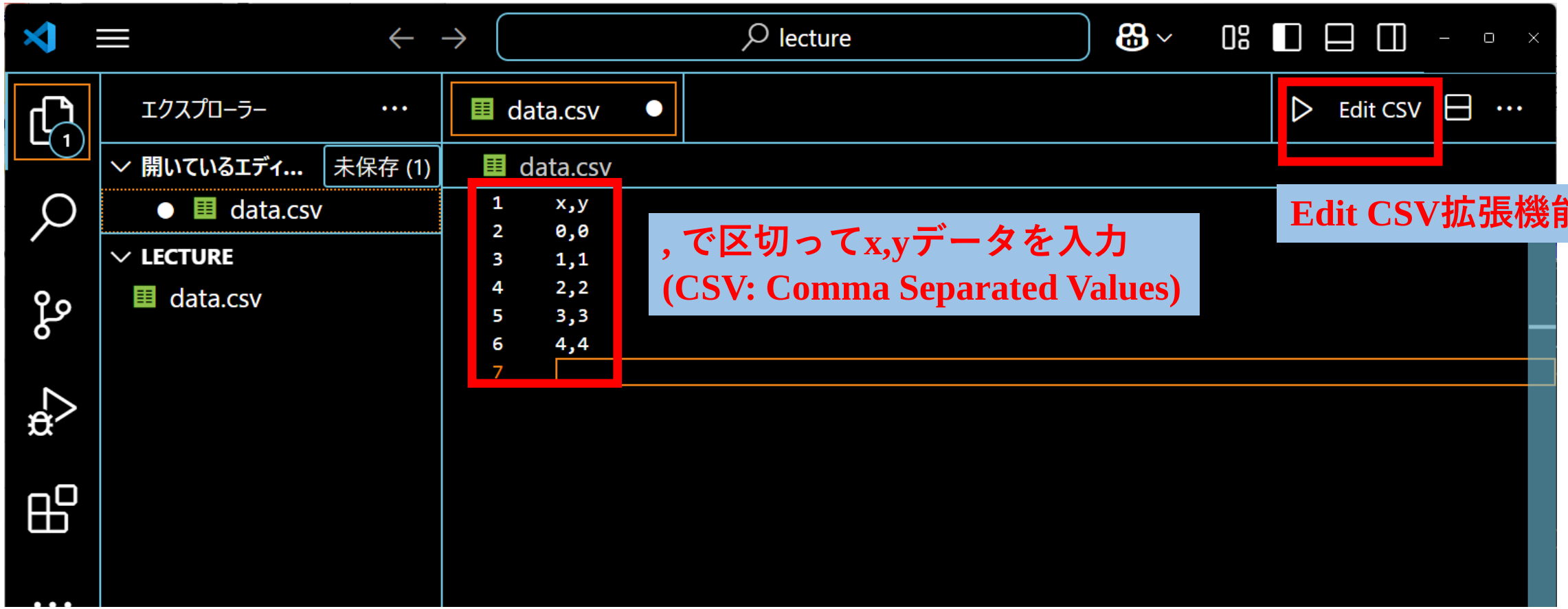
作業フォルダーに移動


ファイル(F)
編集(E)
選択(S)
検索
新しいテキスト ファイル Ctrl+N
新しいファイル... Ctrl+Alt+Windows+N
新しいウィンドウ Ctrl+Shift+N
プロファイルを含む新しいウィンドウ
ファイルを開く... Ctrl+O
フォルダーを開く... Ctrl+K Ctrl+O
作業フォルダーを開く
例: d:¥lecture
名前を付けてワークスペースを保存...
ワークスペースを複製
保存 Ctrl+S
名前を付けて保存... Ctrl+Shift+S
すべて保存 Ctrl+K S
共有
自動保存
ユーザー設定
ファイルを元に戻す
エディターを閉じる Ctrl+F4
ウィンドウを閉じる Alt+F4
終了
すべてのコマンドの表示 Ctrl + Shift + P
ファイルを開く Ctrl + O
フォルダーを開く Ctrl + K Ctrl + O
最近開いた項目 Ctrl + R
チャットを開く Ctrl + Alt + I

CSVファイルを作って動作確認

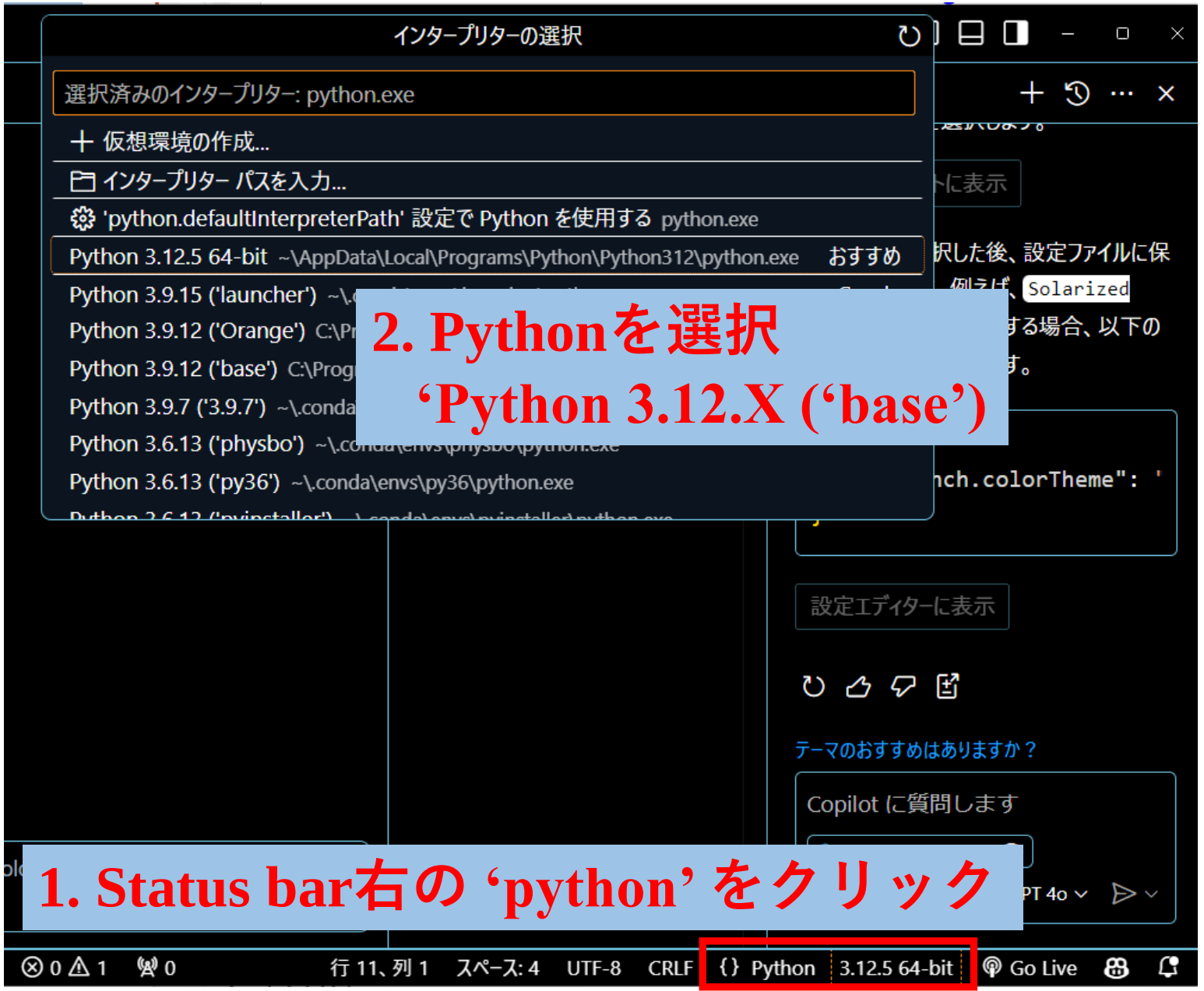
CSVファイルを作って動作確認

Tips:
Ctrl+; で拡大
Ctrl+- で縮小表示
(メニューからショートカットがわかる)



Edit CSV拡張機能

.pyファイルを実行するpython.exeの選択



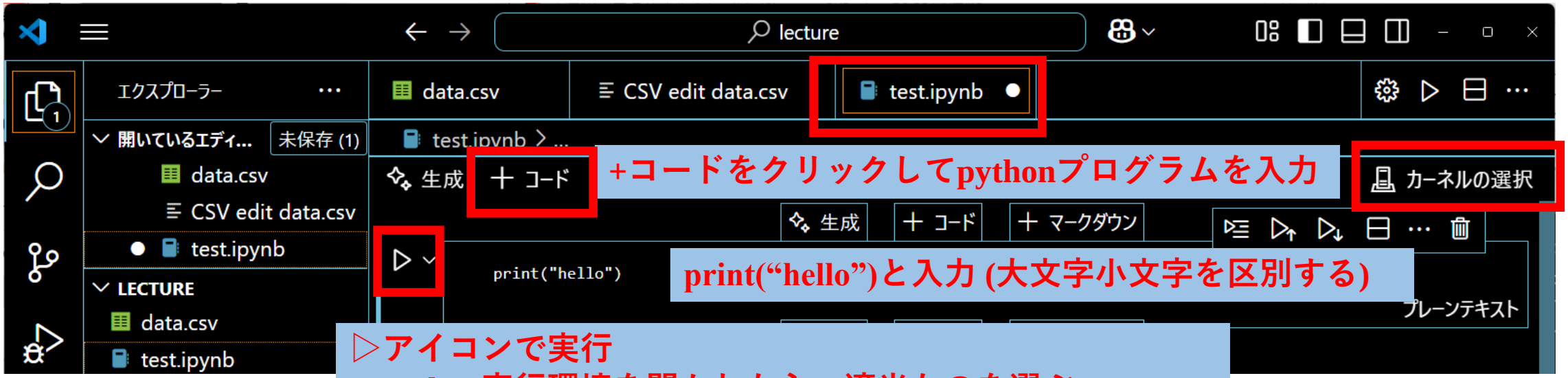
pythonプログラムを動かしてみる

Jupyter notebookを使用:

- python付属の Jupyter notebook
- Jupyter Lab
- Google Colab
- **Visual Studio Codeのpython拡張機能**

Jupyter notebookファイルを作って動作確認

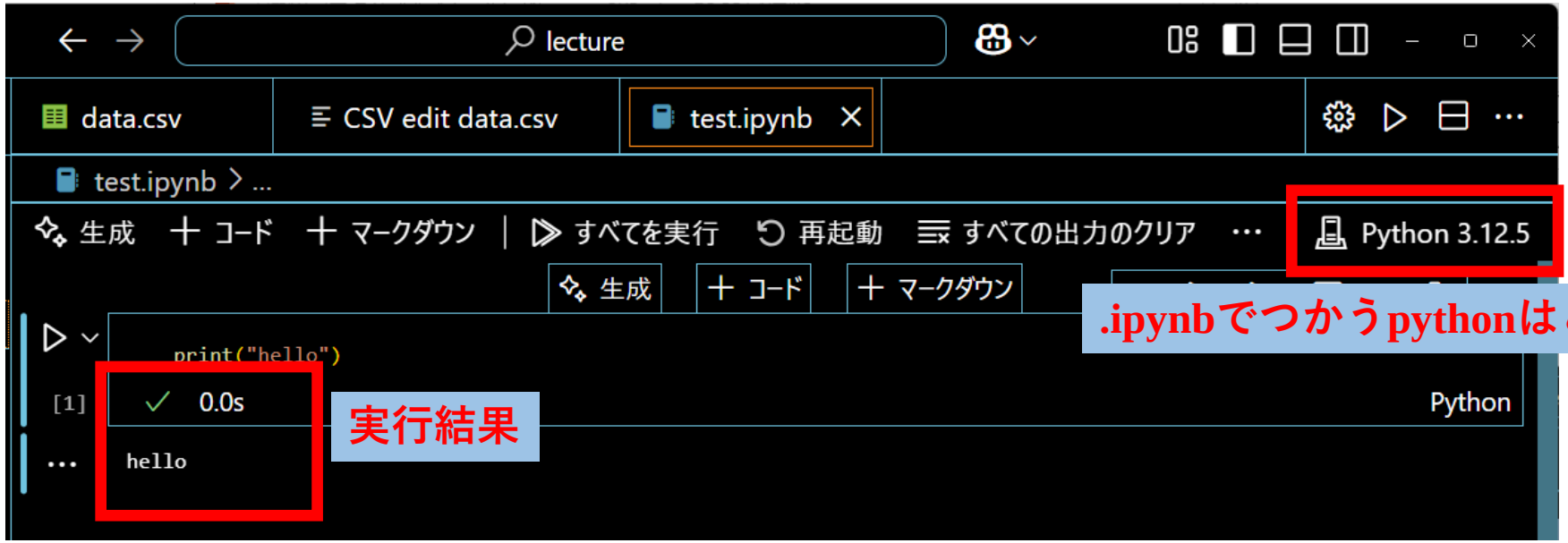
Jupyter notebookファイルを作って動作確認



+コードをクリックしてpythonプログラムを入力

print(“hello”)と入力 (大文字小文字を区別する)

▷アイコンで実行
python実行環境を聞かれたら、適当なのを選ぶ
ipykernelが必要と言われたら、インストールをクリック



実行結果

.ipynbでつかうpythonはここで変更できる

Jupyter notebook: トラブルシューティング

“▷アイコンで実行” が働かないとき
PCによって環境が違う／ネットワークアクセスのタイミングの違い により、
Jupyter notebookがうまく働かないことがあった (2025/1/10)

- ターミナルを開いて
  > pip install ipykernel
  で、ipykernelライブラリをインストールする
- VS Codeを再起動する

いろいろ試してみよう


test.ipynb
LECTURE
data.csv
test.ipynb
print("hello")
[1]
0.0s
Python
hello
3*3
[2]
0.0s
Python
9
Jupyterの場合、
数式を入力するだけで結果が出力される
(pythonプログラムの場合は print() が必要)
exp(1)
[3]
0.2s
Python
NameError
Traceback (most recent call last)
Cell In[3], line 1
----> 1 exp(1)
NameError: name 'exp' is not defined
exp(1)を計算しようとしてら、not definedエラーがでた
コード セルの追加
生成
コード
マークダウン
アウトライン
Launchpad
1
1
0
Spaces: 4
セル 3 の 3
Go Live

Pythonでは、基本機能以外はimportで読み込む

```
exp(1)
```

[3] 0.2s Python

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 exp(1)

NameError: name 'exp' is not defined
```

```
from math import exp
exp(1)
```

[4] 0.0s

```
2.718281828459045
```

Mathライブラリにはいろいろな数学関数が入っている
expを使う場合は、
from math import exp
とする
その他、log, log10, sin, cos, acosなど

Ctrl+Sで保存

0 Spaces: 4 セル 4 の 4 Go Live

どんな数学関数が使えるか: 生成AIに聞いてみる

プロンプト: python mathライブラリで使える数学関数を全て挙げてください

回答:

Pythonのmathライブラリには、多くの便利な数学関数が用意されています。以下はそのリストです：

- acos: 逆余弦関数
- acosh: 双曲線逆余弦関数
- asin: 逆正弦関数
- asinh: 双曲線逆正弦関数
- atan: 逆正接関数
- atan2: 座標平面における点の角度を計算
- atanh: 双曲線逆正接関数
- ceil: 切り上げ
- comb: 組み合わせ（コンビネーション）を計算
- copysign: 数の符号をコピー
- cos: 余弦関数
- cosh: 双曲線余弦関数
- degrees: ラジアンを度に変換
- dist: 2点間の距離を計算
- erf: 誤差関数
- erfc: 補誤差関数
- exp: 指数関数
- expm1: exp(x) - 1を計算
- fabs: 絶対値
- factorial: 階乗
- floor: 切り捨て
- fmod: 浮動小数点剰余
- frexp: 倍精度浮動小数点数を仮数と指数のタプルに分解
- fsum: 高精度で全ての項を合計
- gamma: ガンマ関数
- gcd: 最大公約数
- hypot: ユークリッド距離を計算
- isclose: 2つの数が近いかどうかを判定
- isfinite: 数が有限かどうかを判定
- isinf: 数が無限かどうかを判定

どんな特殊関数が使えるか: 生成AIに聞いてみる

プロンプト: pythonのライブラリで使える特殊関数を教えてください。コピペできる形で回答してください

回答:

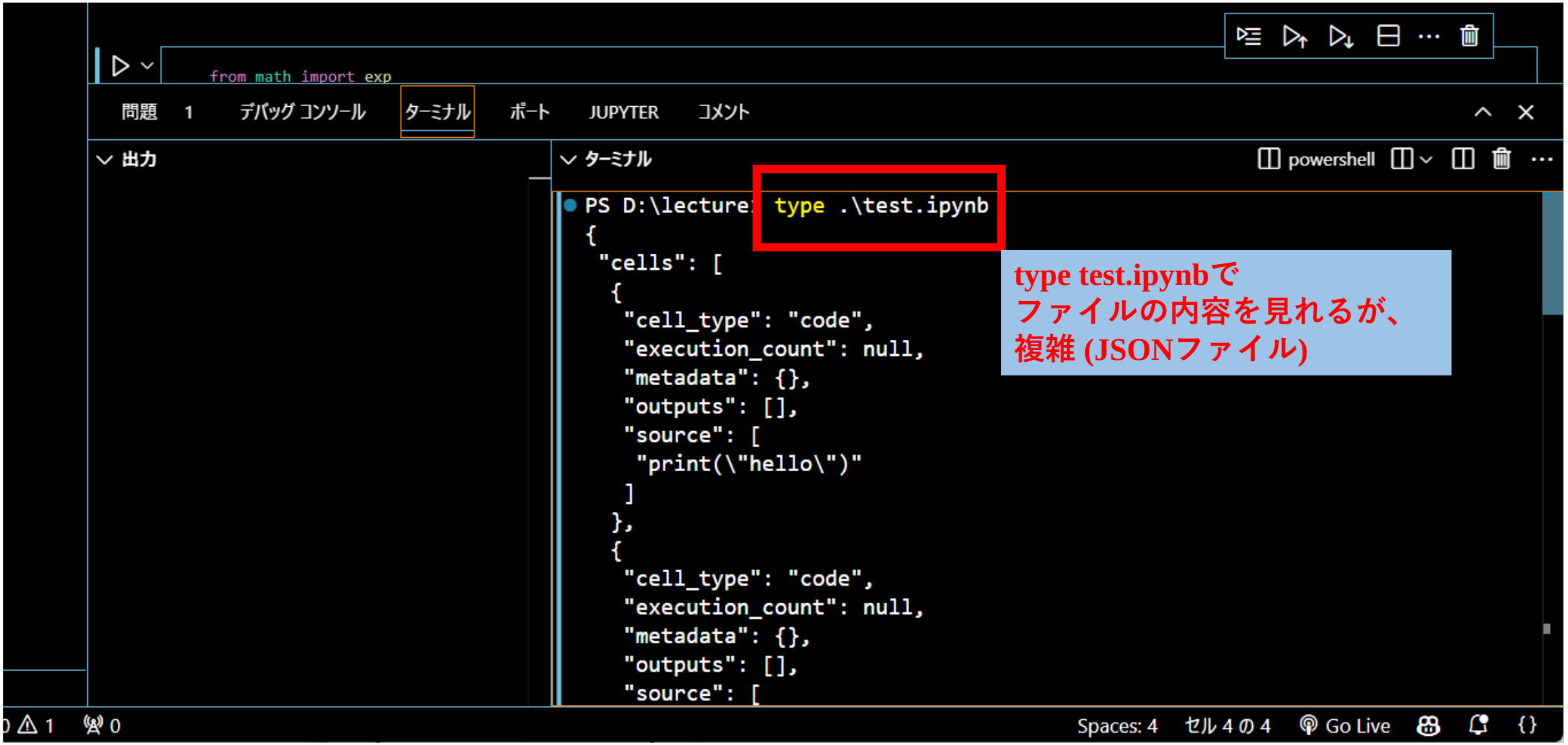
Pythonのscipy.specialモジュールでは、多くの特殊関数が利用できます。以下にリストアップしましたので、コピペしてご利用いただけます：

```
from scipy.special import (
    airy, airye, bdtr, bdtrc, bdtri, bei, beip, ber, berp, besselpoly, beta,
    betaln, betainc, betaincinv, binom, boxcox, boxcox1p, boxcox_normmax,
    boxcox_llf, btdtr, btdtri, btdtrc, cbrt, chdtr, chdtrc, chdtri, chndtr,
    chndtrix, chndtridf, chndtrinc, cosdg, cosm1, cotdg, dawsn, ellipj,
    ellipk, ellipkm1, ellipe, ellipeinc, ellipkinc, erf, erfc, erfcinv,
    erfcx, erfi, erfinv, exp1, exp10, exp2, expi, expm1, fdtr, fdtrc, fdtri,
    fresnel, gamma, gammainc, gammaincinv, gammaln, gdtr, gdtrc, gdtri,
    gdtrix, hankel1, hankel2, hyp0f1, hyp1f1, hyp2f1, hyp3f0, hyp3f2, hyp2f0,
    iv, ive, ivp, j0, j1, jn, jv, jve, jvp, kei, keip, ker, kerp, kl_div, kn,
    kv, kve, kvp, lmbda, log1p, log_ndtr, loggamma, logit, logsumexp, lpmv,
    mathieu_a, mathieu_b, mathieu_cem, mathieu_sem, mathieu_modcem1,
    mathieu_modcem2, mathieu_modsem1, mathieu_modsem2, mathieu_even_coef,
    mathieu_odd_coef, modfresnelp, modfresnelm, nbdtr, nbdtrc, nbdtri, ndtr,
    ndtri, nrdtrimn, nrdtrisd, obl_ang1, obl_ang1_cv, obl_cv, obl_rad1,
    obl_rad1_cv, obl_rad2, obl_rad2_cv, pbdv, pbvv, pbwa, pdtr, pdtrc, pdtri,
    pdtrik, perm, poch, pro_ang1, pro_ang1_cv, pro_cv, pro_rad1, pro_rad1_cv,
    pro_rad2, pro_rad2_cv, psi, rgamma, round, shichi, sici, smirnov,
    smirnovi, spence, stdtr, stdtridf, stdtrinc, struve, tandg, tklmbda,
    wofz, xlog1py, xlogy, y0, y1, yn, yv, yve, yvp
)
```

ターミナルを起動


lecture
data.csv
CSV edit data.csv
test.ipynb
ターミナルを起動
test.ipynb > ...
生成
コード
マークダウン
すべてを実行
再起動
すべての出力のクリア
Python 3.12.5
3*3
[ ]
Python
exp(1)
[ ]
Python
from math import exp
exp(1)
[ ]
Python

ターミナルを起動: test.ipynbの内容を見る



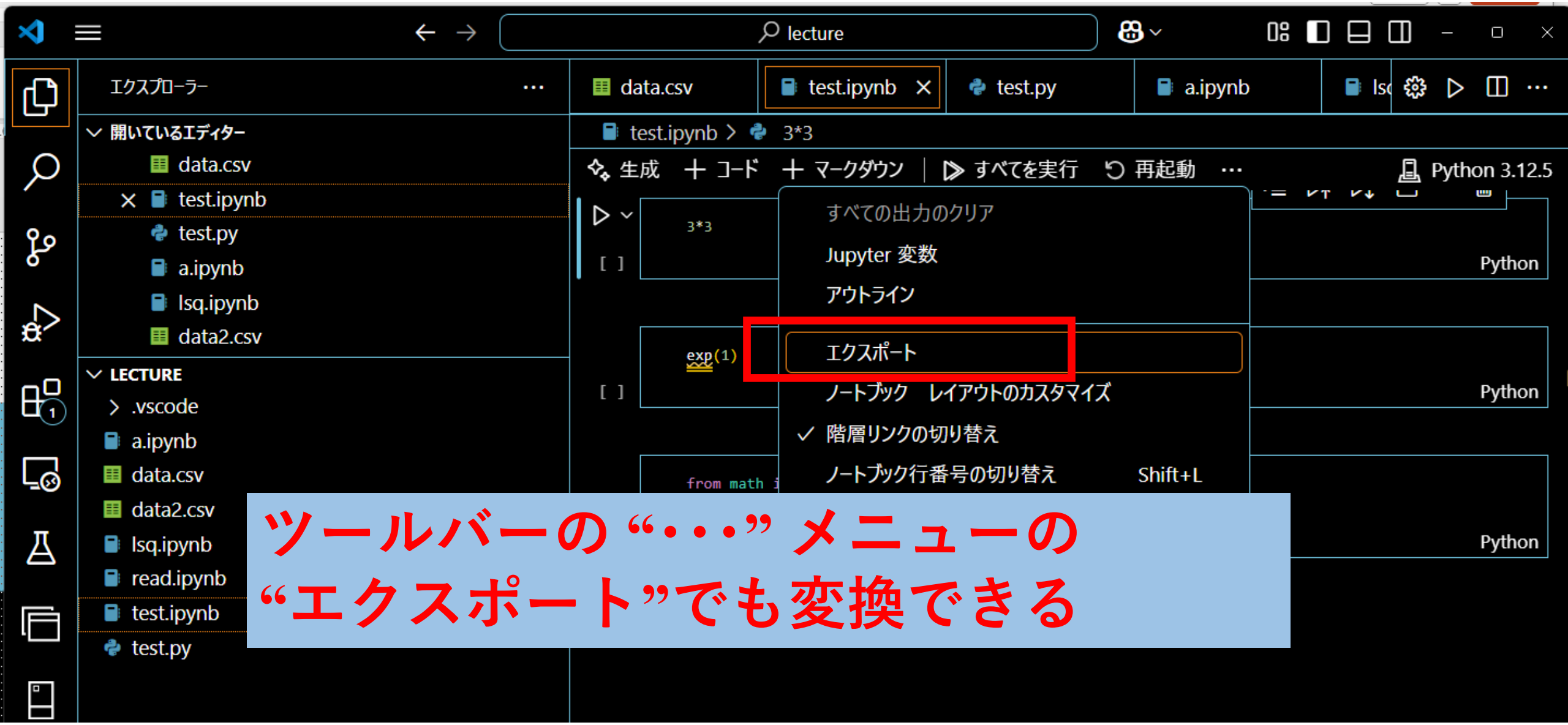
ターミナルを起動: pythonプログラムに変換

```
ValueError: Please specify an output format with '--to <format>'.
The follo                                                    m', 'html',
, 'markdo                                                    ng', 'rst',
', 'slide
PS D:\lecture> jupyter nbconvert --to script .\test.ipynb
[NbConvertApp] Converting notebook .\test.ipynb to script
[NbConvertApp] Writing 144 bytes to test.py
PS D:\lecture>
```

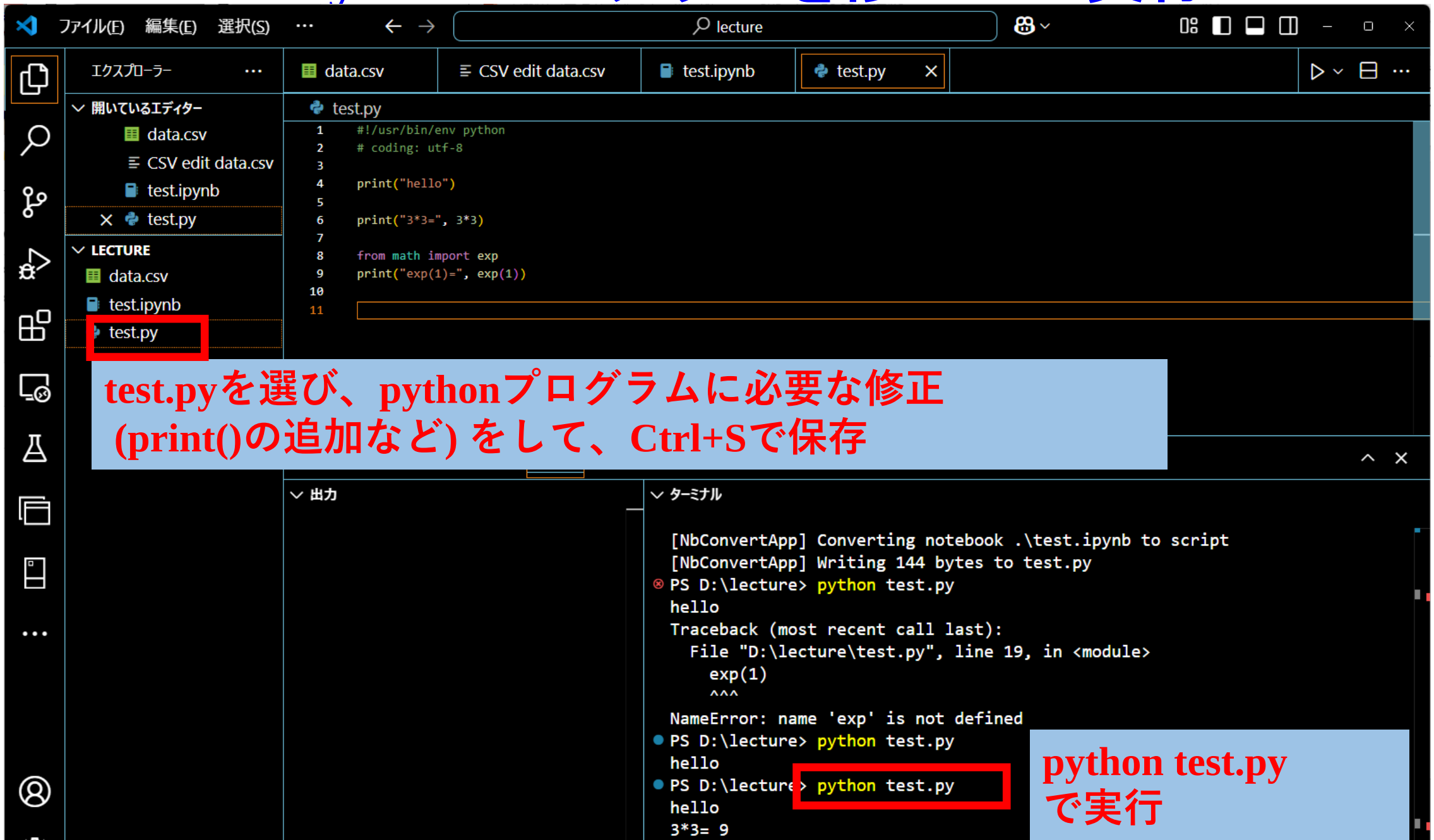
jupyter nbconvert --to script test.ipynb
で、pythonプログラムtest.pyへ変換

Spaces: 4　セル 4 の 2　Go Live

エクスポートメニューでpythonプログラムに変換



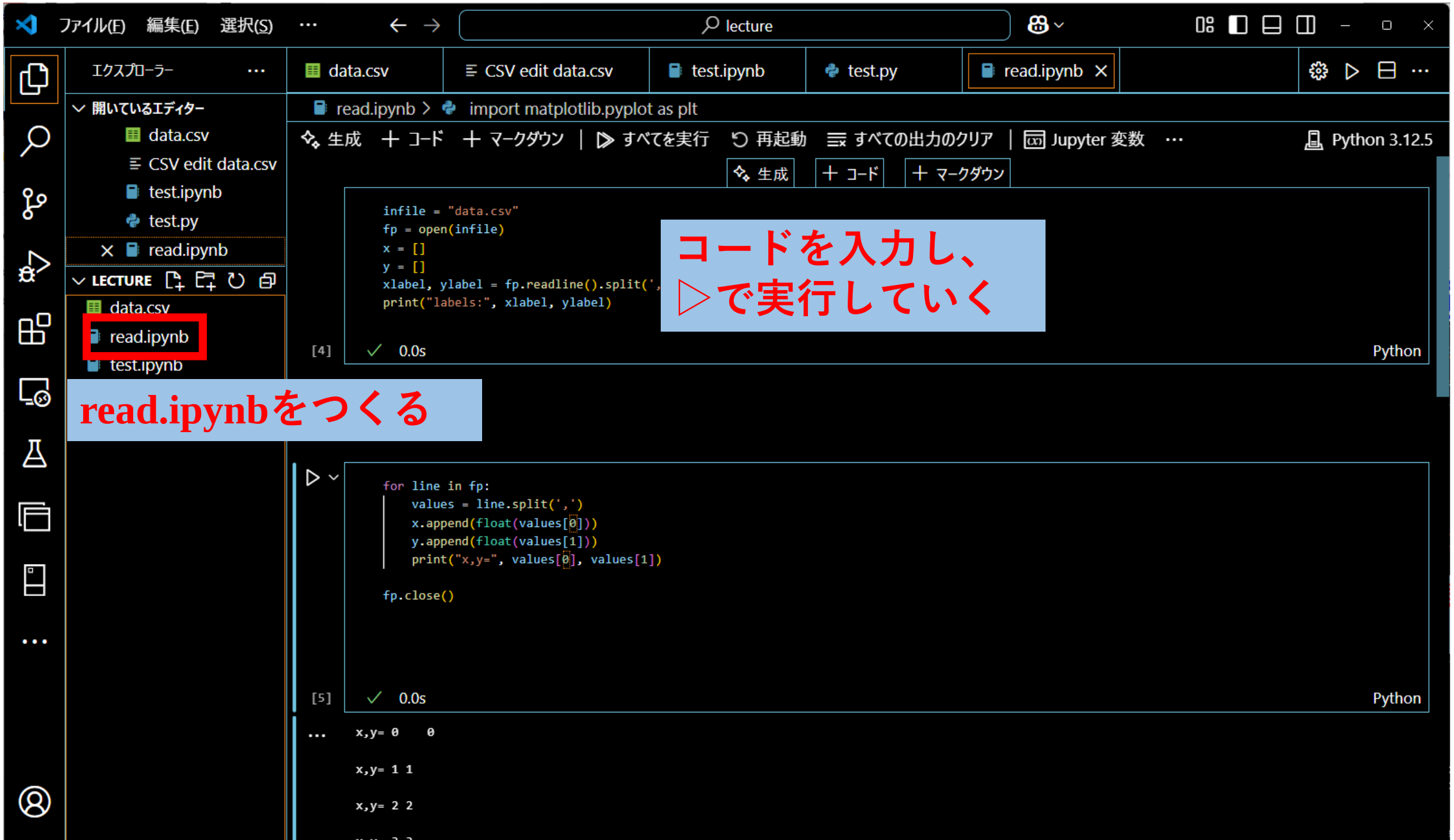
Pythonプログラムを修正して実行



もう少し本格的なプログラムを作ってみよう

- data.csvを読み込み、実数型 (浮動小数点型) に変換してリスト変数 (配列) xs, ys に入れる
  - 注: 変数名はわかりやすく
    - 例: 単一の値を保存する: x, y
    - 複数の値を保存する (リスト): xs, ys
- xs, ys のグラフを描く (matplotlib)

Pythonプログラムを修正して実行



Pythonプログラム例: テキストファイルを読み込む

注: # 以降はコメント文。ここでは入力しなくてもいい

```
infile = ”data.csv”      # infileという名前の変数に ファイル名 data.csv を設定
                         # 文字列は ’ ’ あるいは ” ” で囲む

fp = open(infile)        # infile を読み込みモードで開き、ファイルハンドルをfpに代入
xs = []                  # xs データのリスト変数を作る
ys = []                  # ysデータのリスト変数を作る

line = fp.readline()                  # 最初の１行を読み込み、lineという変数に代入
xlabel, ylabel = line.split(‘,’)      # line文字列を , で分離し、リスト変数として返す
                                      # 正確にはタプルとして返るが、ここでは気にしない
                         # line.split(‘,’) の戻り値を、xlabel, ylabelという形で受け取ると、
                         # 戻り値の最初の要素が xlabel, 2つ目の要素が ylabel に入る
print(”labels:”, xlabel, ylabel)          # print()で出力して確認する
```

Pythonプログラム例: テキストファイルを読み込む

```
for line in fp:                 # ファイルハンドルでforを回すと、1行ずつlineに入る
    values = line.split(’,’)    # lineを , で区切り、valuesリスト変数に代入
    x = float(values[0])        # valuesの第1要素values[0]を実数に変換してx変数に代入
    y = float(values[1])        # valuesの第2要素values[1]を実数に変換してy変数に代入
    xs.append(x)                # xをxsリスト変数に追加
    ys.append(y)                # yをysリスト変数に追加
    print(“x, y=”, x, y)        # 念のため、x,y の値を出力して確認

fp.close()      # 重要: ファイルハンドルをopenしたら、不要になったときに必ずcloseする
```

関連:

```
  with open(infile) as fp:
      …
```

とすると、withブロックが終了したところでfp.close()が実行されるので、
一般的にはこの書き方が推奨される

Pythonプログラム例: グラフを描く

```
import matplotlib.pyplot as plt
        # グラフを描くお約束。Matplotlibのpyplotモジュールをpltという名前で読み込む

plt.plot(xs, ys)          # xs, ysリスト変数をx軸、y軸のデータにしてグラフを描く
plt.xlabel(xlabel)        # x軸のラベルにxlabelを設定
plt.ylabel(ylabel)        # y軸のラベルにylabelを設定
plt.show()                # グラフを描画
```

最小二乗法のプログラム

data.csvには精確に直線に乗っているデータしかないので、
ノイズを入れたデータを **data2.csv** としてつくる

data2.csv:

(MS copilotで
” y=3xに従い、ノイズを含むデータのCSVフォーマットのテキストを出力してください。xの範囲は0~10で1置きで”
で作成)

x,y
0,0.34
1,2.65
2,5.12
3,8.21
4,12.03
5,14.90
6,18.24
7,20.93
8,24.75
9,27.58
10,30.45

read.ipynbを“名前を付けて保存”で lsq.ipynbで保存

```
infile = ”data2.csv”              # ファイル名を修正
fp = open(infile)
xs = []
ys = []
xlabel, ylabel = fp.readline().split(',')
print("labels:", xlabel, ylabel)

for line in fp:
    values = line.split(',’)
    x = float(values[0])
    y = float(values[1])
    x.append(x)
    y.append(y)
    print("x, y=", x, y)

fp.close()
```

read.ipynbを“名前を付けて保存”で lsq.ipynbで保存

```
# 多項式の最小二乗には、numpyモジュールの polyfit() 関数を使う

import numpy as np
a, b = np.polyfit(xs, ys, 1)
        # xs, ysデータに対して、1次式でフィッティングし、係数をa, bに代入
print(“a, b=”, a, b)                    # aが傾き、bがy切片    y = a * x + b

#a,bから回帰データを補間。xの範囲は0~10、0.1間隔
xcal = np.arange(0, 10, 0.1)
ycal = a * xcal + b                     # numpyを使うと、配列の値を一度に計算して配列を返す
print(“xcal=“, xcal)
print(“ycal=“, ycal)

import matplotlib.pyplot as plt
plt.plot(xs, ys, ‘o‘)                   # ‘o‘ により、データを線でつながず、〇で散布図を描画
plt.plot(xcal, ycal, ‘-‘)               # ‘-‘ により、データを線でつないで折れ線グラフにする
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.show()
```

多項式線形最小二乗法: numpyライブラリのpolyfit()関数

ChatGPTへの質問: pythonで logN = a + b * Tinv にフィッティングするプログラム

```
# numpyライブラリを 別名 np でimport
import numpy as np


# numpy.polyfit() の呼び出し。
# numpyは 別名 np でimportしているので、np.polyfit() で呼び出す
a, b = np.polyfit(Tinv, logN, 1)
```

```
# numpyライブラリから polyfit を import
from numpy import polyfit


# polyfit() だけで呼び出せる
a, b = polyfit(Tinv, logN, 1)
```

でも、この時の回答のプログラムは間違っていた

numpy polyfitを検索してみる

Bingで “numpy polyfit” を検索

NumPy
https://numpy.org › stable › generated · このページを訳す

numpy.polyfit — NumPy v1.26 Manual

公式のAPIリファレンス

Relative condition number of the fit. Singular values smaller than this relative to the largest singular value will be ignored. The default value is len(x)*eps, ...

Numpy.poly1d · Numpy.polyval · Numpy.polyder · Convenience Classes

Qiita
https://qiita.com › Python

知恵袋系: Qiita, Quoraの質が高い

Numpy.polyfit を使ったカーブフィッティング #Python

2019/05/10 — 様々な補間法と最小2乗法をPythonで理解する のうち、「**Numpy.polyfit** を使ったカーブフィッティング」を、実データっぽい模擬データを解析するように ...

numpy polyfitを検索してみる

Bingで “numpy polyfit” を検索

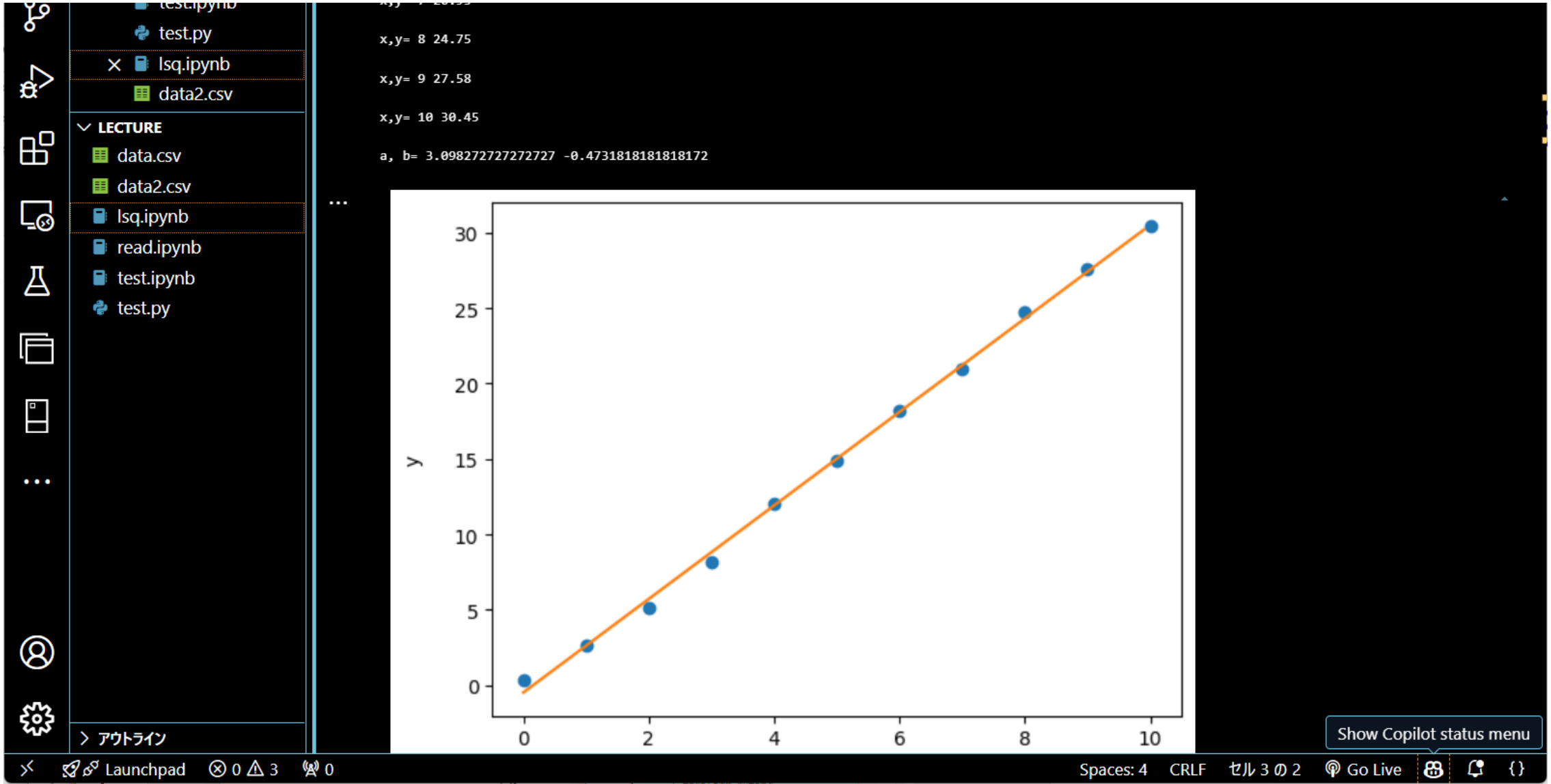
https://chayarokurokuro.hatenablog.com/entry/2020/01/22/190133

```python
# 近似線の係数を算出
k = np.polyfit(x, y, 1)
# 変数に置き換えとこ
a, b = k


# 適当にx軸の両端の値
x_reg = np.array([0, 25])


# 直線の式を作って配列xを代入
y_reg = a * x_reg + b
```

polyfitで返ってくる係数は、
y = a * x + b
の順番

ChatGPTは、y = a + b * x の係数を
a, b = np.polyfit(Tinv, logN, 1)
で受け取っていたので、a と b がひっくり返っている

実行結果


test.py
lsq.ipynb
data2.csv
LECTURE
data.csv
data2.csv
lsq.ipynb
read.ipynb
test.ipynb
test.py
アウトライン
x,y= 8 24.75
x,y= 9 27.58
x,y= 10 30.45
a, b= 3.098272727272727 -0.4731818181818172
y
0
5
10
15
20
25
30
0
2
4
6
8
10
Launchpad
0 3
0
Spaces: 4
CRLF
セル 3 の 2
Go Live
Show Copilot status menu

より進んだ問題

- 2次以上の多項式の場合
- 偶数次数のみの多項式の場合
- データにもっと大きなノイズを加えてみる
- Jupyterプログラムをpythonプログラムに変換して実行してみる

次回

- curve_fit()を使った一般関数の(線形)最小二乗法
- 非線形最小二乗法
- minimize()を使った最小化問題

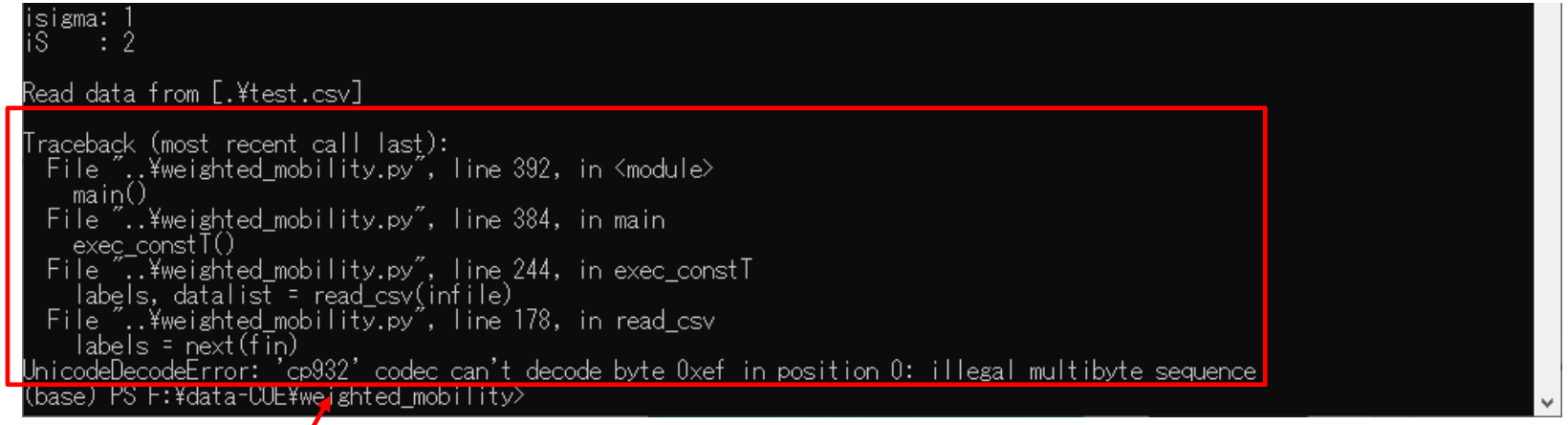
自習に生成AIを使う

プロンプト: 以下のプログラムが何をしているか、わかりやすく説明してください
＊＊ここにpythonプログラムをコピペ＊＊

回答:
このプログラムは、CSVファイルからデータを読み取り、線形回帰を実行し、その結果をプロットするものです。
以下に詳細を説明します:
1. CSVファイルの読み込み:
   ・data2.csvというファイルを開きます。
   ・空のリスト xs と ys を初期化して、CSVファイルから読み取る x 値と y 値を保存します。
   ・ファイルの最初の行を読み取り、x 軸と y 軸のラベルを取得して xlabel と ylabel に格納します。
2. データの処理:
   ・残りの行をループで読み取ります。
   ・各行をカンマで分割し、x 値と y 値を取得します。
   ・これらの値を浮動小数点数に変換して、それぞれ xs と ys のリストに追加します。
   ・各行の x 値と y 値を表示します。
以下、続く

講義で聞き逃したことも質問してみましょう

エラー・バグ報告の仕方



Pythonなどのinterpreter型言語は、エラーを起こしたプログラム行数、内容、原因などを表示してくれる。

バグレポート: 以下の内容をまとめて報告すること

1. OS (Windows10, OS Xなど) とプログラム環境 (python、Jupyterなど）とバージョン
2. 実行したプログラムの**バージョン (不明な場合はファイルのタイムスタンプ)**。
   あるいはプログラム自身を添付
3. 入力ファイル、出力ファイルを添付
4. エラーメッセージのスクリーンショット、
   あるいはコンソール出力のファイル

コンソール出力の取り方 (リダイレクト > を使ってファイルに落とす)

c:> 実行コマンドライン > out.txt

とすると、”実行コマンドライン” のコンソール出力が out.txtに保存されます。

生成AI

生成AI (大規模言語モデル LLM) の利用

VSCodeの**ChatGPT**拡張機能を使うときは、**OpenAI API Key** (課金) が必要

1. https://platform.openai.com/docs/overview
2. ただし、無料のMicrosoft 365 Copilot でも Chat GPT4を使っているので、
   それ以上のGPTモデルを使う必要が無ければ、API Keyの課金は勧めない

Github copilotの利用

VSCodeの**github copilot**拡張機能を使うと、コード内でコメントを書いてコード生成ができる

1. 科学大の学生は無料で使える
   https://docs.github.com/ja/education

Pythonの文法

講義Webを参照のこと

http://d2mate.mdxes.iir.isct.ac.jp/Lecture/FundamentalsCMS

D-MatL Top
original HTML | display HTML

2024年度Q4 材料計算科学基礎

python
プログラム例と文法

講義予定
#01 ～ #08:
・機械学習の基礎・演習
#09:
・Visual Studio Codeによる環境構築
・pythonプログラム実習:
polyfit()を用いた多項式最小二乗プログラムの作成
#10:
・より複雑な最小二乗法
線形最小二乗法
非線形最小二乗法
#11:
・生成AIを利用したプログラム作成
#12～#14:講義

python文法と作法

ファイル名の作法

▶ 詳細

python特有の規則: インデント

▼詳細　詳細をクリックすると内容が展開される

- ブロックはインデント (行頭の空白文字数) の数で決まる
- インデント数が同じかわからなくなるので、**TABを使わない**

▶ プログラムコード例:

コメント文

▼詳細

- # 以降、行末まではコメント文
- document stringを複数行のコメント文として使うことも多い

```
'''
 コメント文
'''
```

注: ''' ～ '''は正しくはコメント文ではないため、インデント規則を受ける

実行順序

予備スライド

pythonの作法と文法

- ファイル名の拡張子は .py にするのが良い
- Jupyter notebookを使うと、対話形式でプログラムを作れる。拡張子は .ipynb
- .ipynbファイルは、
  jupyter nbconvert --to script test.ipynb
  でpythonプログラムに変換できる。この場合は、test.ipynbをtest.pyに変換する
- pythonプログラムはpythonコマンドで実行する
  python test.py

- pythonでは、行頭の空白文字の数 (インデント) が重要。
  ・同じブロックではインデントの数を合わせる
  ・一段深いブロックではインデントの数を増やす
  ・ブロックが終了したらインデントの数を元に戻す
  ・紛らわしいので、TAB は使わない

Pythonに特有の規則: インデントによるブロック定義

- **ブロックはインデント (行頭の空白文字数) の数で決まる**
  注意 : TABを使わないように注意する
  TABキーを一定数の空白文字に置き換えるエディタを使う

以下のコードで '.' はスペースの意味です

```
for i in [1, 2, 3, 4, 5]:
....x = 2.0 + i * 0.1

....print(f"i={i}  x={x}")

for s in ["a", "b", "c", "d", "e"]:
..line = 'string is ' + s
..print(f"line={line}")

exit()
```

0文字インデントから開始
4文字インデント。Forブロック開始
空行は無視される
空行は無視される
0文字インデントに戻る。Forブロック終了
2文字インデント。Forブロック開始
0文字インデントに戻る。Forブロック終了

Pythonのコメント文

	python	C	Basic
インライン コメント (行末まで)	# 以降	C99以降 // 以降	Visual Basic ’ 以降
複数行 コメント	docstringを流用 *single quotation x3* ’’’ ’’’ あるいは *double quotation x3* ””” ”””	/* */	

pythonスクリプトの実行順序

コマンド python に続けて pythonスクリプトファイルパス (メインファイル) を与えて実行

> python test.py

- **メインファイル test.py の先頭から順番に実行**
- **メインファイル test.py の最後で終了**
  exit() 関数を呼び出すと途中で終了できる

- 関数定義はどこにでも書ける
  (関数内、for/if などの ブロック内でも関数を書ける)
  実行時には関数定義は読み飛ばされる
- 関数呼び出しを見つけたら関数内部へジャンプ、実行
  return をみつける or 関数ブロック終了時に呼び出し元の次の行を実行

pythonの特徴: 変数名、関数名、モジュール名、予約語

大文字と小文字を区別する

- 変数 a と変数 A は別の変数

- print(“Hello”) は正しいが、
  PRINT（“Hello”) は関数 PRINT() が見つからずエラーになる

Pythonの文法: 変数作成

- **Pythonに型定義はない** (python3.5以降は type hint として型宣言できる)
- **変数に値を代入すると、変数を作成したことになる**
- **代入した値により変数型が決まる**

a = 3	aは整数型 (int)、 3で初期化される
a = 3.0	aは実数型 (float)、3.0で初期化される
a = “3”	aは文字列型 (str)、”3”で初期化される
a = [3, 3.0, “3”]	aは配列 (list型)、 要素として 3 (int), 3.0 (float) ”3” (str) をもつ

Pythonでは自然な形で整数型と実数型を比較できる場合が多いが、
実数型として使う変数は、整数になる場合でも実数型で代入する
方がトラブルが少ない

型変換 (05-read_text_file.py)

・テキストファイルからは、基本的には **文字列型** になる

例外: openpyxl, pandas: Excel内での型に依存

numpy: 数値型しか読み込めない

・**文字列型 (str) を実数型 (float) に変換しないと、計算に使えない**

型変換関数:

文字列型 s => 実数型 v : v = float(s)

文字列型 s => 整数型 i : i = int(str)

実数型、整数型 v => 文字列型 s : s = str(v)

s = f”{v}”

入力データの確認: 画面出力 print()

基本的な書き方: print(arg1, arg2, arg3, …)

arg1, arg2, arg3を文字列に変換して順番に出力

最後に改行する *(改行したくないときは、引数の後に end = ‘’ を入れる:* 例: print(“x=“, x, end = ‘’) *)*

 # リスト変数 T N を画面に出力

print("T:", T)

print("N:", N)

実行結果 : 読みにくい

T: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160]

N: [1, 1.4142135623730951, 1.7320508075688772, 2, 2.2360679774997 9, 2.449489742783178, 2.6457513110645907, 2.8284271247461903, 3, 3.1622776601683795, 3.3166247903554, 3.4641016151377544, 3.605551275463989, 3.7416573867739413, 3.872983346207417, 4]

よく変更する変数の変更: キーボード入力 input()

ret = input(“Input>>”)

Input>>” を画面に表示し、キーボード入力を待つ
ENTERを押したら、入力した値を返す

retには 改行文字 (¥n) を含めた文字列が変える
=> 改行文字を削除するため、.strip() を使う

ret = ret.strip()　　# retから前後の空白、改行文字を削除

```
input_path = ‘input.xlsx’                          # デフォルト値を設定する
str_key = input(“Input file name [default: input.xlsx]>>”)
str_key = str_key.strip()
if str_key != ‘’:
    input_path = str_key    # 入力されたのが 空文字 でなければ input_pathを置き換える
                            # 空文字の場合は、input_pathはデフォルト値のまま
```

リスト変数 (配列) の使い方

２種類の配列型がある
・リスト型　(list)　: 要素の書き換えが可能
・タプル型 (tuple): 要素の書き換えはできない

・要素には異なる変数型を入れられる
・要素には配列を入れられる => 多次元配列を作れる

配列 (リスト) を作る

```
a = []                          # list型変数は [ ] でくくる
a.append(1.0)                   # 要素を追加
print(a[0])                     # 1番目の要素は a[0], i番目の要素は a[i-1]で参照できる
len(a)                          # 配列の要素数を返す

a = (1.0, [1.0, 3.0], “a”)      # tuple型変数は ( ) でくくる
  # tupleの要素は変更できないので、初期化時にすべての要素を与える
  # tupleの要素は変更できないので、.append()は使えない
```

繰り返し構文 (ループ)

- ループ変数は整数型や実数型
- ループ開始時にループ変数を初期化
- ループブロック終了時に変数を更新して条件判断をする
  条件が False になったらループを抜ける
  Fortranの場合: i を 1 ずつ増やしながら、
  i が 10以下の場合にdoブロックを実行
  Cの場合: for文内で定義
  for(初期化 ; ループブロック続行条件 ; ループ変数更新)

Fortranの繰り返し

```
do I = 1, 10
   print *, 'I=', I
end do
```

Cの繰り返し

```
for(i = 0 ; i < 10 ; i++) {
   printf("i=%d¥n", i)
}
```

```
for(x = 0.0 ; i < 10.0 ; x += 2.0) {
   printf("i=%d  x=%lf¥n", I, x)
}
```

Pythonに特有の文法: 繰り返しはiterator (反復子) で回す

01c_fitting_gpt_rev2.py のExcelファイル読み込み部分:

```
for row in sheet.iter_rows(min_row=2, values_only=True):
    T.append(row[0])
    N.append(row[2])
```

Pythonの繰り返し構文

```
for 変数 in iterator:    # Pythonの繰り返し変数は iterator
    forブロック開始      #    iterator: 呼び出されるたびに次の値を 変数 に返し、
                         #              値がなくなったら None を返す
```

上の例では、sheet.iter_rows() が iterator を返す

Pythonに特有の文法: 繰り返しはiteratorで回す

例1: 配列変数を一つずつ実行

```
i_list = [1, 2, 3, 4, 5]
for i in i_list:
    print(f"i={i}")
```

例2:

```
str_list = ['a', 'b', 'c', 'd', 'e']
for s in s_list:
    print(f"s={s}")
```

例3: 指定した範囲の整数を一つずつ実行

```
for i in range(0, 10):
    print(f"i={i}")
```

例5: 指定した範囲の実数を一つずつ実行

```
for v in numpy.arrange(0.0, 10.0, 2.0):
    print(f"v={v}")
```

例6: ファイルを1行ずつ読み込む

```
fp = open('input.txt', 'r')
for line in fp:
    print(f"line: {line}")
```

Pythonに特有の文法: 繰り返しはiteratorで回す

iteratorとして使える変数: list, tuple, dict.keys()/values()/items()
numpy.ndarray()
ファイルハンドル (open()やcsv.reader()の戻り値)

iteratorを返す関数: range() (range型)
numpy.arrange() (numpy.ndarray型)

iteratorの使い方

- **for var in iterator 文で使う**
- **next(iterator): iteratorから値を取り出し、要素位置を一つ進める**

Pythonでの繰り返し構文 (ループ)

Fortranの繰り返し

```
do I = 1, 10
    print *, 'I=', I
end do
```

Cの繰り返し

```
for(i = 0 ; i < 10 ; i++) {
    printf("i=%d¥n", i)
}
```

python

```
for i in range(0, 10):
    print(f"i={i}")
```

Cの繰り返し

```
for(v = 0.0 ; v < 10.0 ; v += 2.0) {
    printf("v=%lf¥n", v)
}
```

python

```
for v in numpy.arrange(0.0, 10.0, 2.0):
    print(f"v={v}")
```

Python特有の文法: リスト内包表記
(01c_fitting_gpt_rev2.py)

data = [float(d) for d in data]

リスト内包表記

繰り返し構文 (ループ)

＋

配列 (list) 作成

を同時に行える

Python特有の文法: リスト内包表記とfor文 (01c_fitting_gpt_rev2.py)

```
data = [float(d) for d in data]
```

以下の繰り返し構文と同じ:

```
data = []
for d in data:
    data.append(float(d))  # 配列 (list) への追加は .append() を使う
```

長所: プログラムが短くなる。場合によっては可読性が上がる

forループよりも早い (ほとんどの場合無視できる違い)

短所: わかりにくい

個人の意見: 非常に単純なテンプレパターン以外では使わないことを推奨

リスト内包表記: ChatGPTで解決

Q: data = [float(d) for d in data] をfor文に展開してください

A:

```
data = []
for d in data:
    value = float(d)
    Tinv.append(value)
```

Q: 次の2重for文をリスト内包表記に変換してください

```
l = []
for i in range(10):
    for j in range(10):
        l.append([i, j, i*j])
```

A: `l = [[i, j, i*j] for i in range(10) for j in range(10)]`

テキストファイルの読み込み

(05-read_text_file.py)

・最も基本的なファイル処理:
　open()を使って受け取ったファイルハンドルオブジェクトで操作する

```
fp = open(input_path, 'r')          # open()関数でファイルを開き、ファイルハンドルオブジェクト fp を取得

data_list = []                      #  データを格納するためのリストを作成
labels = fp.readline().split()      # １行目のラベルを取得してリストに追加。. split()によりスペースで分割
for line in fp:                     # １行ずつ読み込んで line に返す
    data = line.split()             # line文字列を スペース で分割してリスト変数として返す
    data = [float(d) for d in data] # dataの要素 (文字列型) を実数型 (float) に変換してリストを作る
    data_list.append(data)          # 1次元リスト data_list の要素として、1次元リスト dataを追加し、2次元リストにする

T = data_list[0]                    # data_listを リスト型変数 T, N に代入する。
N = data_list[1]
```

テキストファイルの読み込み: with文

(05-read_text_file_with.py)

```
with open(input_path, 'r') as file:      # with ブロックで open() すると、withブロックから抜けるときに
                                         # 自動的に .close() してくれる
    data_list = []
    labels = file.readline().split()
    for line in file:
        data = line.split()              # line文字列を スペース で分割して返す
        data = [float(d) for d in data]  # dataの要素 (文字列型) を実数型 (float) に変換してリストを作る
        data_list.append(data)           # 1次元リスト data_list の要素として、1次元リスト dataを追加し、2次元リストにする

T = data_list[0]                         # data_listを リスト型変数 T, N に代入する。
N = data_list[1]
```

テキストファイルの読み込み: numpy.loadtxt()

(05-read_text_file_loadtxt.py)

- 2次元数値データを読み込むなら1番簡単
- 実数型の numpy.ndarray型 2次元配列を返す
- 数値データしか読み込めない

```
data_list = np.loadtxt(input_path, skiprows=1)  # 最初の 1行をスキップし、2行目以降を 2次元配列に読み込んで返す

T = data_list[0]                                # data_listを リスト型変数 T, N に代入する。
N = data_list[1]
```

出力文字列の整形: f-string (f文字列、フォーマット済み文字列リテラル)

- **print() には整形機能はない**
- **整形は、文字列変数自身が持っている機能**

  .format() 関数、C形式の書式指定 なども使えるが、

  f-string を使うべき

例: f“T: {T}“

- **文字列の前に ‘f’ を書く**
- **{}の中に変数名を書くと、変数の内容が展開される**

T = 297.14 の時、**f“T: {T}“**は

“T: 297.14”

と出力される

出力文字列の整形: f-string

変数の値を桁数を指定して出力

参考URL: https://note.nkmk.me/python-f-strings/

```
print(f“T: {T:.2f}”)          # 実数変数Tを有効数字2桁で表示
print(f“T: {N:.4e}”)          # 実数変数Nを e 形式 (1.0e2 など)、小数点以下4桁で表示
print(f“Tinv: {Tinv}”)        # 実数変数Tinvをフォーマット指定なしで表示
print(f“logN: {logN:.6g}”)    # 実数変数logNを有効数字6桁で表示

print(f“nAtom: {nAtom:6d}”)   # 整数変数 nAtom を 整数6桁で表示
print(f“name: {name:10}”)     # 文字列変数 name を 10桁で表示

print(f“name: {1000/T}”)      # {}内にはpython文を書ける
                              # ここでは 1000 / T を計算した結果を出力
```

pythonの特徴: ライブラリ

- 必要な機能は ライブラリを読み込んで利用する
  - 長所: 複雑なプログラムを自分で作る必要がない
    - “車輪の再発明はしない”
    - Cなどで書かれたライブラリが多く、高速
      - 典型例: numpyはC/Fortran互換の配列 ndarray() を使い、LAPACK、Intel MKLなどを直接呼び出す
  - 短所: いちいち **import文** でライブラリを読み込む必要がある
    - 欲しい機能がどのライブラリにあるかわからない、
    - どう書いたらいいかわからない

とにかく 検索 や ChatGPT で調べる

import文

- **プログラム中に import文 import lib_name を見つけたら、lib_name.py を探して実行**

  ライブラリの検索パスは
  - 環境変数 PYTHONPATH
  - sys.path (参考URL: https://note.nkmk.me/python-import-module-search-path/)

  で設定できる

- **lib_name.py は lib_name オブジェクトとしてアクセスできる**

  lib_nama.pyの関数 f()、グローバル変数 var は
  lib_nama.f(), lib_nama.var として使える

注意: ライブラリ名には ‘-’ は使えない

pythonの特徴: 基本機能は最小限
必要な機能はライブラリを使う

- **基本的な機能 (組み込み構文、組み込み関数) は最小限**

組み込み変数型	: int, float, str, list (その他 tuple, dict, set, map, zipなど)
オブジェクト定義	: class
定数	: None, True, False
変数型変換	: int(), float(), str()
制御構文	: if, for, while
比較構文	: ==, !=, <, <=, >, >=, is, not
演算子	: = (代入演算子), +, -, *, /, ** (べき乗),
演算子 (整数)	: % (剰余), // (整数同士を割り算した整数部分)
その他	: next(), isinstance(), setattr(), getattr()

注意: べき乗 a^b を a^b とする言語もある (MS-Excel) が、
pythonでは **a**b**
pythonでは、/ による計算では整数型変数も実数型変数に変換される
整数同士の割り算をしたい場合は、// を使う

CSVファイルの読み込み: csvライブラリ

(05-read_csv_file.py)

- **CSV (Comma Separated Values) ファイルの読み書き**
  データ中に , があったり、” ” でクオートされているデータも正常に読み込める

```
with open(input_path, ‘r’) as fp:        # input.csvファイルを開いてファイルハンドルを取得
    csv_reader = csv.reader(fp)          # CSVファイルを読み込むため、csv.reader() でreaderオブジェクトを取得
    labels = next(csv_reader)            # labelを読み込む
    data_list = []
    for row in csv_reader:               # 1行ずつ読み込み、list型配列をrowに入れる
        row = [float(d) for d in row]    # rowの要素 (文字列型) を実数型 (float) に変換してリストを作る
        data_list.append(row)
```

数学関数: math/numpy/scipy モジュールを使う

- Pythonの標準 (組み込み関数) には数学関数はない。
- math、numpy、scipy モジュールなどに含まれている関数を使う

01c_fitting_gpt_rev2.pyの例:

```
import numpy as np

logN = []
for n in N:
    logN.append(np.log(n))
```

別の書き方:

```
from numpy import log

logN = []
for n in N:
    logN.append(log(n))
```

mathモジュールを使う例:

```
from math import log

logN = []
for n in N:
    logN.append(log(n))
```

その他の数学関数: exp(), log(), sin(), cos(), tan(), sinh(), cosh(), tanh()
arcsin(), arccos(), arccot() など

ライブラリの使い方とオブジェクト

- **pythonはオブジェクト指向言語**

  オブジェクト: 変数に、関連する変数 (メンバ変数: attribute/property) と
  関数 (メンバ関数: attribute/method) を結び付けたもの

  class: オブジェクトの定義。class構文を使って定義する

  インスタンス: classを変数として実体化させたもの

例: 猫クラスの定義

```
class Cat():
   def __init__(self, x0, y0):
       self.x = x0
       self.y = y0

    def move(self, dx, dy):
       self.x += dx
       self.y += dy

    def where(self):
       print(f"current position: ({self.x}, {self.y})
```

例: 猫クラスを使用して、猫を移動して位置を表示する

```
mycat = Cat(3.0, 5.0)    # Catクラスのインスタンスを作る
mycat.move(0.5, -1.0)    # メンバ関数.move()の呼び出し
mycat.where()
```

出力結果:

```
current position: (3.5, 4.0)
```

オブジェクトの変数や関数を呼び出す方法:

. でつなげる

インスタンス変数名.attribute名

pythonのすべての変数、関数はオブジェクト

- **文字列変数 (ここではsとする) の代表的な メンバ関数**

```
s = s.strip()          # 文字列 s の前後の空白文字を削除して返す
s_list = s.split()     # 文字列 s を スペース ‘ ’ で分割した結果をリストで返す
```

- ファイルの操作: open()関数の**返り値**もオブジェクト

```
fp = open(‘input.txt’, ‘r’)          # ファイル input.txtをオープンし、
                                     # ファイルハンドルを返す
lines = fp.readlines()               # ファイルを、行ごとにリスト変数として返す
fp.close()                           # ファイルハンドルを閉じてファイル処理を終了
```

- ファイルの存在確認: **osモジュールの os.path モジュール**のisfile()関数を使う

```
import os
print(“Does file exist? “, os.path.isfile(‘input.txt’))
```

- **Pythonスクリプト** (ライブラリ) 自体がオブジェクト

mylib.pyというpythonスクリプトを import mylib で読み込む

=> mylib.py の関数 f(x) は mylib.f(x) で呼び出せる

よく使うpythonライブラリ

システム関係など

- **sys**
  sys.argvで起動時引数を取得
- **csv**
  CSVファイルの読み書き
- **openpyxl**
  Excelファイルの読み書き
- **pandas**
  Excelファイルの読み書き
  scikit-learnと組み合わせて使われる

グラフ関係

- **matplotlib**
  グラフの描画

科学計算関係

- **math**
  基本的な数学関数。sin, cos, tan, asin, log, exp　など
- **numpy**
  配列、行列を含む数値計算などにはほぼ標準 (高速)。
  numpy.ndarray からリストへは、ほとんど場合に自動的に型変換してくれる
  線形代数関数 (逆行列、一次連立方程式の解など)
- **scipy**
  numpyの機能に加え、信号処理 (FFT) など多様な数学関数がある
- **scikit-learn**
  機械学習関係

numpy.polyfit()関数呼び出し (01c_fitting_gpt_rev2.py)

多項式線形最小二乗法は、numpy モジュールの .polyfit() 関数で実行できる

```
# numpyモジュールのimport。numpyだと長いので、npという別名をつける
import numpy as np


# x軸の値リストに Tinv、y軸の値リストに logN を渡し、
# 次数 1 の多項式で最小二乗法を行う。
# フィッティング関数は y = b * x + a  <= 直観的な係数の順序と逆なので注意！
b, a = np.polyfit(Tinv, logN, 1)
```

グラフを描く: matplotlib (01c_fitting_gpt_rev2.py)

```
# matplotlib モジュールの pyplotクラスを plt という別名でimport
import matplotlib.pyplot as plt

# xデータに リスト変数 Tinv、yデータに logN を指定して散布図を描く
# 凡例に表示する文字列を label="Data"で与えている
plt.scatter(Tinv, logN, label="Data")
# xデータに リスト変数 Tinv、yデータに logNcal を指定して折れ線グラフを描く
# 線の色は red
# 凡例に表示する文字列を label="Fitted Curve"で与えている
plt.plot(Tinv, logNcal, color='red', label="Fitted Curve")
plt.xlabel("1/T")              # x軸の文字列を 1/T
plt.ylabel("log(N)")           # y軸の文字列を log(N)
plt.legend()                   # .scatter()、.plot()で与えた label を凡例として表示する
plt.show()                     # グラフを表示する
                               # グラフウィンドウを閉じるまで、プログラムは停止
```

条件分岐 (if … else if … else) と比較構文

	python	C	Fortran	Basic

If文	if 比較構文1:
	…
	elif 比較構文2:
	…
	else:
	…
		if(比較構文1) {
		…
		} else if(比較構文2) {
		…
		} else {
		…
		}
			if(条件構文1) then
			…
			else if(条件構文2) then
			…
			else
			…
			end if
				If 条件構文1 then
				…
				elseif 条件構文2 then
				…
				else
				…
				end if

数値の比較	==, !=, <, <=, >, >=, **a < b < c**	==, !=, <, <=, >, >=	== (.eq.), != (.ne.) < (.lt.), <= (.le.), > (.gt.), >= (.le.)	=, !=, <, <=, >, >=,
文字列の比較	==, !=, <, <=, >, >=, **a < b < c**	strcmp() などの関数を使う		=, !=, <, <=, >, >=,
定数、変数が同一であるか (値の比較ではなく、実体の比較)	is, is not	ポインタを比較する *a == *b, *a != *b		

数値データを横に並べて見やすく (01c_fitting_gpt_rev2.py)

```
# 文字列中の ¥t はTAB文字に置き換えて出力される
print("T¥tN¥t1/T¥tlog(N)")

# zip()を使うと、複数のリスト変数を1つずつ取り出してくれる
for t, n, t_inv, log_n in zip(T, N, Tinv, logN):
    print(f"{t:.2f}¥t{n:.4e} ¥t{t_inv:.4f}¥t{log_n:.6g}")
```

グローバル (大域) 変数とローカル (局所) 変数

参考URL: http://conf.msl.titech.ac.jp/Lecture/python/python-tips.html

- 関数外で代入するとグローバル変数　　: 同じスクリプト内で参照できる
- 関数内で代入するとローカル変数　　: 関数を抜けたら破棄される

大きなプログラムでは、大域変数は極力使わない方がいい

多くの大域変数を使う場合、class宣言をしてオブジェクトの attribute として変数を宣言する

【注意】関数内でグローバル変数と同じ名前の変数に代入すると、その名前のローカル変数が生成される。
関数内でグローバル変数へ代入する場合は、**global 宣言**を行う

```
a = 1

def f():
    a = 2          局所変数を生成、大域変数は変えない
    print("in func:", a)

print("outer func1:", a)
f()
print("outer func2:", a)
```

実行結果

```
outer func1: 1      大域変数
in func: 2          局所変数
outer func2: 1      大域変数
```

```
a = 1

def f():
    global a
    a = 2                  大域変数に代入
    print("in func:", a)

print("outer func1:", a)
f()
print("outer func2:", a)
```

実行結果

```
outer func1: 1      大域変数
in func: 2          大域変数
outer func2: 2      大域変数
```