

Python プログラム実行, Python 環境

(Python 入門)

URL: <https://www.kkaneko.jp/pro/pf/index.html>

金子邦彦



Python 実行環境の概要



- Pythonは複数の環境で実行可能である.
- 関連ツール：コマンドプロンプト, Visual Studio Code, PyCharm, Spyder, Jupyterノートブックなど（それぞれ特徴がある）.
- 機能：プログラム作成・編集・実行, シンタックスハイライト, 自動補完, 自動インデント, 変数探索など
- オンライン実行環境も選択可能：Google Colaboratory, Trinket, Python Tutor, Repl.itなどのオンライン実行環境も利用可能である.
- 作業の目的や内容に応じて適切な環境を選択することで, 作業効率が向上する.

実行環境の基本機能



- プログラム作成・編集・実行および結果表示
- シンタックスハイライト（可読性向上）
- 自動補完（プログラム作成支援）
- 自動インデント（プログラム作成支援）
- 変数探索, ファイル管理（閲覧, 編集, 作成）
- マニュアル表示による情報参照
- AI連携

これらの機能は, 効率的なプログラム開発を支援する.

コマンドプロンプトによるPython実行

WindowsでコマンドプロンプトからPythonを実行する場合

- Pythonのインストール (<https://www.python.org> から)
- **python**コマンドでPythonを起動すると、**プログラムを入力するたびに結果が得られる対話的実行が可能**
- 終了は**exit()**コマンド

```
C:\Users\user>python
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec 6 2022)
Type "help", "copyright", "credits" or "license()"
>>> x = 100
>>> if (x > 20):
...     print("big")
... else:
...     print("small")
...
big
>>> s = 0
>>> for i in [1, 2, 3, 4, 5]:
...     s = s + i
...
>>> print(s)
15
>>>
```

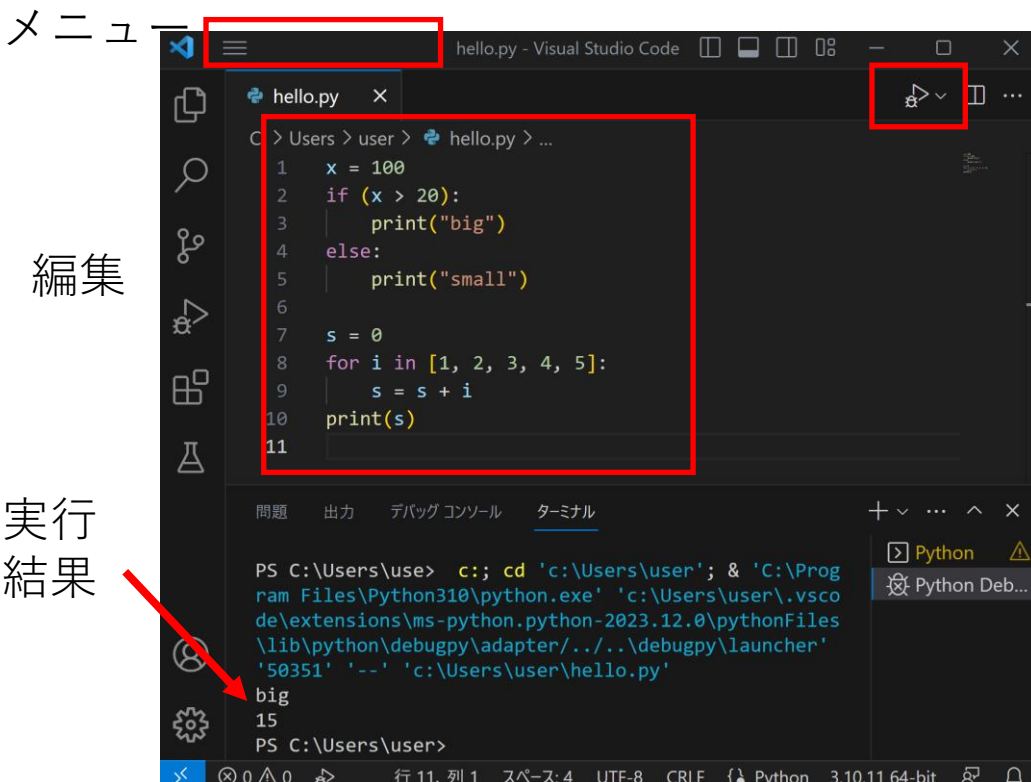
Python
プログラム
実行
結果

コマンドプロンプトで Pythonで開始

- Windowsでは、**python**コマンドで実行
- 終了は **exit()**

Visual Studio Codeによる開発

- Visual Studio Codeは、Microsoftが開発した多言語対応の統合開発環境
- Pythonのインストール (<https://www.python.org> から)
- Visual Studio Codeのインストール (<https://www.microsoft.com/ja-jp/dev/products/code-vs.aspx>) が必要
- 編集画面でプログラムを編集し、ターミナル（端末）で実行結果を確認できる。デバッグ機能により変数探索も可能である。



実行ボタン

※ 「デバッグ」の機能により変数探索も可能

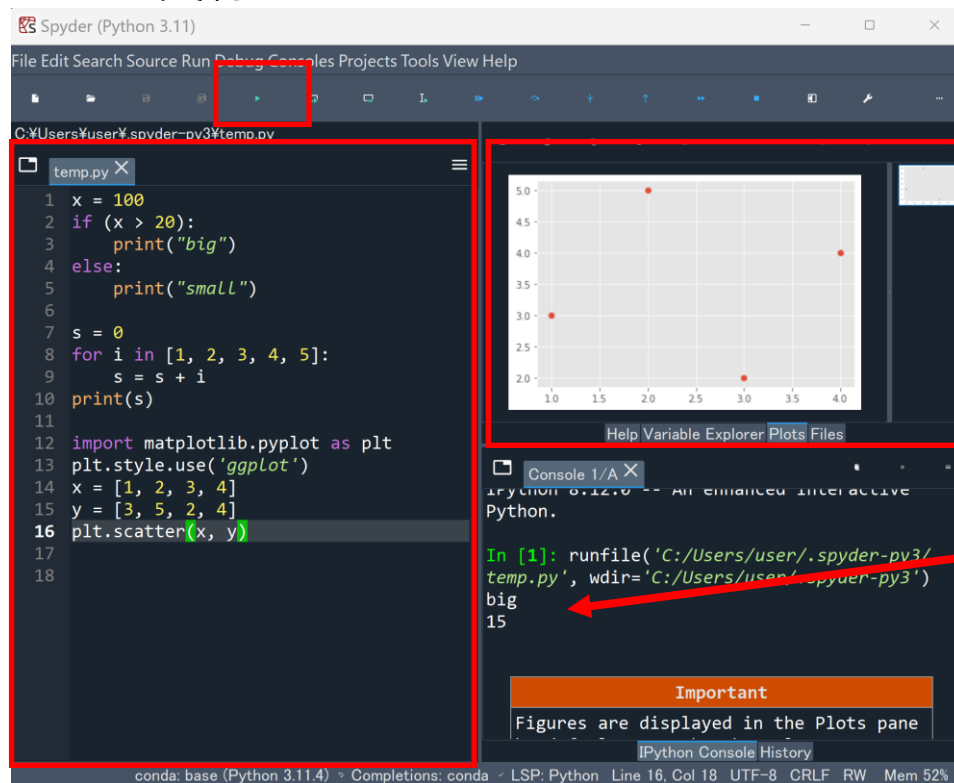
Spyderによる開発とデータ分析



Spyderは科学技術計算やデータ分析に特化した開発環境

- Pythonのインストール (<https://www.python.org> から)
- Anacondaに含まれている (<https://www.anaconda.com>)
- 編集画面でプログラムを編集し、コンソールで実行結果を確認できる。グラフ表示や画像の表示が容易である。

実行ボタン



編集

変数の値, 図やグラフの表示 (タブ切り替え)

実行結果

※ 画面右上の「Variable Explorer」タブにより変数探索も可能

Jupyter ノートブックによる開発とドキュメント化



Jupyter ノートブックは、Anaconda に含まれる対話型の開発環境。

- Python のインストール (<https://www.python.org> から)
- Anaconda に含まれている (<https://www.anaconda.com>)
- ノートブックの画面上で、コードセルやテキストセルを追加可能
- 実行ボタンで実行でき、グラフ表示や画像の表示が容易である。
- プログラム、実行結果、テキスト、画像を含む全体をノートブックとして保存できる。

実行ボタン

編集
実行結果
編集
実行結果
編集

実行結果

※ 変数探索は「%whos」

```
%whos
```

Variable	Type	Data/Info
i	int	5
plt	module	<module 'matplotlib.pyplot' from ...>
s	int	15
x	list	n=4
y	list	n=4

開発環境の共通機能



Python開発における**Visual Studio Code**, **Spyder**, および**Jupyterノートブック**. いずれも,

- プログラム作成・編集・実行および結果表示
 - シンタックスハイライト
 - 自動補完
 - 自動インデント
 - 変数探索
 - ファイル管理
 - マニュアル表示による情報参照など
- の機能を提供. これらの機能により効率的な開発が可能.

各開発環境の特徴



Visual Studio Code

- 豊富な言語対応

Spyder

- グラフ表示や画像表示
- 便利な変数探索（プログラム終了後も変数確認可能）

Jupyter ノートブック

- プログラム・実行結果・テキスト・画像・数式を含むノートブック形式
- 高い再現性（実行の一連の手順のノートブックに残り再現可能）

開発環境活用のメリット



- 1.多様な開発環境の理解**：適切な環境を選択することで作業効率が向上
- 2.データ分析と視覚化**：データ視覚化と分析が容易になる
- 3.デバッグ機能**：変数探索などのデバッグ機能は効率的なコード開発に役立つ
- 4.効率の向上**：自動補完や自動インデントなどの機能は作業効率とコーディングスキルの向上につながる
- 5.Jupyterノートブックの特色**：ノートブックである Jupyter ノートブック はドキュメンテーションと結果の再現性に役立つ

オンライン開発環境の概要



Colaboratory へようこそ
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ 変更を保存できませんでした

+ コード + テキスト ドライブにコピー

```
[6] x = 100
if (x > 20):
    print("big")
else:
    print("small")

big
```

```
[7] s = 0
for i in [1, 2, 3, 4, 5]:
    s = s + i
print(s)

15
```

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
x = [1, 2, 3, 4]
y = [3, 5, 3, 5]
plt.scatter(x, y)
plt.show()
```

MiniatureOldlaceHexagons
kunihikokaneke

main.py

```
1 x = 100
2 if (x > 20):
3     print("big")
4 else:
5     print("small")
6
7 s = 0
8 for i in [1, 2, 3, 4, 5]:
9     s = s + i
10    print(s)
11
12 import matplotlib.pyplot as plt
13 plt.style.use('ggplot')
14 x = [1, 2, 3, 4]
15 y = [3, 5, 3, 5]
16 plt.scatter(x, y)
17 plt.show()
```

Figure 1

Console

```
big
15
```

Google Colaboratory

<https://colab.research.google.com>

ノートブックの画面が開き，コードセルやテキストセルを追加可能。

Repl.it

<https://replit.com>

編集画面でプログラムを編集し，コンソールで実行結果を確認。グラフ表示や画像の表示が簡単にできる。多数の言語をサポート。

教育向けオンライン開発環境



Python Tutor: Visualize code in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

Python 3.6
[known limitations](#)

```
1 x = 100
2 if (x > 20):
3     print("big")
4 else:
5     print("small")
6
7 s = 0
8 for i in [1, 2, 3, 4, 5]:
9     s = s + i
10 print(s)
```

Print output (drag lower right corner to)

```
big
15
```

Frames Objects

Global frame	
x	100
s	15
i	5

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Done running (16 steps)

Ads keep this tool free; we are not responsible for contents of displayed ads
[Move and hide objects](#)

Python Tutor

<https://pythontutor.com>

編集画面でプログラムを編集し、
実行結果を確認。

Home / My Trinkets / [Untitled](#)

Save Cancel

main.py

```
1 x = 100
2 if (x > 20):
3     print("big")
4 else:
5     print("small")
6
7 s = 0
8 for i in [1, 2, 3, 4, 5]:
9     s = s + i
10 print(s)
```

Powered by **trinket**

```
big
15
```

Trinket

<https://trinket.io>

編集画面でプログラムを編集し、
実行結果を確認。に学習目的に適
している。

オンラインの開発環境のメリット



オンラインでプログラミングを実行する Google Colaboratory、trinket、Repl.it、Python Tutor など

- 1.アクセスの容易さ** インターネット接続と Web ブラウザを通じてすぐに利用可能で、開発環境のインストール不要.
- 2.プログラムの共有や公開の容易さ** 複数の利用者による共有やプログラムの公開が容易
- 3.ビジュアルでインタラクティブな実行:** インストール不要で、実行結果をリアルタイムでビジュアルに確認可能

プログラミングを学ぶときや、プログラムの開発プロジェクトでも有用. 多くの場合、**アカウントの作成やログインが必要. 有料の利用料金**が発生する場合もある. よく確認してから利用すること.

オンラインの開発環境の特徴比較



- Google Colaboratory
 - Jupyter ノートブックをオンラインで利用可能
 - グラフ表示や画像表示
 - AIに関連する多くのパッケージがインストール済み
 - Google Drive と統合
- Repl.it
 - グラフ表示や画像表示
 - 必要なパッケージの追加
 - リアルタイムの共同編集機能
- Python Tutor
 - 変数探索やステップ実行をビジュアルに可能で、主に学習目的である。
- Trinket
 - タートルグラフィックスをサポート
 - 作成したプログラムを他者が容易に実行可能。

ステップ実行と通常実行の概要



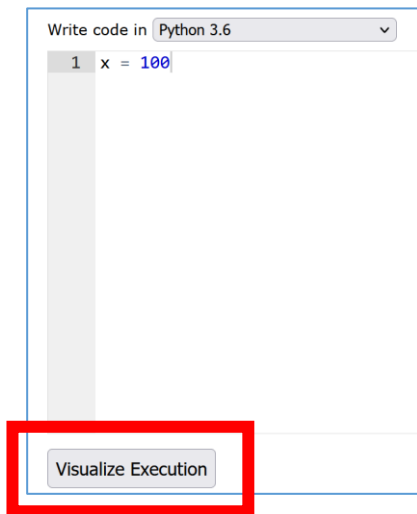
- **ステップ実行**では、**1行ずつの実行**が行われ、そのときの変数の値の変化などを**確認**できる。**プログラムの動作を細かく追跡**でき、不具合が発生している箇所の特定、プログラムの学習に役立つ
- **通常実行**は、**プログラムを最初から最後まで一度に実行**するもので、プログラム実行中の変数の値の変化を確認するなどは困難。

Python Tutor によるステップ実行の活用



- Python Tutor は Python などのプログラムを書き実行できるサイト。ステップ実行, 変数の値表示などの機能がある。
- Python Tutorのウェブサイト: <https://www.pythontutor.com/> で「Python」を選択
- メイン画面でプログラムを書き, Visualize Executionボタンをクリックすると実行される。通常実行はLastボタン, ステップ実行は他のボタンで行う。

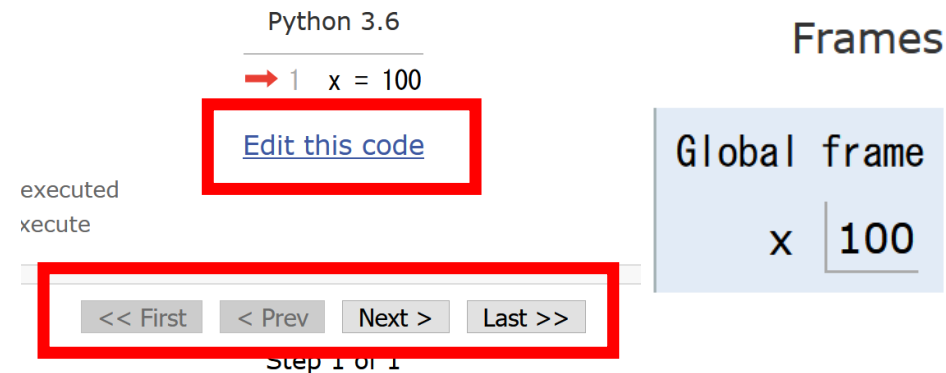
メイン画面で、プログラムを書く



Visualize Execution ボタン

メイン画面に
戻るには
Edit this code

変数の値を
視覚的に
確認できる



通常実行: Last

ステップ実行: 他のボタン