

Visual Studio Codeで始めるPythonプログラミング：

VS CodeからJupyter Notebookを使ってみよう

<http://www.atmarkit.co.jp/ait/articles/1806/12/news041.html> [この記事はPDF出力に対応していません]

Python人気を支えるツールの1つ「Jupyter Notebook」。VS Codeからこれを使ってみよう。Jupyter拡張機能が提供する機能も一覧する。

2018年06月12日 05時00分 更新

[かわさきしんじ, Insider.NET編集部]

インデックス

[連載目次](#)

[前回](#)はVisual Studio Code（以下、VS Code）でPythonコードをデバッグする上での基本を見た。今回はVS Codeから[Jupyter Notebook](#)を使ってみよう。なお、本稿ではWindows版VS Code（64ビット）のバージョン1.24およびJupyter拡張機能のバージョン1.1.3で動作を確認している（macOS版でもある程度の確認は行っている）。

VS CodeでJupyter Notebookを使うために必要なもの

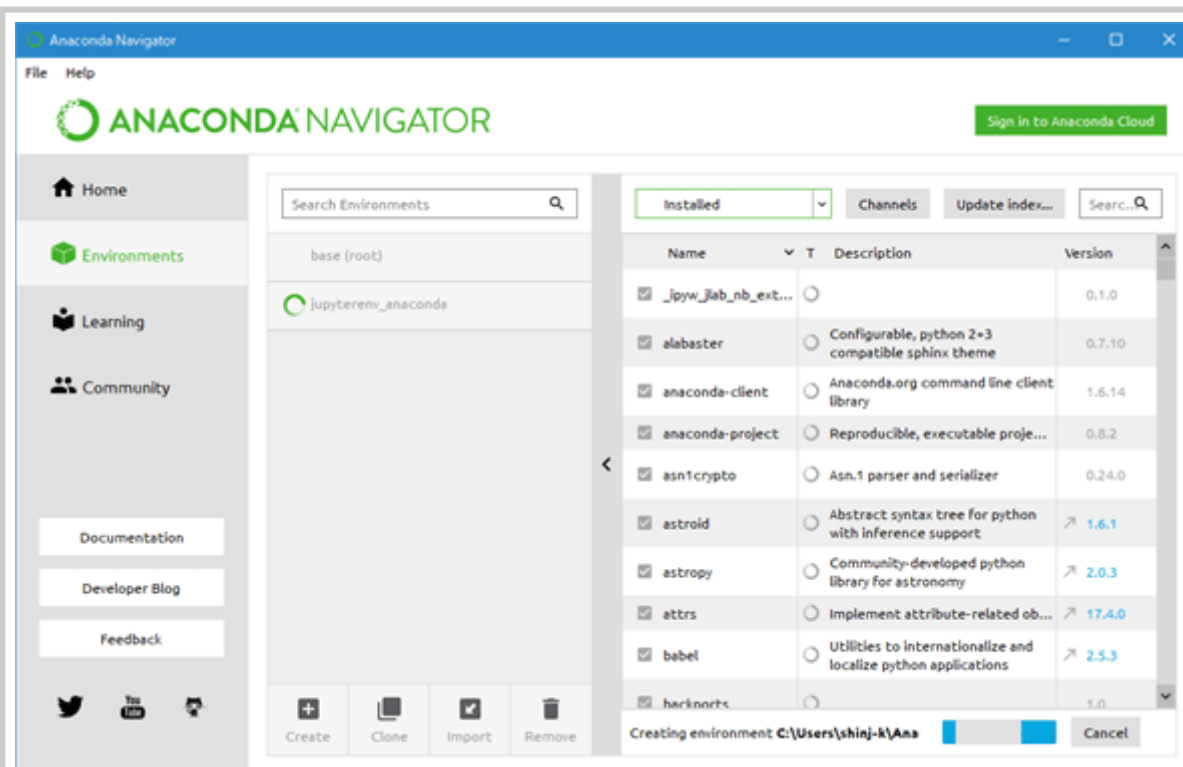
Jupyter Notebookはもともとは「実行可能なコードとそれに関連するテキストをひとまとめたドキュメント」をWebブラウザ上で作成したり共有したりすることを可能とするWebアプリだ。ドキュメントはセルと呼ばれる部分に分割され、そこにMarkdown形式でメモやコメントを記述したり、プログラムコードを記述したりしていくことで、ドキュメントとコードを1つの「ノートブック」にまとめて記述できるものだ。セルに記述したコードは実際に実行が可能であり、コードの編集と実行、そのドキュメントの編集をWebブラウザ上で効率よく行える（ことや、それを他者と簡単に共有でき、知見を広めるコストが大きく低下する）のが大きな特徴だ。機械学習やデータサイエンスの分野でよく使われているが、本稿ではそこまでは扱わずに、VS CodeでJupyterするための基礎について見ていこう。

VS CodeのPython拡張機能とJupyter拡張機能を利用すると、Jupyter Notebook（のサーバ機能）がVS Codeに統合される。これにより、Pythonモジュール（.pyファイル）の内容をセルに分割して、1つのセルだけを実行し、その結果を見ながら、コードに微修正を加えて再実行してみるといったことが手軽に行える（コードとドキュメントが一体化した「ノートブック」というよりは、コード記述と実行を簡便に行えるようになるというのが、Jupyter拡張機能がもたらす大きなメリットと筆者は感じている）。

VS CodeでJupyterするには以下が必要になる。

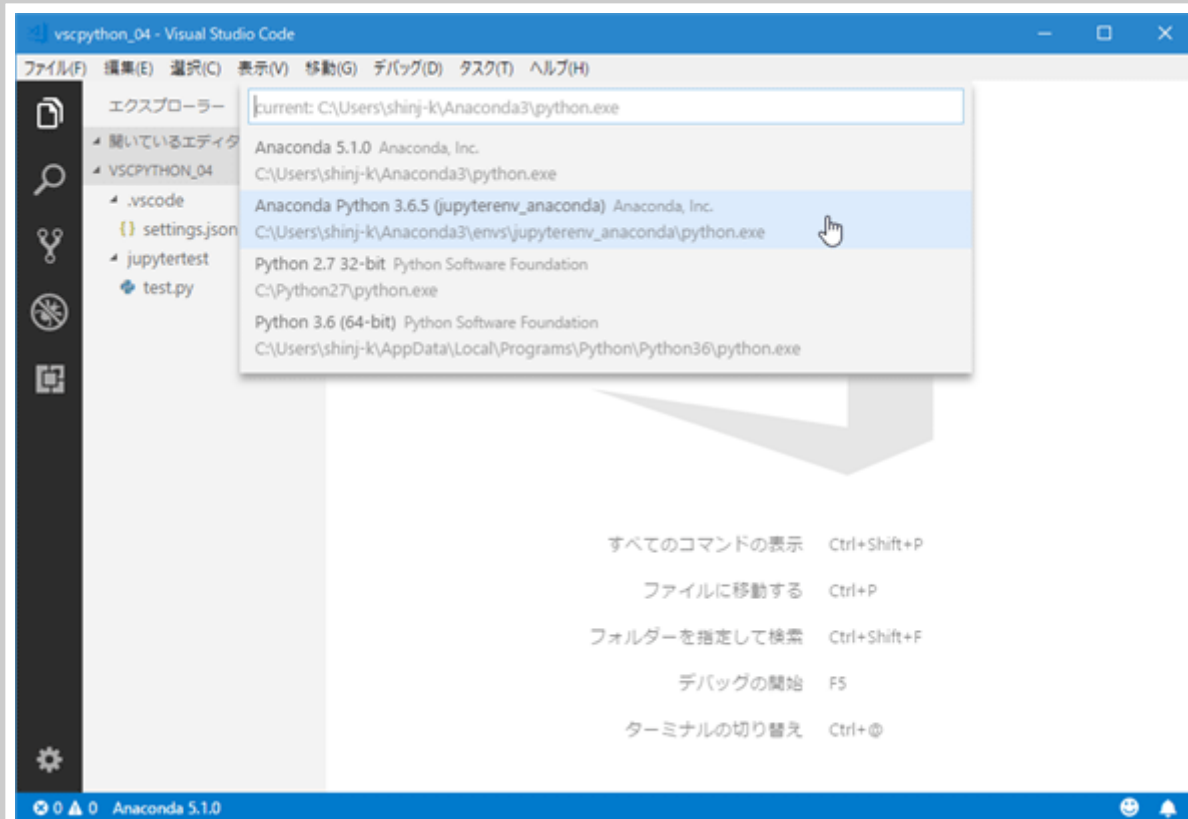
- Python処理系
- Jupyter（と、コードの実行に必要なその他のパッケージ）
- Python拡張機能
- Jupyter拡張機能

Jupyterの公式サイトでは、「[Anacondaディストリビューションを利用してPythonとJupyterをインストールすることを強く推奨](#)」している。そこで今回は、筆者の環境に既にインストール済みのAnacondaのバージョン5.1.0をベースに作成したPython環境を例としよう（最新バージョンは5.2）。また、Python拡張機能は既にインストール済みであるものとする。ここでは、手抜きをして、Anaconda Navigatorから「jupyterenv_anaconda」という環境を作成している。



「jupyterenv_anaconda」環境を作成しているところ

その後、VS Codeでコマンドパレットから「Python: インタープリターを選択」コマンドを実行して、作成した環境をPython環境として指定してやる。



作成したAnacondaベースの環境をPython環境として指定

そうではなく、Anaconda以外のPython処理系（をベースとした仮想環境など）にJupyterをインストールするのであれば、Jupyterの公式サイトにもあるようにpipコマンドでJupyterおよび必要となるパッケージをインストールする。例えば、以下はmacOS版のVS Codeでvenvパッケージを使用して「jupyterenv_python365」という仮想環境を作成した上で、そこにJupyterをインストールしているところだ（macOSではこの環境で動作を確認している）。

```

問題 出力 デバッグ コンソール ターミナル 1: bash
:~$ python3 -m venv jupyterenv_py365
:~$ pip3 install jupyter
Requirement already satisfied: jupyter in /usr/local/lib/python3.6/site-packages (1.0.0)
Requirement already satisfied: notebook in /usr/local/lib/python3.6/site-packages (from jupyter) (5.2.1)
Requirement already satisfied: qtconsole in /usr/local/lib/python3.6/site-packages (from jupyter) (4.3.1)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.6/site-packages (from jupyter) (5.3.1)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.6/site-packages (from jupyter) (4.6.1)
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.6/site-packages (from jupyter) (7.0.5)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.6/site-packages (from jupyter) (5.2.0)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.6/site-packages (from notebook->jupyter) (4.4.0)
Requirement already satisfied: traitlets>=4.2.1 in /usr/local/lib/python3.6/site-packages (from notebook->jupyter) (4.3.2)

```

macOS版のVS Codeで「jupyterenv_py365」仮想環境を作成し、そこにJupyterをインストールしたところ（統合ターミナル）

こちらの場合も、コマンドパレットで「Python: インタープリターを選択」コマンドから、作成した仮想環境をPython環境に指定する。

Jupyter拡張機能はサイドバーの「拡張機能」ビューで「jupyter」を検索して、最初に出てくるDon Jayamanne氏が作成したものをインストールすればオーケーだ。

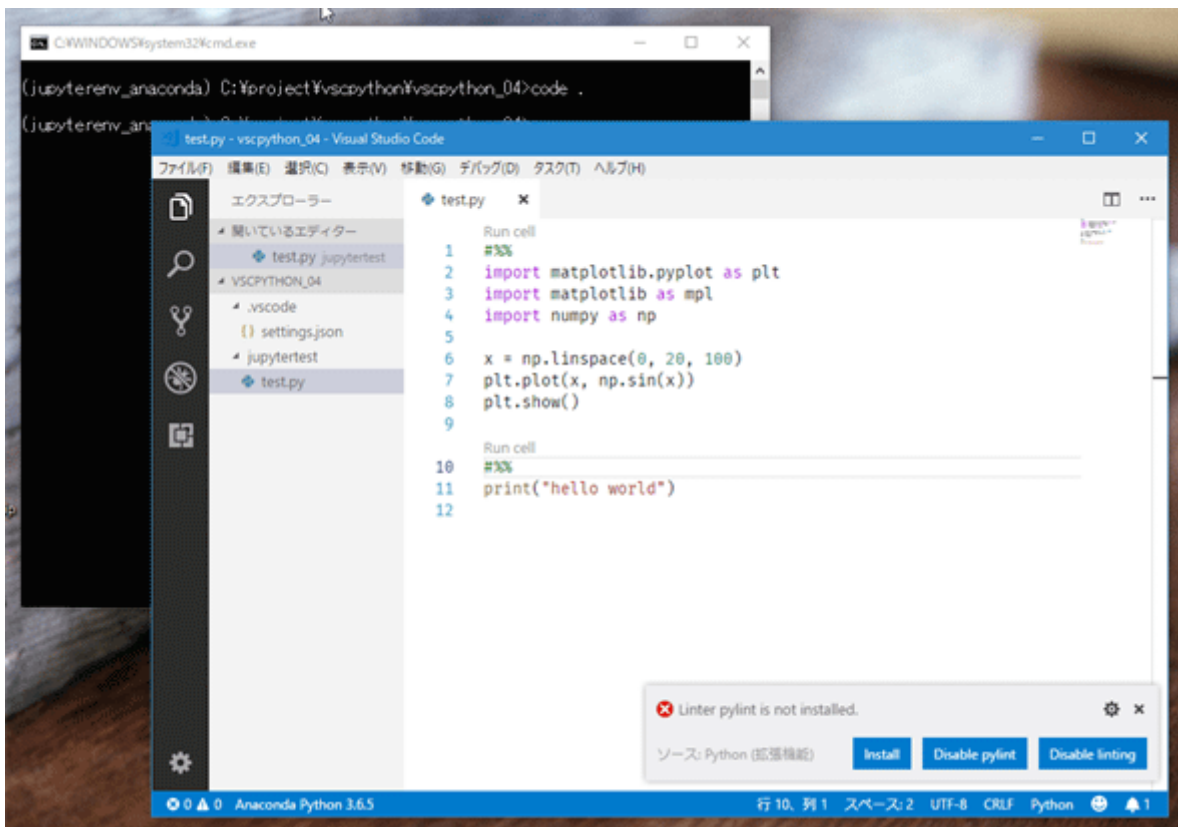


Jupyter拡張機能（画面は既にインストールが終わっている状態）

Jupyterを利用する環境のセットアップとJupyter拡張機能のインストール、必要となるその他のパッケージやモジュールのインストールが終われば、VS CodeでJupyterを利用する準備は完了だ（例えば、以下の例ではnumpyパッケージおよびmatplotlibパッケージが必要になる）。

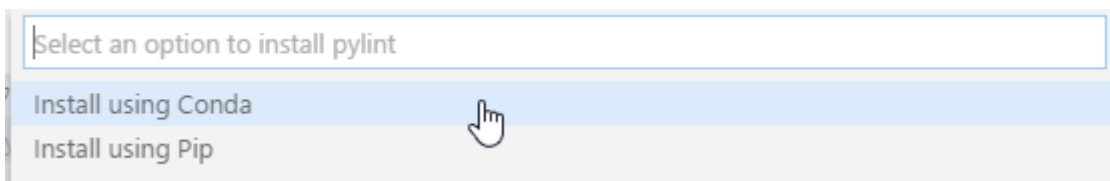
AnacondaベースのPython環境

余談となるが、Python環境としてAnacondaベースのものを使用する場合には、その環境を有効化したコマンドプロンプトやシェルからVS Codeを起動しておくのが便利かもしれない。例えば、以下は前述した「jupyterenv_anaconda」環境を有効化したAnacondaプロンプトからVS Codeを起動しているところだ。



Anacondaベースの環境を有効化した状態でVS Codeを起動

VS Codeのウィンドウ右下にはおなじみの「pylintがない」エラーが表示されている。ここで [Install] ボタンをクリックすると、Anaconda環境が有効化されているときには次のようにcondaコマンドとpipコマンドのどちらを使用するかのプロンプトが表示される。Anaconda環境が有効でないときには、python.pythonPath項目で指定されているpythonコマンドに対してpipパッケージを指定して、パッケージのインストールが行われるようだ（ただし、検証したのはWindows版のみなので、他の環境では異なる振る舞いとなるかもしれない）。



パッケージのインストールにcondaコマンドとpipコマンドのどちらを使用するかを指定

この他にもコマンドパレットの [Python: Create Terminal] コマンドを実行したときの挙動に差がある（筆者が試したところでは、単にターミナルを開くだけでPython環境が有効化されていないような挙動となった）ので、AnacondaベースのPython環境を使うときには、それを有効化した状況（コマンドプロンプトなど）からVS Codeを起動するのがオススメだ。

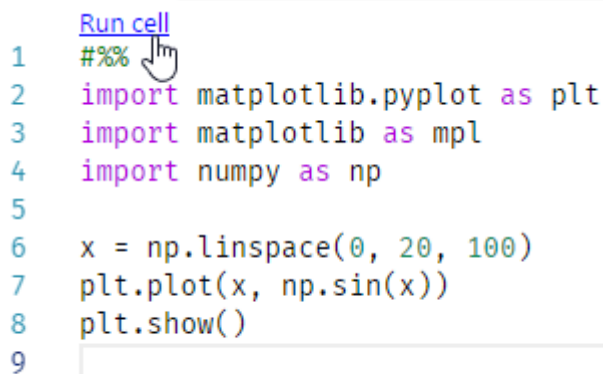
サンプルコードを実行してみよう

準備が整ったら、[Jupyter拡張機能の説明ページ](#)にあるサンプルコードを実行してみよう。ここではjupytertestディレクトリを作り、その下にtest.pyファイルを作成して、そこにこのコードを記述している。

```
#%%  
import matplotlib.pyplot as plt  
import matplotlib as mpl  
import numpy as np  
  
x = np.linspace(0, 20, 100)  
plt.plot(x, np.sin(x))  
plt.show()
```

サンプルコード

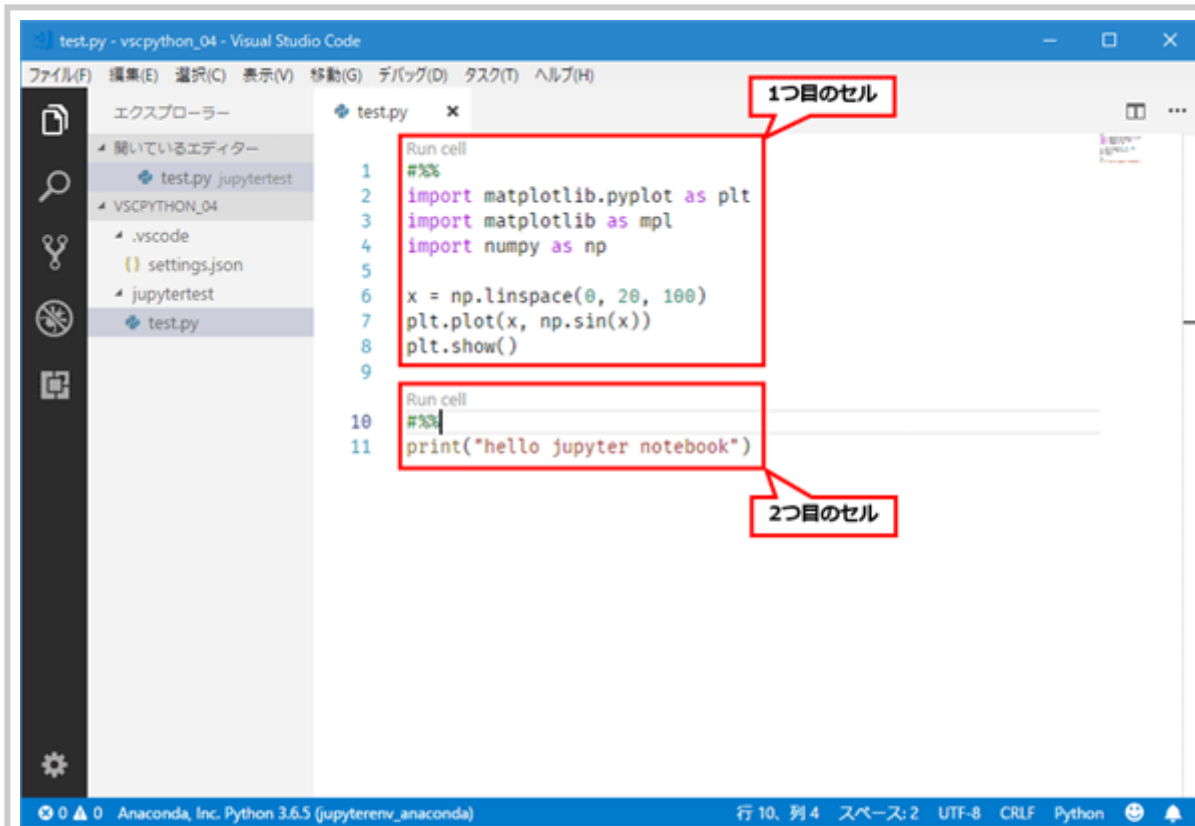
先頭行の「#%%」は、Jupyter Notebookにおける「セル」の区切りを示すもので、これで区切られた領域が1つの実行単位となる。そのため、この行の上には「Run cell」というリンクが表示されている。



```
1  Run cell  
1  #%%  
2  import matplotlib.pyplot as plt  
3  import matplotlib as mpl  
4  import numpy as np  
5  
6  x = np.linspace(0, 20, 100)  
7  plt.plot(x, np.sin(x))  
8  plt.show()  
9
```

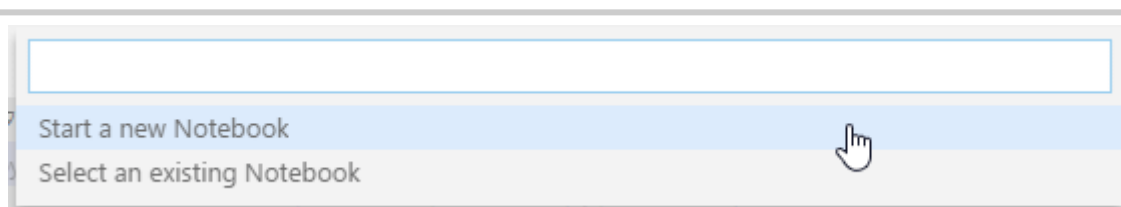
【Run cell】リンクをクリックすると、そのセルの内容が実行される

また、以下のように「#%%」で区切ってprint関数呼び出しを記述すると、これは2つのセルを含んだコードとなる。



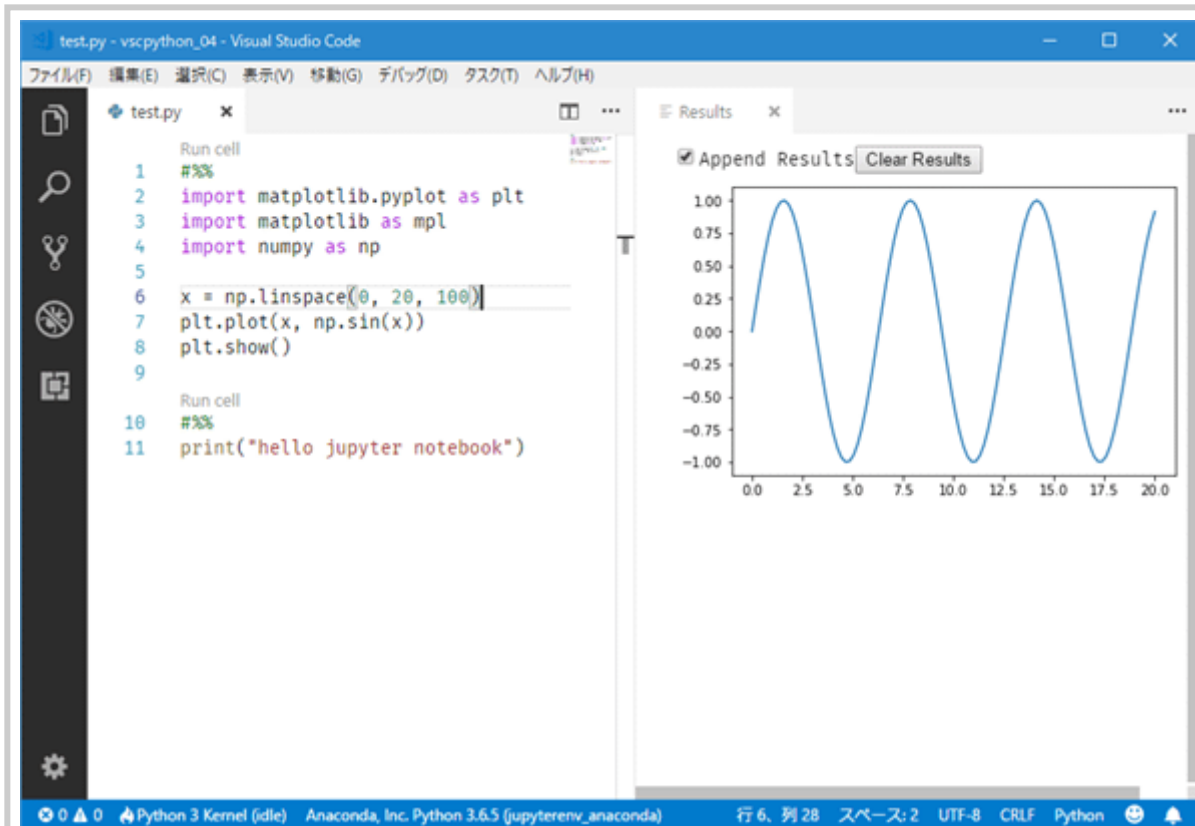
2つのセルで構成されるコード

では、1つ目の「Run cell」リンクをクリックしてみよう。すると、次のようにJupyter Notebookを新規に起動するか、既に起動されているJupyter Notebookを指定するかを選択画面が表示される。



「Start a new notebook」を選んで新規にJupyter Notebookを起動する

ここでは「Start a new notebook」を選択する。すると、必要なパッケージがインストールされていれば、先頭のセルの内容が実行されて、次のように別のエディタに実行結果が表示される。されない場合には、condaコマンドあるいはpipコマンドで足りないものをインストールしよう。

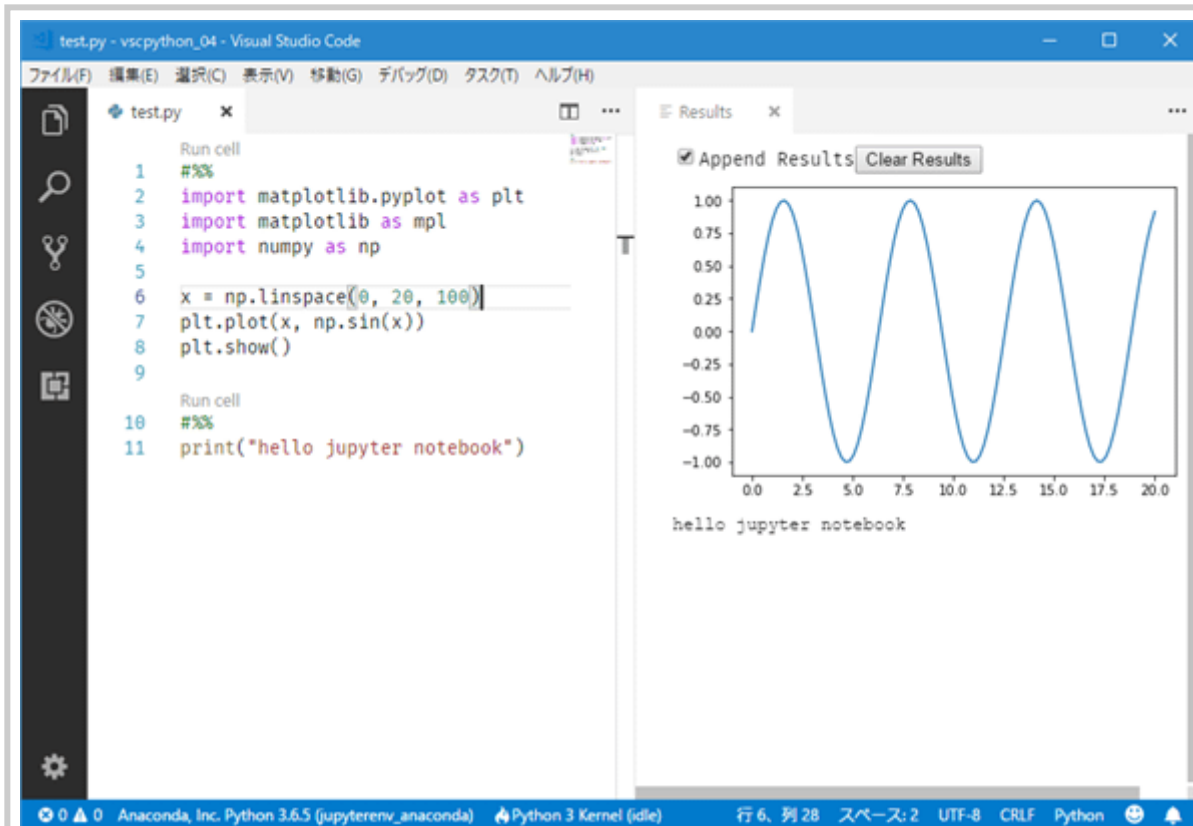


実行結果の表示

このときに「Failed to Detect Jupyter Notebook」のようなメッセージが表示されたら、コマンドパレットから [Jupyter: Select an existing (local) Jupyter Notebook] コマンドを選択して、起動しているJupyter Notebookの中のいずれかを選択してから実行するとうまく実行できるかもしれない（Jupyter Notebookのインスタンスの選択については後述）。

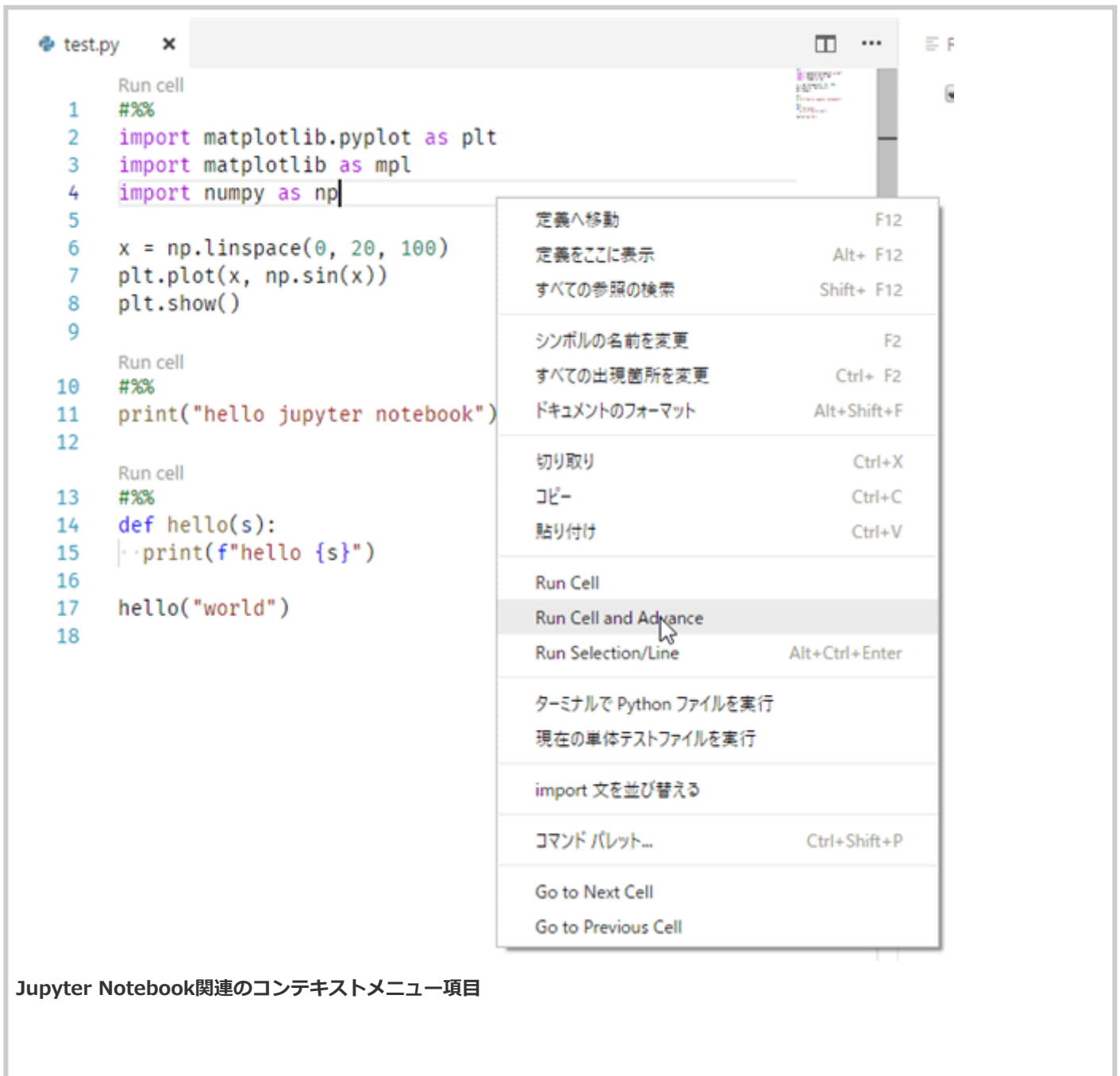
ここで覚えておいてほしいのが、ステータスバーに表示されている [Python 3 Kernel (idle)] という表示だ。Kernel（カーネル）とは、ある言語のコードを実行するためのエンジンであり、上の画像ではVS Codeから起動されたJupyter Notebookのインスタンス上でPython 3のカーネルが実行されている（が、アイドル状態にある）ことが分かる。

2つ目の [Run cell] リンクをクリックすれば、その結果がグラフの下に表示される。



print関数呼び出しの実行結果がグラフの下に表示されたところ

Pythonコードを表示しているエディタを右クリックすると、さらに細かく実行を制御できる。



上に示したコンテキストメニューには、Jupyter拡張機能が提供するメニュー項目として以下のものが表示されている。

- [Run Cell]：右クリックした位置にあるセルの内容を実行する
- [Run Cell and Advance]：右クリックした位置にあるセルの内容を実行して、次のセルにカーソルを移動する
- [Run Selection/Line]：選択範囲あるいは現在行だけを実行する
- [Go to Next Cell]：次のセルに移動する
- [Go to Previous Cell]：前のセルに移動する

連続するセルの内容を続けて実行したいときには、[Run Cell and Advance] が便利に使えるはずだ。また、セル内で特定の範囲だけをテストしたいというときには、[Run Selection/Line] が有用だ。[Go to Next Cell] と [Go to Previous Cell] はセル単位でカーソルを前後に移動する。

例えば、以下のようにhello関数を定義して、それを呼び出すセルを書いたとする。そして、このセル全体を実行しないまま、hello関数呼び出しを実行してみる（これは使い方としては間違っている例だ）。



```

8 plt.show()
9
10 Run cell
11 #%%
12 print("hello jupyter")
13
14 Run cell
15 #%%
16 def hello(s):
17     print(f"hello {s}")
18
19 hello("world")

```

現在行だけを実行

すると、次のようにエラーが発生する。

```

-----
NameError                                Traceback (most recent call last)
<ipython-input-8-82ecdd467ddd> in <module>()
----> 1 hello("world")

NameError: name 'hello' is not defined

```

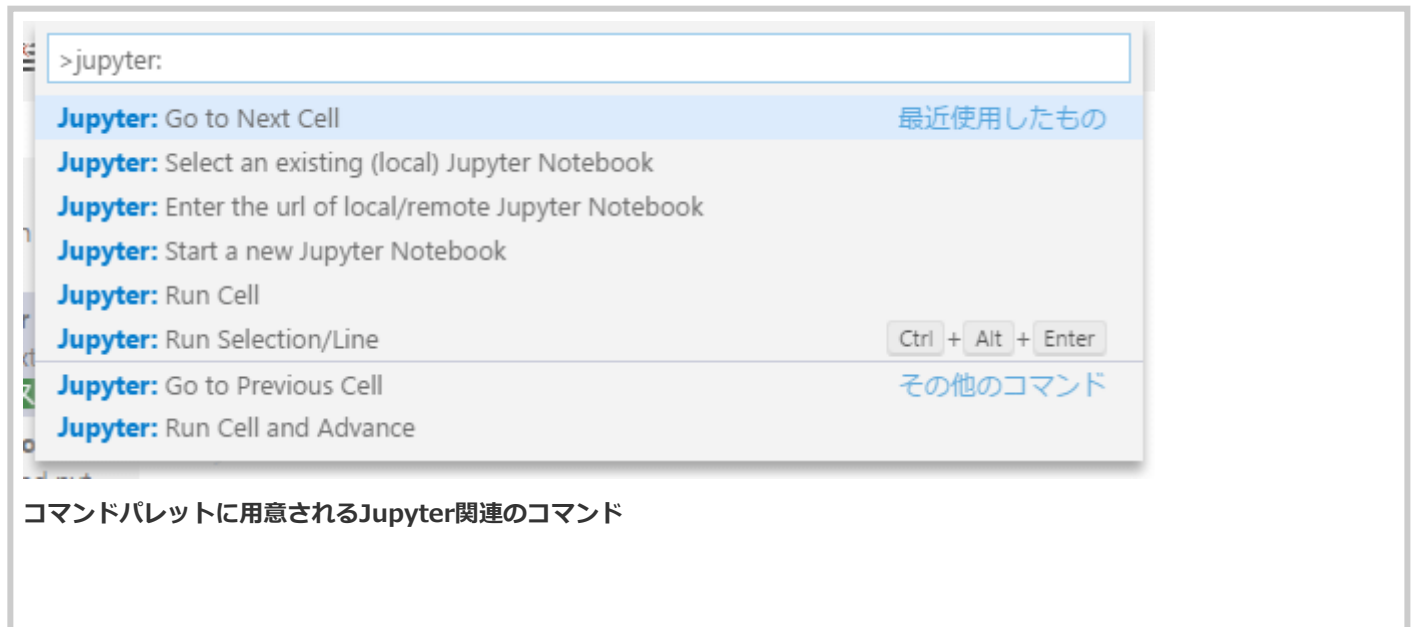
hello関数を定義していないのでNameErrorが発生する

例として間違った使い方を見てみたが、実際には関数定義とその呼び出しを別のセルに分けておき、関数定義を修正してからそのセルを実行、さらに関数呼び出しのセルを続けて実行のよう

な使い方をすることで、細かなコードの修正とそのテストをインタラクティブな環境で簡単に行える。

コマンドパレットにあるJupyter関連のコマンド

コマンドパレットにも上で見たコンテキストメニューの項目に相当するコマンドや追加のコマンドがある。

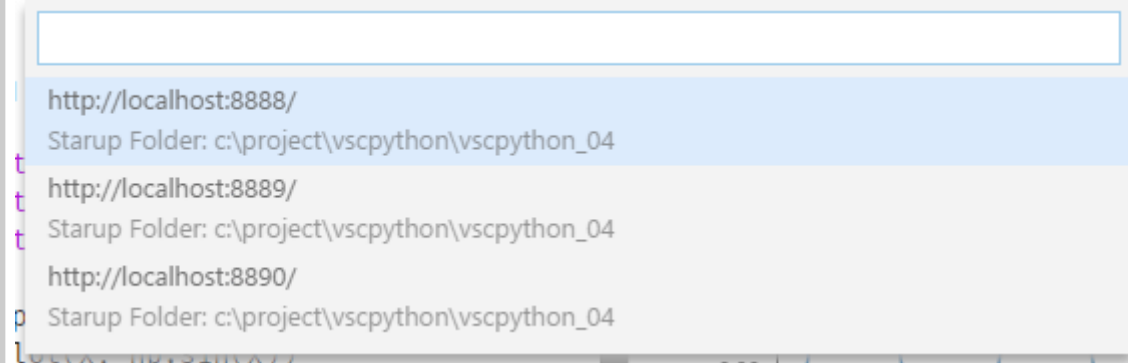


セルの実行や移動に関連するコマンドは既に説明した通りなので、他のコマンドについて触れておこう。

「Jupyter: Select an existing (local) Jupyter Notebook」コマンドは既にJupyter Notebookを起動している場合に、どれをセルの実行に利用するかを選択するのに使用する。

「Jupyter: Enter the url of local/remote Jupyter Notebook」コマンドはローカル／リモートで動作しているJupyter NotebookサーバのURLを入力して、サーバに接続するためのコマンドとなる。「Jupyter: Start a new Jupyter Notebook」コマンドは新規にJupyter Notebookを起動する。

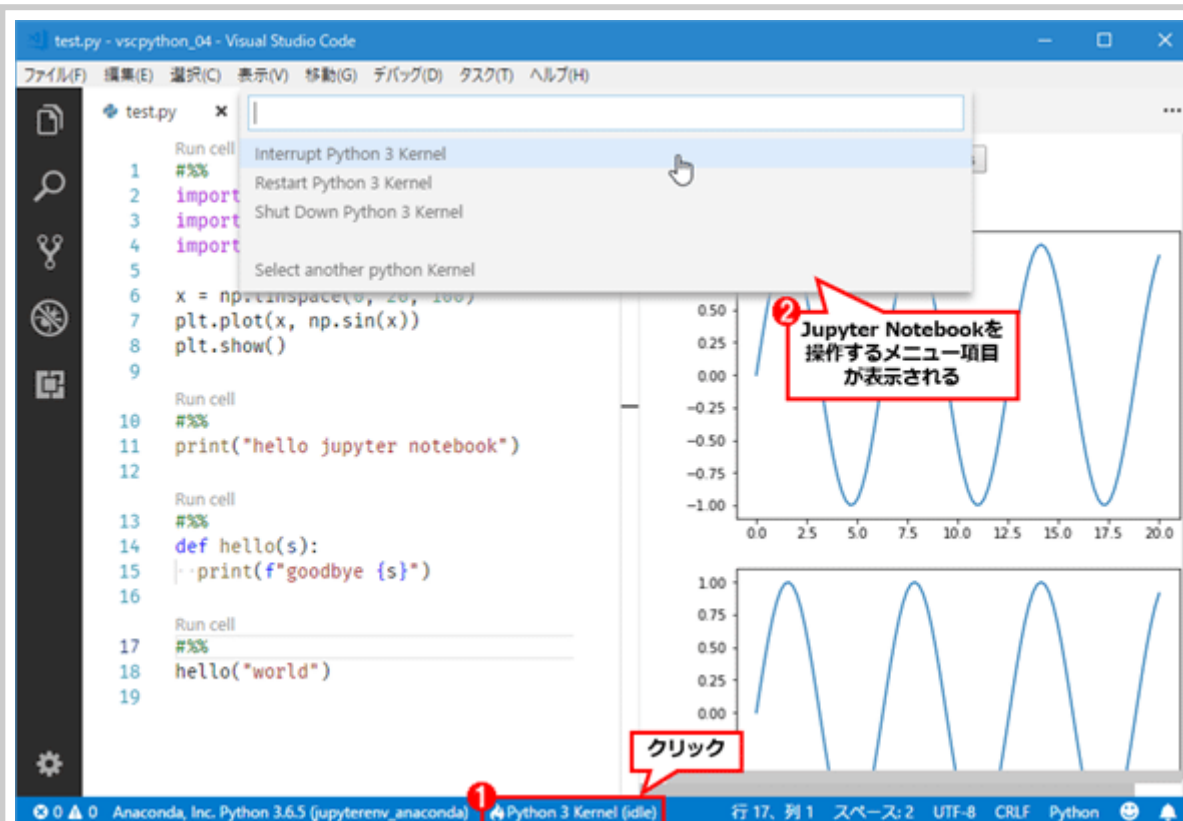
先ほども述べたが、「Run Cell」リンクをクリックすると、「Failed to Detect Jupyter Notebook」といったエラーメッセージが表示されることがある。また、気が付かないうちにJupyter Notebookを幾つも起動していると、同じエラーが発生することもある。こうしたときには、「Jupyter: Select an existing (local) Jupyter Notebook」コマンドを実行して、どれを使用するかを指定してやるとよい。このコマンドを実行したところを以下に示す。



Jupyter Notebookインスタンスの選択

上の画像では、Jupyter Notebookが3つ表示されている（これは筆者がいろいろとテストしていたからだろう）。ここで、セルの実行に使用するものを選択できる。セルを実行しようとして「Failed to Detect Jupyter Notebook」といったメッセージが表示された場合は、このコマンドを実行して、どれを使用するかを指定するとよい（あるいは、1つも起動していないときには[Jupyter: Start a new Jupyter Notebook] コマンドを実行してから、[Jupyter: Select an existing (local) Jupyter Notebook] コマンドを実行する）。

また、ステータスバーに表示されている[Python 3 Kernel (……)]という部分をクリックすると、そのJupyter Notebookのカーネルを操作可能だ。

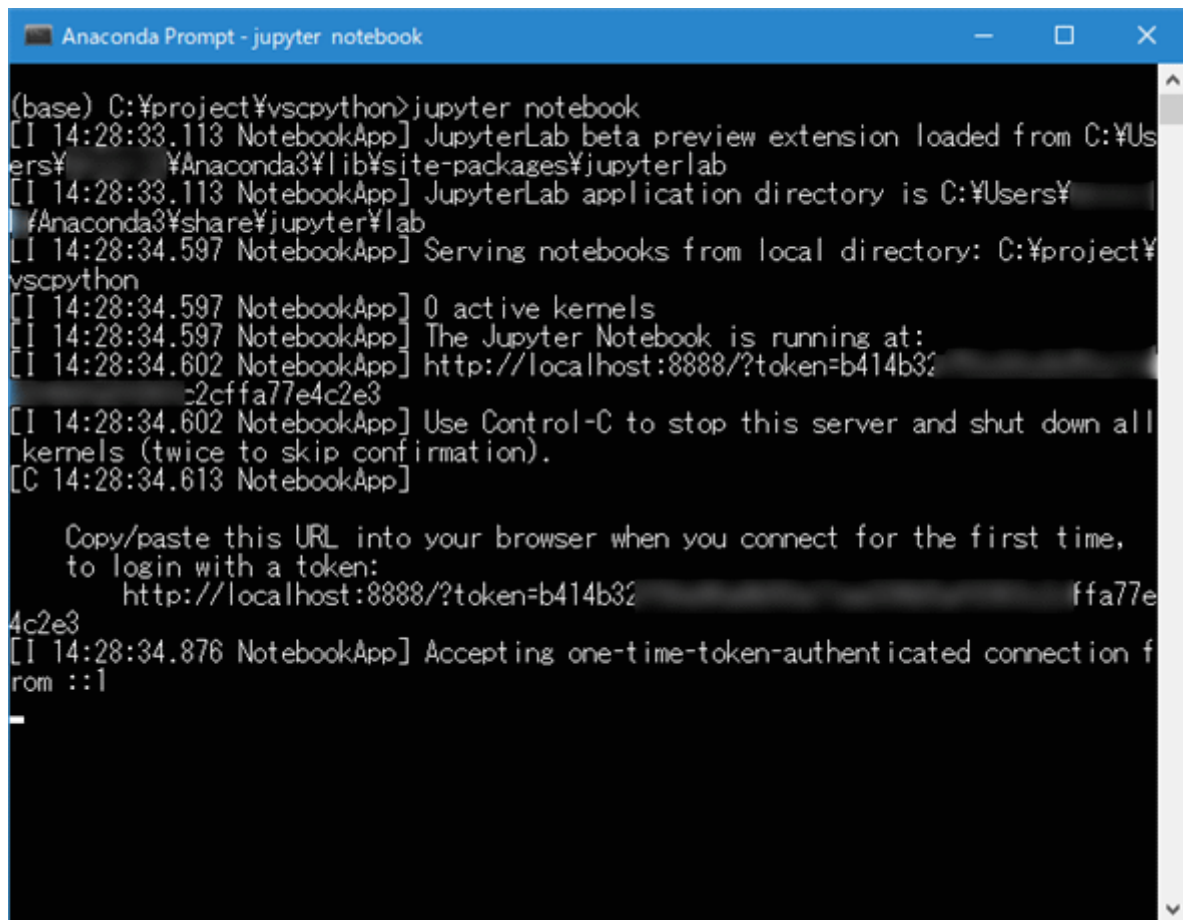


Jupyter Notebookの操作メニュー

ここではPython 3を実行するカーネルの中断、リスタート、シャットダウンなどを指定できる。[Select another python Kernel] は別のバージョンのカーネルを使用したい場合に利用すると思われる。

ローカルサーバに接続

最後にWebブラウザで実行中のJupyter Notebookアプリに接続する方法についても見ておこう。ここでは例として、本稿で作成したAnacondaベースの環境を使わずにAnacondaの(base)環境で「jupyter notebook」コマンドを実行している（実際に実行されるのは、ここではAnacondaに付属のjupyter-notebook.exeコマンドとなっている）。

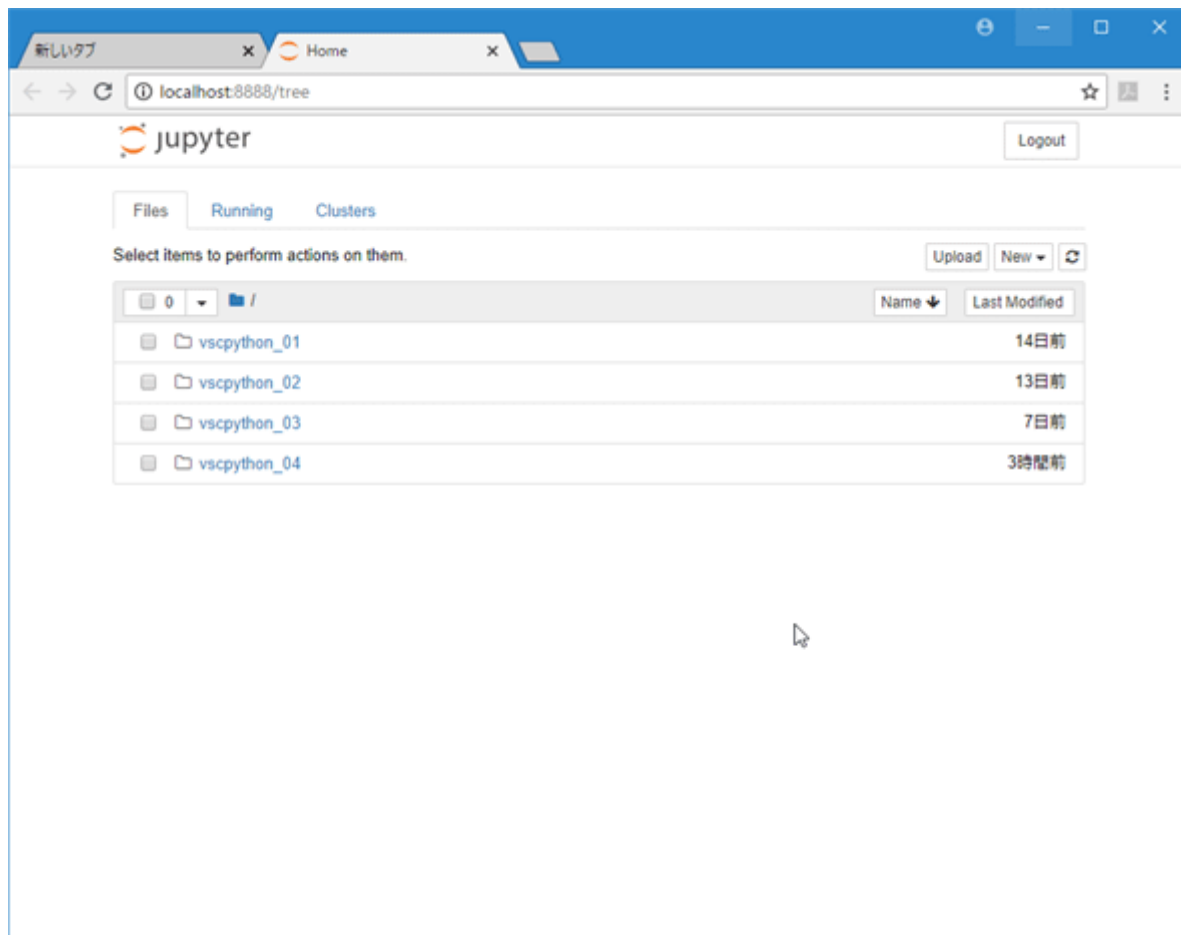


```
Anaconda Prompt - jupyter notebook
(base) C:\project\vscodepython>jupyter notebook
[I 14:28:33.113 NotebookApp] JupyterLab beta preview extension loaded from C:\Users\
ers\Anaconda3\lib\site-packages\jupyterlab
[I 14:28:33.113 NotebookApp] JupyterLab application directory is C:\Users\
\Anaconda3\share\jupyter\lab
[I 14:28:34.597 NotebookApp] Serving notebooks from local directory: C:\project\
vscodepython
[I 14:28:34.597 NotebookApp] 0 active kernels
[I 14:28:34.597 NotebookApp] The Jupyter Notebook is running at:
[I 14:28:34.602 NotebookApp] http://localhost:8888/?token=b414b32c2cffa77e4c2e3
[I 14:28:34.602 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[C 14:28:34.613 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=b414b32c2cffa77e4c2e3
[I 14:28:34.876 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
```

Anacondaプロンプトで「jupyter notebook」コマンドを実行

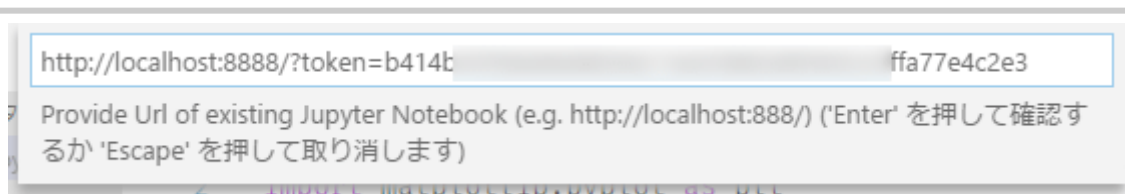




WebアプリとしてのJupyter Notebookが起動される

Jupyter Notebook Webアプリを起動

ここではWebアプリとしてのJupyter Notebookを使用するのではなく、これをバックエンドとしてVS Codeから接続して実際のコード実行を任せてしまうのが目的だ。そのときに重要なのが、Jupyter Notebookアプリの起動時に表示されているURLだ。これをコピーしておく。そして、コマンドパレットから [Jupyter: Enter the url of local/remote Jupyter Notebook] コマンドを実行する。これにより、次のようなURL入力プロンプトが表示されるので、コピーしておいたURLを貼り付ける。



Jupyter Notebookアプリ（Webサーバ）のURLを入力

なお、Jupyter Notebookアプリを起動している場合、 [Jupyter: Select an existing (local) Jupyter Notebook] コマンドでも選択肢にそれが表示されるので、そちらを使ってもよいだろう（見えない場合には、上記のように自分でURLを指定しよう）。

この状態で、[Run Cell] リンクをクリックすると、Webアプリの方のJupyter Notebookでセルの内容が実行され、その結果がVS Codeに表示されるようになる。既にJupyter Notebookを運用しているのであれば、それをバックエンドとして使えるので便利だ。



本稿ではVS Code用のJupyter拡張機能が提供する機能と、それを使ってJupyter NotebookのセルをVS Codeで実行する方法を見た。Pythonコードを記述する際にうまく活用できれば、VS Codeを利用したPython開発がより快適になるはずだ。

[インデックス](#)[「Visual Studio Codeで始めるPythonプログラミング」](#)

Copyright© Digital Advantage Corp. All Rights Reserved.

