第1章 対話的に実行できてビギナにも便利

おすすめの動くPython電子ノートJupyter Notebookをはじめる

佐藤 聖

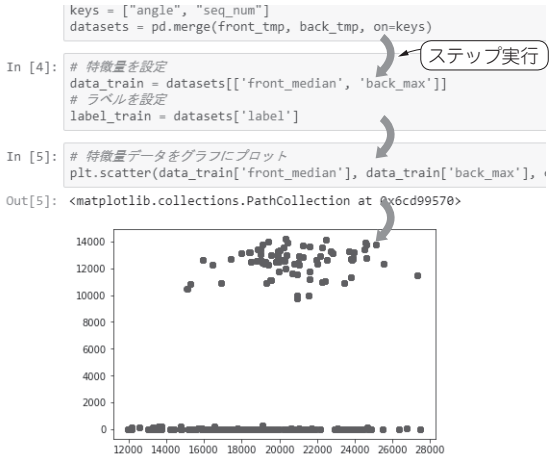

図1 ステップ実行＆結果表示が可能なJupyter Notebook
開発の小回りがきくからおすすめ

● 覚えることが少ないわりに機能が豊富

Jupyter Notebook（ジュパイターまたはジュピター・ノートブック）は，ウェブ・ブラウザからログインすれば，すぐに利用できます．作成したプログラムの実行や実行結果を記録できるウェブ・ツールです（**図1**）．

Pythonカーネルにはインタラクティブな Python実行環境のIPythonを利用しています．Jupyter NotebookはAnaconda（Pythonのデータ・サイエンス向けの科学技術演算用ディストリビューション）にも同梱されています．機械学習のセミナやワークショップなどでもよく利用されており，初心者でも使いやすく，仕事で利用しようとした場合にも豊富な機能が利用できるため，開発者にとっては必須のツールです．

もしエディタでPythonコードを書いているのであれば，Jupyter Notebookに変えるとプログラム開発がかなり快適になるはずです．もちろん統合開発ツールを使うともっと高機能なのですが，豊富な機能を生かすためにはそれらの使い方を覚える必要があります．仕事でコードを頻繁に書くようなことがなければ学習コストが高くつくのでホビー・ユーザや学習者にとっては覚えることがより少なくさまざまな機能を備えたJupyter Notebookが最適でしょう．

特徴

● 数行ずつ試しながら進められる

最もJupyter Notebookらしい特徴は，セル・コーディングというスタイルです．コードを意味のまとまりの単位で枠内に記述することができ，一部だけ実行したいときにステップ実行が容易です．通常のpyファイルでプログラムを記述すると，一気にプログラムが実行されてしまうので，一部のコードの動作だけを確認するには，行の先頭に「#」を記述してコメント・アウト（コードからコメント文へ変更）する修正が必要です．

Jupyter Notebookなら，コード記述と修正を繰り返して動作を確認しながらプログラムを開発できるようにツール設計されているので，Python学習の初学者には非常に理解しやすいはずです．

● 他のライブラリやフレームワークとの連携が容易

Jupyter Notebookは，サード・パーティ製ツールとの連携も簡単です．ビッグ・データと機械学習を使ってサービスを開発するような場合でも，Jupyter NotebookからApache Sparkを活用したりscikit-learnやTensorFlowと連携したりできるので，プログラム開発だけでなくサービス運用もJupyter Notebook上で行うと，情報を集約できて管理が簡単になります．

● 共有やエクスポートがしやすい

ウェブ・ブラウザ上でコーディング作業が行えるため，エディタ代わりに利用したり，複数の開発者でプログラムを共有したりできます．ノートブックはipynbファイルとして保存されるので，ファイルを直接他のPCにインストールされているJupyter Notebookで開いて実行や修正できます．Jupyter Notebookを利用していない開発者とコードを共有するにはノートブックをPythonプログラムだけのpy

ファイルにしたり，PDFファイルやHTMLファイルにエクスポートしたりできます．

● 記述＆その場で結果確認

▶セルにコードを記述する

ノートブックではセル・コーディングという記述スタイルで行います．イメージとしてはExcelの表にあるセルにコードを記述するような感じです．

セルには4種類あります．Pythonプログラム用の「Codeセル」，テキストやHTMLタグ用の「Markdownセル」，タイトル用の「Headingセル」，LaTeXで数式を記述できるファイル変換用の「Raw NBConvertセル」があります．

▶実行結果を記録できる

Codeセルではコード記述ができるだけでなく実行結果を記録に残せます．テキスト，Pandasのデータフレーム，NumPyの配列，Matplotlibのグラフなども表示できます．IPythonでグラフを表示するとグラフ用のウィンドで表示されるので，ノートPCのように画面が小さいとコードとグラフを並べて表示しづらいですが，ノートブックはウェブ・ブラウザ上にセルごとに全て表示されます．アクティブにしたいウィンドウを切り替えながら作業しなくてもよいので効率が良いです．

▶文字サイズや画像を入れたメモを入れられる

コーディングするときにプログラムの大まかな流れをメモしてプログラム中にコメントとして残すとPythonコードが埋もれて見づらくなります．このようなときにはCodeセルにPythonコード，Markdownセルにメモを書くようにするとコードとメモを分離でき，全体が整理されて見やすくなります．

箇条書きでアルゴリズムをメモするなど，プログラムのアイデアをメモするときにはMarkdownセルを使用するとHTMLタグが利用できるので文字サイズや箇条書きなどの文字スタイル，けい線や表を簡単に設定できます．通常のHTMLファイルのように画像や動画を利用したノートブックも作れます．

● コードの自動補完

プログラム開発向けのエディタによるあるコードの自動補完がJupyter Notebookにもあります．変数名だけでなくインポートしたライブラリのクラス名やメソッド名なども補完されます．例えば「pri」と入力した後でTabキーを押すと「print」と表示されます．もし複数の候補がある場合には候補リストが表示されるのでその中から選択します．最も利用頻度が高い機能だと思います．

Jupyter Notebookのノートブックは，手軽にコード記述と実行結果を確認できるので中小規模の開発に向いていると思います．コードを共有することが前提ならば機能は十分で，機械学習のチューニングに何度もPythonコードの実行と修正するような場面では，Jupyter Notebookを使った方がエディタとコンソール画面を行き来しなくても済むので作業効率が良いです．複数のノートブックでプログラムを同時実行させて検証できるので開発期間を短縮できるかもしれません．

なお，Jupyter Notebookで自動補完できるのは便利ですが，本格的なエディタ（例えばVisual Studio Code）を利用すると，自動補完だけでなくPythonコードをコーディング規約（PEP8）に沿って記述されているかチェックしたり，より高度な補完機能を追加できたりします．仕事でサーバなどを運用する場合にグループでコーディング・ルールを決めているときに独自のルールをエディタに拡張機能として組み込むこともできます．

またPEP8規約を全て覚えている人は少ないと思います．規約に沿ったコードを書くことに労力をかけたくないのであればVisual Studio Codeなどのツールを使うと作業効率が向上すると思います．

● 実はさまざまな言語にも対応している

実はJupyter NotebookはPython専用ツールではなく，さまざまなコンピュータ言語で利用できます．例えばJuliaやR，Haskell，Scala，node.js，Go，MathJaxで数式記述にも対応しており，40以上のコンピュータ言語に対応しています．

OSもWindows，Mac，Linuxに対応しているので，一度使い方を覚えればさまざまな場面で応用しやすいツールだと思います．

Jupyter Notebookは後継ツールが開発されており，より規模の大きな開発に対応できるJupyter Lab（`https://github.com/jupyterlab/jupyterlab`）が知られています．IPythonやIPython NotebookからJupyter Notebookに進化したように，今後も時代の要求に合わせた機能拡張が進むと思います．Jupyter LabはJupyter Notebookに統合開発環境（IDE）が組み込まれるイメージのようで既存の商用版統合開発ツールの代替えになると予想しています．

● シングル・ページ・アプリケーションのように使う

Jupyter NotebookをPythonプログラムの開発・実行環境としてだけ使うのはもったいないと思います．インタラクティブなデータの可視化機能があるのでプログラムの処理結果をグラフや表としてノートブック内で表示させることができます．この特徴を生かして機械学習プログラムの中間データを確認したり特定セルのプログラムだけを再実行したりすると，デバッグ

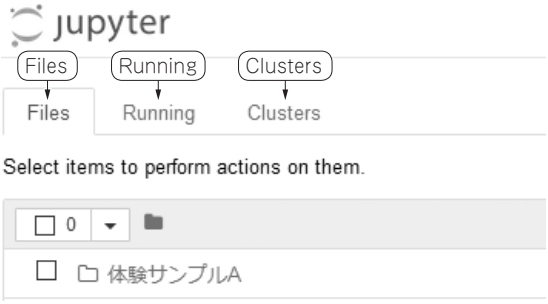

図2　Jupyter Notebookのログイン後に表示される画面

やパラメータ修正が効率良く行えます．

例えばrunipyを使えばipynbファイルをパースして中のPythonコードをIPythonで順に実行できます．これを応用するとcronで定期的にプログラムを実行できるのでノートブックからpyファイルへの書き出しを手作業で行わなくても済みます．

またJupyter Notebookのファイル共有機能を生かしてラズベリー・パイをJupyterサーバにすると便利です．Pythonプログラムの実行はラズベリー・パイ上で処理されるのでPCと比べると高速な処理は行えませんが，LinuxとPythonを同時に学習したいときPCに開発ツールをセットアップしなくても済むので，スティックPCやタブレットなどのディスク容量が少ないPC環境でもPythonプログラミングを手軽に行えます．

ラズベリー・パイを低消費電力のファンレス・サーバとして運用すれば，ノートブックをそのままシングル・ページ・アプリケーション的なツールや簡易的なRPA（Robotics Process Automation）ツールとして応用できると思います．最近話題のRPAツールをPythonとJupyter Notebookで作ってしまうのはなかなか良いアイデアだと思います．Pythonでデータベース・サーバやウェブ・サーバと連動する機械学習を開発してウェブ・サービスとして公開するのも結構簡単かもしれません．

● Jupyter Notebookと構成管理ツールを組み合わせて基盤構築も自動化できる

Jupyter Notebookをプログラム開発のエディタにしか利用しないのはもったいないと思います．例えばノートブックにインフラ構成のパターン，その構成手順を記述して実行する応用が考えられます．

構成管理ツール（Ansible，Chef，Puppetなど）と組み合わせるとサーバを自動構築できます．たくさんのサーバを基盤構築するような場面は手順書とスクリプトを分けて作成することがよくあると思います．Jupyter Notebookを利用すれば手順書，スクリプト，レポートを1つのノートブック上で表現することもできます．サーバのデプロイと同時にレポートまで自動生成してしまうような使い方も考えられます．パブリック・クラウドを活用して仮想サーバや仮想ネットワークを構成するときに威力を発揮しそうです．通常，パブリック・クラウドでサーバなどのインスタンスを設定するにはウェブ・ブラウザで環境を構築する際に全てGUI操作すると煩雑な作業になります．Jupyter Notebookと管理ツールを使ってパブリック・クラウドをCUI操作すればデプロイ作業を自動化できるはずです．

よくパブリック・クラウド環境を新入社員の研修や一時的な検証作業，システムのプロトタイピングに利用することがあるので，Jupyter Notebookを利用して本番環境構築用ノートブックとか，開発環境構築用ノートブックなどを作成すれば構成情報を管理するのも簡単になると思います．ノートブックの実行結果はPDFファイルとしてエクスポートすれば構築記録代わりに履歴を残すことができます．

画面表示

Jupyter Notebookへログイン後に表示される画面にはFiles，Running，Clustersの3つのタブがあります（**図2**）．

● タブその1…Files

Filesタブではjupyter Notebookのカレント・ディレクトリが表示されます．ここではノートブック（ipynb形式ファイル）だけでなく，カレント・ディレクトリ配下のファイルを見ることができ，複製（Duplicate），停止（Shutdown），ビュー（View），編集（Edit），リネーム（Rename），移動（Move），ダウンロード（Download），削除（ゴミ箱アイコン）が行えます．

これらの操作を行うにはファイル名の前にあるチェック・ボックスにチェックを入れる必要があります．分かりにくいのがビュー（View）と編集（Edit）です．前者がノートブック形式の表示，後者がテキスト・エディタ形式の表示になります．

● タブその2…Running

Runningタブではjupyter Notebookで起動中の全てのターミナルやノートブックを一覧表示できます．Filesタブではカレント・ディレクトリの起動状態しか見られませんが，Runningタブでは全ノートブックの起動状態を一覧でき，不要なターミナルやノートブックのプロセスを「Shutdown」ボタンで手動停止させることもできます．

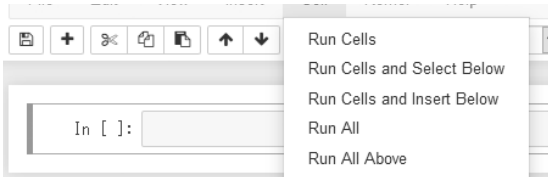

図3　新規ノートブックの画面

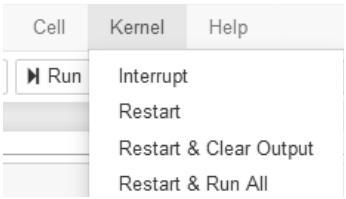

図4
ノートブックをリセットしたいなら「Kernel」メニューの「Restart & Clear Output」をクリック

● タブその3…Clusters

Clustersタブは複数台のコンピュータで並列演算するときに利用します．Jupyter Notebookの並列演算はあらかじめプロファイルを作成しておき，クラスタ起動時にエンジン数（コンピュータ数）を指定して実行，プロセスが不要になれば停止できます．

このようにJupyter Notebookは並列コンピューティングのインターフェースとして利用できるツールですので大規模なシステムでの活用も可能です．

基本的な使い方

● 新規作成＆名前付け

Jupyter Notebookで新規ノートブックを作成するには，ホーム画面のFilesタブを開いた状態で右上にある「New」ボタンをクリックして「Python 3」を選択します．ノートブック以外にも新規テキスト・ファイルを作成する「Text File」，新規フォルダを作成する「Folder」，ターミナル画面を表示する「Terminal」もあります．

新規ノートブックが作成されるとウェブ・ブラウザに表示されます．新規タブへの名称登録は，ノートブックの上に「Untitled」と表示される箇所に記述できます．「Untitled」と表示される箇所をクリックして「Rename Notebook」ウィンドウでタイトルを入力して「Rename」ボタンをクリックする（またはEnterキーを押す）ことで，ノートブックの名称を変更できます．

作成済みのノートブックを開かずに名称を変更するにはJupyter Notebookのホーム画面のFilesタブから該当ノートブックが「Running」でないことを確認してチェック・ボックスにチェックを入れます．もしノートブックが「Running」であればリネームできないため，チェックを入れてメニューの「Shutdown」をクリックして停止します．停止状態でチェックを付けてメニューの「Rename」ボタンをクリックすることで変更できます．

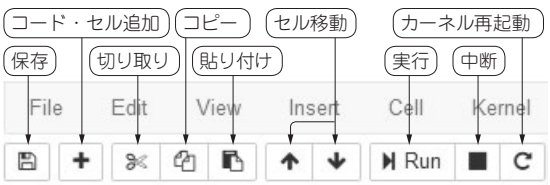

図5　セル・コード・タイプのプルダウン・メニュー

● 基本の操作メニュー

Filesタブの下にはファイルにチェックを付けたときだけ表示されるメニューがあります．メニューにはFile，Edit，View，Insert，Cell，Kernel，Helpがあり，各メニューをクリックするとプルダウン・メニューが表示され，さまざまな機能が利用できます（**図3**）．

Jupyter Notebookでプログラムを実行する方法を簡単に説明します．ノートブックでプログラムをセルごとに実行するには，実行したいセルを選択して「Run」ボタンをクリックするとセルごとに実行されます．セルごとに処理結果を確認したり，特定のセルのプログラムを再実行したりするときに便利です．

全部一度に実行したい場合にはCellメニューの「Run All」をクリックするとよいでしょう．

Notebookは一度実行すると結果が記録されます．もし実行結果を全て消してノートブックをリセットしたいなら「Kernel」メニューの「Restart & Clear Output」をクリックします（**図4**）．このとき変数などに格納された値も全てリセットされて消えてしまいます．ノートブックのリセットと全てのセルのプログラムを実行するには「Kernel」→「Restart & Run All」をクリックするとよいです．

● ツール・バー

メニューの下にはアイコン表示のツール・バーがあります（**図5**）．

◆参考・引用*文献◆

(1) Using IPython for parallel computing.
https://ipyparallel.readthedocs.io/en/latest/
(2) Parallel examples.
https://ipyparallel.readthedocs.io/en/latest/demos.html
(3) 東京大学 金田 康正 教授のウェブ・サイト．
http://www.super-computing.org
(4) runipy.
https://github.com/paulgb/runipy

さとう・せい