Python プログラミングの手引き

京都大学大学院情報学研究科　今井貴史
後藤貴宏
石田達大
小山和輝
京都大学工学部情報学科　枡井啓貴

平成 28 年 12 月 24 日

目 次

§ 1.　Python 環境の構築手順

本節ではデータ解析プラットフォーム Anaconda の導入手順を説明する．Anaconda は Python のディストリビューションの一つであり，その最大の特徴はデータ解析に有用な追加パッケージを多数そろえていることにある．また，それらのパッケージの間の依存関係を処理するために独自のパッケージ管理システム Conda が用意されていることも特徴の一つである．Conda を用いることで，パッケージの導入・アップデート・削除のほか，Python 仮想環境の構築・管理なども容易に行うことが可能となる [†1]．

§ 1.1.　Windows の場合

a) 新規インストール

1. 公式サイトのダウンロードページ (`https://www.continuum.io/downloads/`) から自分の Windows 環境（32 ビット版／64 ビット版）[†2] に合ったインストーラをダウンロードする．
2. ダウンロードしたインストーラを実行し，表示された指示に従ってインストールする．

⚠ ほかの Python ディストリビューション（ActivePython など）がインストールされている場合，Anaconda をインストールしても，設定によっては前者が優先的に使用されてしまう．優先順位を確認するためには，コマンドプロンプト（cmd.exe）上で次のコマンドを実行すればよい．

```
> where python.exe
```

このコマンドの出力の先頭行に Anaconda 以外の Python が表示されたときは，以下の手順で Anaconda の Python が優先的に使用されるように変更する．

1. コントロールパネルから［システム］→［システムの詳細設定］→［環境変数］と進み，システム環境変数の「Path」を選択して［編集］ボタンを押す．
2. 変数値から Anaconda 関係のエントリを切り取り，ほかの Python ディストリビューションに関するエントリより前に貼り付ける．
3. ［OK］ボタンを押して設定を保存・反映させる．

⚠ Cygwin Terminal 上では Windows の「Path」の前に`/usr/local/bin:/usr/bin` が自動で追加され，Cygwin の Python が優先的に使用されてしまう．Cygwin Terminal 上でもデフォルトで Anaconda の Python が使用されるようにするためには，Cygwin Terminal 上でさらに次のコマンドを実行する．

```
$ echo 'export PATH=<Anaconda directory>:$PATH' >> ~/.bashrc
$ echo "alias python='python -i'" >> ~/.bashrc
```

ここで，`<Anaconda directory>`には Anaconda のインストールディレクトリを Cygwin 形式で記入すること（例えば`/cygdrive/c/Anaconda3`）．Cygwin Terminal を開きなおせば，Anaconda の Python が優先されるようになっているはずである．

[†1] この辺りの詳細については公式ドキュメント (`http://conda.pydata.org/docs/index.html`) などを参照されたい．

[†2] コントロールパネルから［システム］を開き，「システムの種類」で確認可能．

b) 旧バージョンからのアップデート

コマンドプロンプト（cmd.exe）を開き，以下の手続きを実行する．

1. パッケージ管理システム自体をアップデートする．

```
> conda update conda
```

2. パッケージ群をアップデートする．

```
> conda update anaconda
```

§ 1.2. Mac OS X の場合

⚠ Mac OS X に関しては，公式インストーラを用いて Anaconda を導入したところ Conda がほかのパッケージ管理システムと衝突した，という事例が報告されている．そこで，よりクリーンな導入方法として，ここではパッケージ管理システム Homebrew と Python バージョン管理ツール pyenv とを用いた方法を紹介する．

a) 新規インストール

ターミナル（Bash[†3]）を開き，以下の手続きを実行する．

1. （OS を El Capitan より前のバージョンから El Capitan 以降にアップデートした場合）`/usr/local` のパーミッションが書き換えられてしまっている可能性がある．以下のコマンドでパーミッションを復元する．

```
$ sudo chown -R $(whoami):admin /usr/local
```

2. （プロキシを介してインターネットを利用している場合）環境変数でプロキシを指定する．

```
$ export http_proxy=http://<proxyhost>:<proxyport>
$ export https_proxy=https://<proxyhost>:<proxyport>
```

ここで，`<proxyhost>`にはプロキシの URL，`<proxyport>`にはプロキシのポート番号を記入すること．

3. （Homebrew がインストールされていない場合）

(i) Command Line Tools for Xcode をインストールする．

```
$ xcode-select --install
```

(ii) Homebrew をインストールする [†4]．

```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com\
  /Homebrew/install/master/install)"
```

（Homebrew がすでにインストールされている場合）

Homebrew 自体とパッケージ一覧をアップデートしたうえで，全パッケージをアップデートする．

[†3] ターミナルとして Bash 以外を使用している場合はコマンドを適当に変更すること．

[†4] Xcode のライセンスに同意していないことで警告が出た場合，表示された指示に従ってライセンスに同意する．

```
$ brew update
$ brew upgrade
```

4. Homebrew を用いて pyenv をインストールする．

```
$ brew install pyenv
$ echo 'export PYENV_ROOT=/usr/local/var/pyenv' \
  >> ~/.bash_profile
$ echo 'eval "$(pyenv init -)"' >> ~/.bash_profile
$ source ~/.bash_profile
```

5. (i) pyenv で導入可能な Anaconda のバージョン一覧を確認する．

```
$ pyenv install --list | grep 'anaconda3'
```

(ii) pyenv を用いて Anaconda をインストールし，さらにデフォルトの Python 環境として設定する．

```
$ pyenv install anaconda3-<Anaconda version>
$ pyenv global anaconda3-<Anaconda version>
$ pyenv rehash
```

ここで，<Anaconda version>にはインストールしたい Anaconda のバージョンを記入すること．特に理由がなければ最新版（2016 年 7 月 21 日の時点では 4.0.0）を指定すればよい．

b) 旧バージョンからのアップデート

ターミナル（Bash）を開き，以下の手続きを実行する．

1. （OS を El Capitan より前のバージョンから El Capitan 以降にアップデートした場合）/usr/local のパーミッションが書き換えられてしまっている可能性がある．以下のコマンドでパーミッションを復元する．

```
$ sudo chown -R $(whoami):admin /usr/local
```

2. （プロキシを介してインターネットを利用している場合）環境変数でプロキシを指定する．

```
$ export http_proxy=http://<proxyhost>:<proxyport>
$ export https_proxy=https://<proxyhost>:<proxyport>
```

ここで，<proxyhost>にはプロキシの URL，<proxyport>にはプロキシのポート番号を記入すること．

3. Homebrew 自体とパッケージ一覧をアップデートしたうえで，全パッケージをアップデートする．

```
$ brew update
$ brew upgrade
```

4. pyenv を用いて最新バージョンの Anaconda を新規インストールする．(手順については新規インストールのステップ 5 を参照.)

5. （必要に応じて）旧バージョンをアンインストールする．

(i) インストールされている Anaconda のバージョン一覧を確認する．

```
$ pyenv versions
```

(ii) pyenv を用いて Anaconda をアンインストールする．

```
$ pyenv uninstall anaconda3-<Anaconda version>
```

ここで，<Anaconda version>にはアンインストールしたい Anaconda のバージョンを記入すること．

§ 1.3. 動作テストのためのサンプルコード

動作テストのために，コード 1 にサンプルコードを掲載する．実行すれば（実行方法については第 2 節を参照）図 1 のような結果が得られるはずである．

コード 1: 動作テストのためのサンプルコード．矩形波に対する三角多項式近似をプロットする．

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  ## 矩形波を三角多項式で近似する関数を定義
5  def square_wave(n):
6      x = np.arange(-8, 8, 0.001)
7      y = [0 for z in x]  # リスト内包記法
8      for k in range(n+1):
9          if k % 2 == 1:
10             y += (4 / (np.pi * k)) * np.sin(x * k)
11     plt.plot(x, y, label = "n = " + str(n))
12         # plt.show()されるまでグラフは表示されない
13
14
15 ## 上で定義した関数を、引数を変えつつ呼び出す
16 square_wave(1)
17 square_wave(5)
18 square_wave(11)
19 square_wave(101)
20 square_wave(10001)
21
22 plt.legend()  # グラフに凡例を表示させるように設定
23 plt.show()  # これまでにplotしたグラフを表示
```

§ 2. Jupyter Notebook の使い方

Python を用いてデータ解析を行ううえでは，対話的コンピューティング向けに Python 標準のシェルを機能強化した IPython が有用である．IPython には外部プログラムから IPython の機能を利用するための機構が用意されている．その機構を用いた IPython のフロントエンドの一つが Jupyter Notebook である．本節では Jupyter Notebook を用いて Python コードを編集・実行する方法を説明する．

なお，IPython の機能を利用して Python のコードを編集・実行するというのは Jupyter Notebook にとって副次的な機能に過ぎない．Jupyter Notebook の主たる機能は，ノートブックとよばれる，コードと出力を文章中に埋め込んだ文書を作成する機能である．この機能の詳細については参考文献 [1] や公式ドキュメント (https://jupyter-notebook.readthedocs.io/en/latest/notebook.html) などを参照されたい．

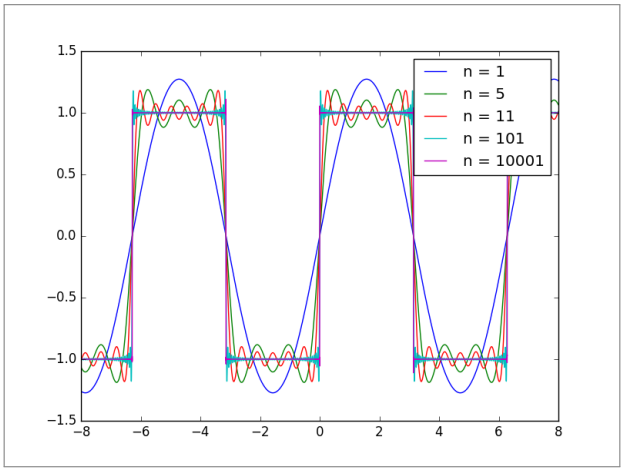

図 1: コード 1 の実行結果.

§ 2.1.　ユーザインタフェースの特徴

a) ユーザインタフェースの構成要素

Jupyter Notebook のユーザインタフェースは Notebook Dashboard，Notebook Editor，および File Editor からなる．

Notebook Dashboard は Jupyter Notebook の起動直後に表示されるページで，簡単なファイルマネージャとしての機能をもち，新しいノートブックを作成したり，既存のノートブックを開いたりするために用いられる．また，動作中の計算エンジンを強制停止させる機能も備えている．

Notebook Editor は Notebook Dashboard からノートブックを開いた際に表示されるページで，その名の通りノートブックを編集するためのインタフェースである．vi に代表されるモーダルエディタ（モードをもつエディタ）の一種であり，後述の二つのモードをもつ．

File Editor は Notebook Dashboard からノートブック以外を開いた際に表示されるページで，ごくシンプルなテキストエディタである．

b) Notebook Editor のモード

Notebook Editor はモーダルエディタであり，編集モードとコマンドモードという二つのモードをもっている．メニューバーの右方に［✎］マークが表示されていれば編集モード，表示されていなければコマンドモードになっている．

編集モードは，セルにテキストを入力するといったセル内部に対する操作を行うためのモードである．編集モードに切り替えるには，セルの入力領域をクリックするか，もしくは [Enter] キーを押す．

コマンドモードは，セルの新規作成などのセル単位での操作やノートブックの保存のようなノートブック全体に関する操作を行うためのモードである．コマンドモードに切り替えるには，入力領域以外をクリックするか，もしくは [Esc] キーを押す．

§ 2.2.　基本的な操作方法

a) Jupyter Notebook の起動

コマンドプロンプト（cmd.exe）もしくはターミナルを開き，作業用のディレクトリに移動したうえで `jupyter notebook` コマンドを実行する．

b) ノートブックの新規作成

Notebook Dashboard において，右上の［New］ボタンを押し，表示されたリストの中から使用する計算エンジン（「Python [Root]」または「Python 3」）を選択する（図 2）[†5]．

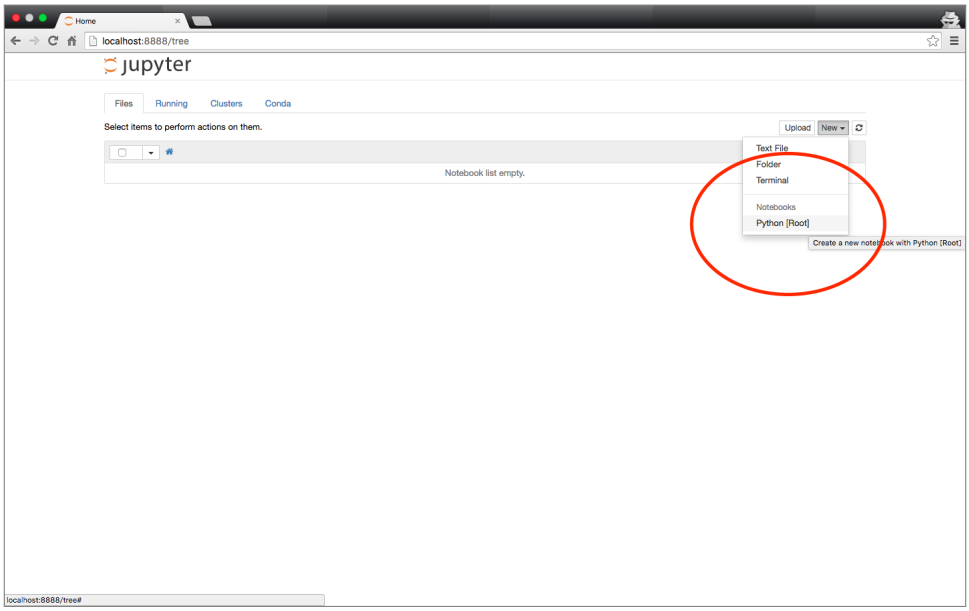

図 2: ノートブックの新規作成．

c) 既存ノートブックの読み込み

開きたいノートブック（拡張子 ipynb のファイル）を Notebook Dashboard から選択する．

d) Python コードの入力

Notebook Editor において以下のように操作する．

1. コードを入力したいセルを選択する．
2. ツールバーのプルダウンメニューにおいてセルタイプ「Code」が選択されていることを確認する．セルタイプが「Code」以外になっている場合は，プルダウンメニューからセルタイプ「Code」を選択するか，コマンドモードに切り替えてキーボードショートカット [y] を使用する．
3. 編集モードに切り替え，セルにコードを入力する（図 3）．

[†5] ブラウザからローカルのプログラムが実行されることになるので，一部のウイルス対策ソフトウェアはこのとき Python をウイルスと判定してしまう．ウイルスと判定された場合，(この問題であることを十分に確認したうえで）Python を例外に追加する．

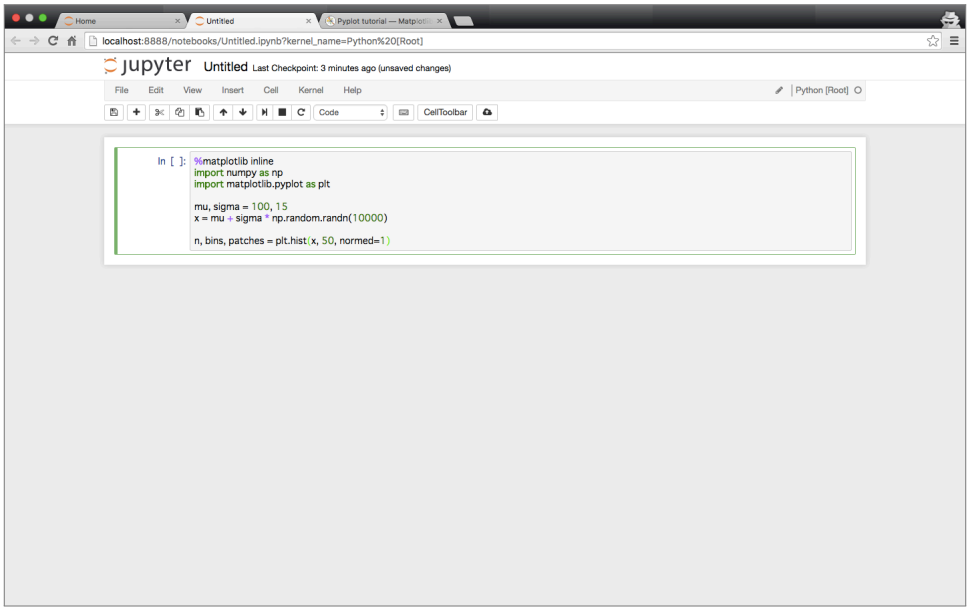

図 3: Python コードの入力.

e) Python コードの実行

Notebook Editor において以下のように操作する.

1. 実行したいコードが入力されたセルを選択する.
2. ツールバーの［⏭］ボタンを押すか，キーボードショートカット [Shift]+[Enter] （コマンドモード／編集モードのいずれでも使用可）を使用する．実行中のセルはセル番号が `In [*]` と表示される.

f) ノートブック名の変更

Notebook Editor の上部，Jupyter ロゴの横に表示されているノートブック名をクリックし，新しいノートブック名を入力する（図 4）.

g) ノートブックの上書き保存

Notebook Editor において，ツールバーの［💾］ボタンを押すか，コマンドモードに切り替えてキーボードショートカット [s] を使用する.

h) ノートブックの別名保存

Notebook Editor の［File］メニューから［Make a Copy...］を選択する.

i) Jupyter Notebook の終了

1. Notebook Editor の［File］メニューから［Close and Halt］を選択することで，Notebook Editor を閉じると同時に計算エンジンを停止させる [†6].

[†6] 計算エンジンを停止させずに Notebook Editor を閉じてしまった場合は，Notebook Dashboard の［Running］タブを開き，当該ノートブックの［Shutdown］ボタンを押して停止させる.

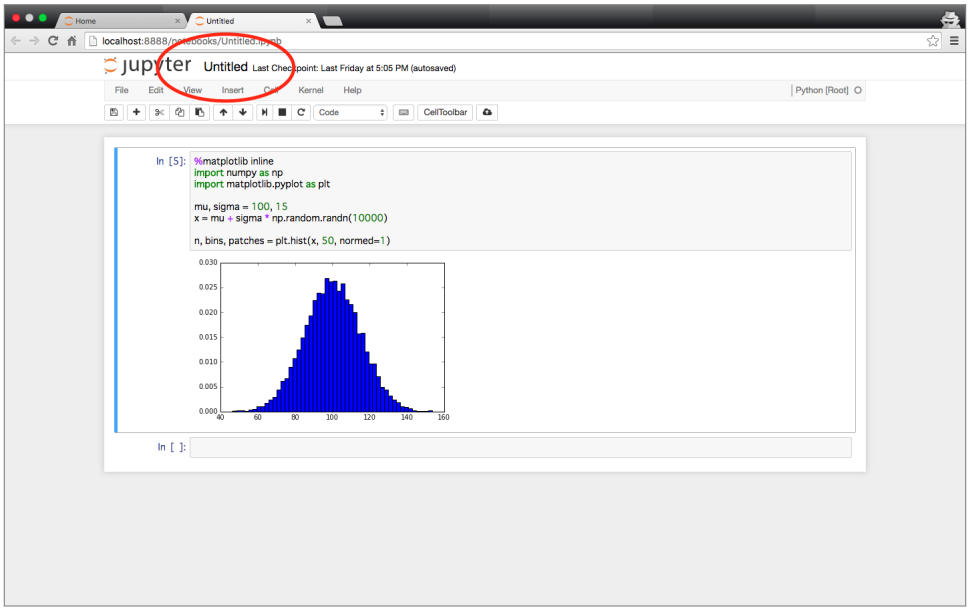

図 4: ノートブック名の変更.

2. Notebook Dashboard を閉じる.(ブラウザを閉じればよい.)

3. シェル(コマンドプロンプトもしくはターミナル)上に残ったサーバジョブを [Ctrl]+[c] キーで停止させる.

参考文献

[1] Cyrille Rossant(著), 菊池彰(訳)『IPython データサイエンスクックブック: 対話型コンピューティングと可視化のためのレシピ集』(オライリージャパン).

§ 3. Python プログラミングのヒント

§ 3.1. MATLAB データの読み込み

Python では Matlab のデータも扱うことができる. ここではその方法を記す. 例えば, 次のような構造体 s が保存されたバージョンが 7 以前の Matlab ファイル 'test.mat' を考える.

- s.label =['node1','node2'] : 2×5 char
- s.samplingrate = 0.05
- s.DATA : 2×5×2 double

$$\text{s.DATA}(:,:,1) = \begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{matrix}, \text{s.DATA}(:,:,2) = \begin{matrix} 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{matrix}$$

- s.average = [3,8;13,18] : 2×2 double

このような Matlab ファイルを読み込むには scipy.io にある loadmat を使えばよい.

```
1 d=scipy.io.loadmat('$ファイル名')
```

と読み込んだ変数 d に対して，d[‘$変数名’][0,0][‘$フィールド名’] とすれば構造体に含まれているフィールドの値を取得することができる．実際に test.mat を用意した上でコード 2 を実行すると上記で記した構造体 s が読み込まれていることを確認できる．

コード 2: loadmat を用いて構造体 s を読み込むコード．各フィールドの値を表示する．

```
import scipy.io

## Matlabファイルの読み込み
d = scipy.io.loadmat("test.mat")

## Matlab構造体の取得
s = d["s"][0,0]

## 各フィールドの内容を表示
print(s["label"],"\n")
print(s["samplingrate"],"\n")
print(s["DATA"],"\n")
print(s["average"],"\n")
```

他方で Matlab ファイルのバージョンが 7.3 であれば，loadmat でファイルを読み込むことができない．この場合はパッケージ h5py を用いることになる．この h5py はデフォルトでは python には導入されていないので h5py をインストールする．

```
> conda install h5py
```

そして h5py にある File を用いて，

```
d=h5py.File(‘$ファイル名’,’r’)
```

とすればよい．このようにして読み込んだ d に対して，d[‘$変数名/$フィールド名’] のようにすれば構造体に含まれているフィールドの値を取得することができる．ただし，loadmat のときとは異なり，フィールドの値取得後の文字列の形式・次元の向きには注意が必要である．実際にバージョンが 7.3 である test.mat を用意した上でコード 3 を実行すると上記で記した構造体 s が読み込まれていることを確認できる．これらの Matlab ファイルの読み込みに関する詳細としてはリンク集にある Scipy のドキュメントおよび h5py のドキュメントを参考されたい．

コード 3: h5py を用いて構造体 s を読み込むコード．各フィールドの値を表示する．

```
import h5py

## Matlabファイルの読み込み
d = h5py.File("test.mat","r")

## 各フィールドの内容を表示
print(list(map(lambda x : ''.join((map(chr,x))),d["s/label"][:].T)))
print(d["s/samplingrate"][:],"\n")
print(d["s/DATA"][:,:,:],"\n")
print(d["s/average"][:,:],"\n")
```

§ 3.2. 図のファイル出力

図のファイル出力は matplotlib.pyplot にある savefig を使えばよい．使い方としては，コード 1 における plt.plot() の部分を

```
plt.savefig(’$ファイル名’)
```

とすればよい．出力画像のフォーマットとしては

eps, pdf, pgf, png, ps, raw, rgba, svg, svgz

などが可能である．

§ 3.3. リンク集

- PythonJapan(`http://www.python.jp/`)：Python の日本語ドキュメント
- SciPy.org(`https://docs.scipy.org/doc/`)：Numpy と Scipy のドキュメント
- Matplotlib(`http://matplotlib.org/`)：Matplotlib のドキュメント
- HDF5 for Python(`http://docs.h5py.org/`)：h5py のドキュメント