一般教育演習

プログラミングで問題を解く：集計から人工知能まで

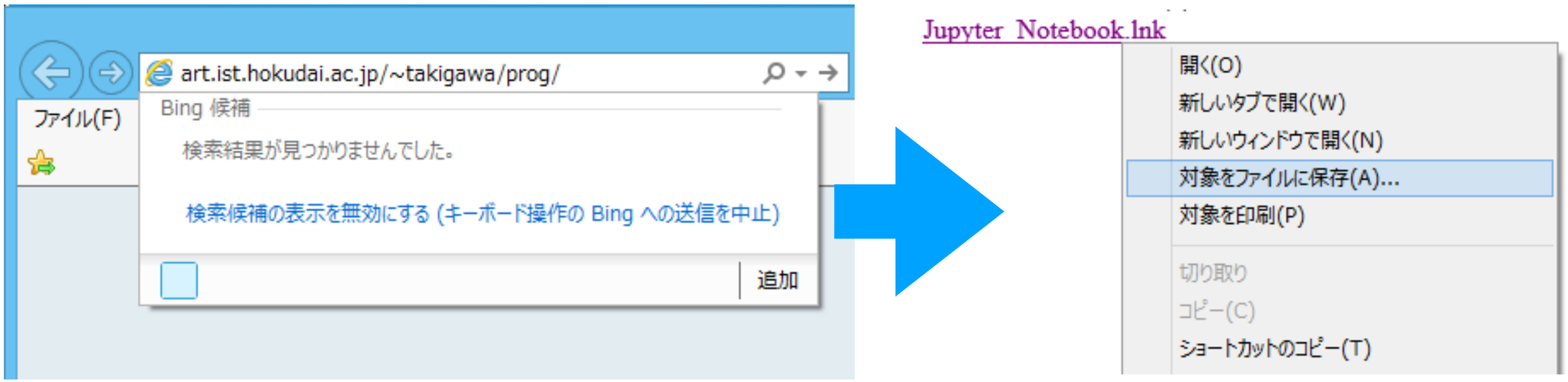
瀧川 一学
工学部 情報理工学コース

今日のお題：Pythonを始めよう

- 前回の復習とふりかえり
- paiza.IOでPythonに触れてみる
- Python言語の基本を学ぶ + 最初はどうやって勉強すればいいか？
- Pythonインタプリタと実行環境について
- USBメモリのPython環境の使い方 (改)
- Jupyter Notebookの動かしかた
- (補足) なぜ前回動かなかったか?
- (補足) USBメモリについて
- ミニレポートタイム

授業の情報

http://art.ist.hokudai.ac.jp/~takigawa/prog/



講義資料やサンプルなどは「右クリック→対象をファイルに保存」で一旦ダウンロードしてから使用

試しに今日のスライドをダウンロードしてダブルクリックで開いてみて

先週のお題：プログラミング事始

- 前回の復習とふりかえり
- コンピュータの仕組みとプログラム、社会における役割
- プログラミング言語とは？
- Pythonとは？
- Lightbotをやってみる / Scratchをやってみる
- (時間があれば) セットアップしたPython環境を試す
- ミニレポートタイム(15分)

1.Lightbotについて
(ア)結構楽しくできた (イ)楽しかったが結構むずいな(ウ)途中から頭が痛くなってきました
(エ)どうもこういうのはすごく苦手(オ)そもそもあらゆるゲーム自体が苦手だ
結果：だいたい(ア)、たまに(イ)や(ウ)

2.Lightbot(やScratch)とプログラミングの関係について
(ア)まだよく分からない (イ)分かったような分からないような(ウ)なんとなく少し分かった (エ) ある程度わかった(オ)なるほどそういうことかと思った (カ)元から分かっていた
結果：だいたい(ウ)か(エ)、たまに(イ)や(オ)

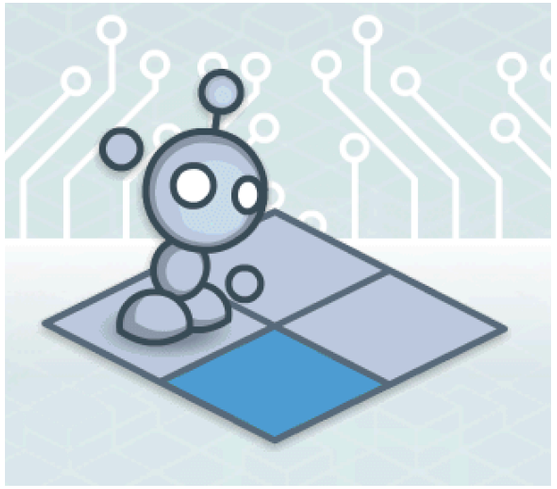
3.USBメモリへのAnacondaのインストールは？
(ア)してない/するのを忘れていた(イ)やろうとしたができなかった(ウ)結構ハマったけどなんとかできた (エ)問題なくできた
結果：かなりバラけた→今日まだできてない人は個別に対応しようと思います

4.USBメモリへのAnacondaのインストールにかかった時間は？(待ち時間含む)
(ア)10分未満 (イ)10分以上40分未満 (ウ)40分以上60分未満(エ)60分以上90分未満(オ)90分以上2時間未満 (カ) 2時間以上 (キ) できなかった
結果：

5.今日の授業の内容は？
(ア)聞いてなかった(イ)聞いてたけどついていけなくなった(ウ)授業前よりは理解が進んだ (エ)なるほどと合点がいった (オ)そこまで詳しく知りたいわけではないかな(カ)もう少し詳しめに知りたい
結果：(ウ)が多く(エ)が次、たまに(オ)や(カ)

6. 今のところの授業で一番楽しめたのを敢えて挙げると？
(ア)lightbotをやってみたこと(イ)Pythonの具体的なセットアップ(ウ)プログラミングとは？というガイダンスの概論 (エ)プログラミング言語とはの話(オ)コンピュータの仕組みの話 (カ)Python自体を少し触って見たこと(キ)全体的に楽しめた
結果：(ア)が圧倒的に多く、それ以外はバラけた

ロボ



特定のコマンド(命令)

しか解さない

ロボが解するコマンド(命令)を

順番に伝える

プログラミング

＝ プログラムする

＝ プログラムを作る

コンサートの「プログラム」

新人研修の「プログラム」

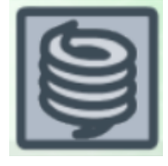
ロボの動作の「プログラム」

：

ロボ

どういう
コマンドを
使える？
(知識の問題)

ロボが解するコマンド(命令)を
順番に伝える



コマンドをどう
効率的に並べる？
(言語・表現の問題)

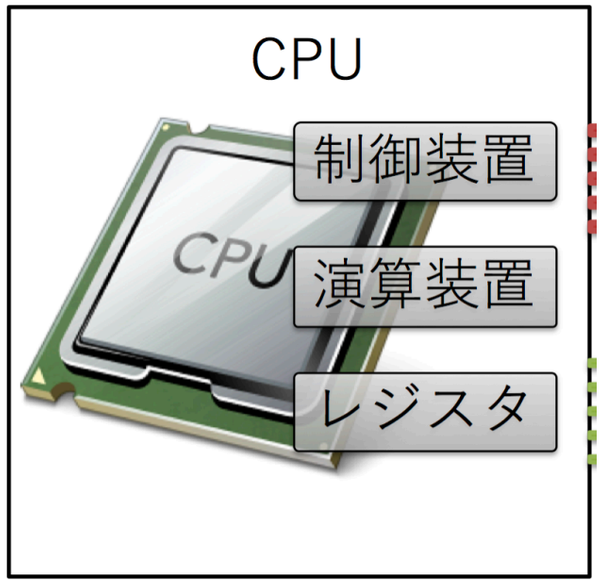
コンサートの「プログラム」

新人研修の「プログラム」

ロボの動作の「プログラム」

：

CPU



特定のコマンド(命令)

しか解さない

ロボが解するコマンド(順番)を

順番に伝える

プログラミング

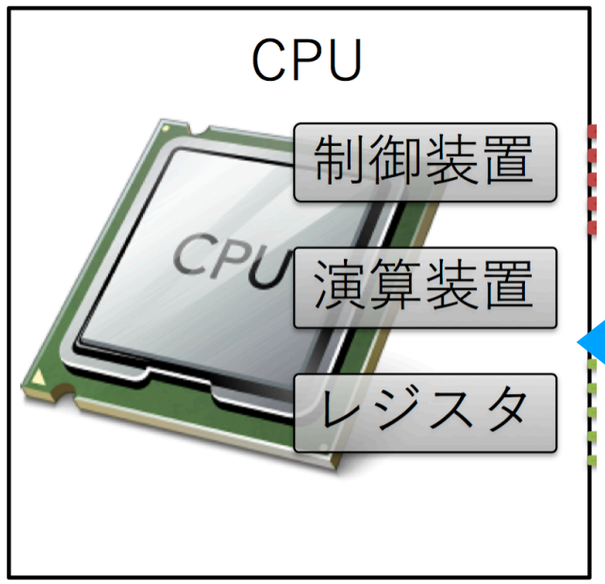
= プログラムする

ラムを作る

大問題！ 😫

これだと人間が長大な0と
1の列を一つのミスもなく
並べなければならない！

CPU



特定のコマンド(命令)

しか解さない

CPUが理解できる

コマンド列(機械語)

翻訳

人間が理解できる
レベルでやってほしい
ことを記述
(プログラミング言語)

1. **画面を真っ黒に塗りつぶす**
2. **左上から100pixel右、2000pixel下の位置に大きさ50pixelの☆を描画**
3. **それを黄色で塗りつぶす**
4. **同じ作業を右に400pixelずつづらしながら5回繰り返す**

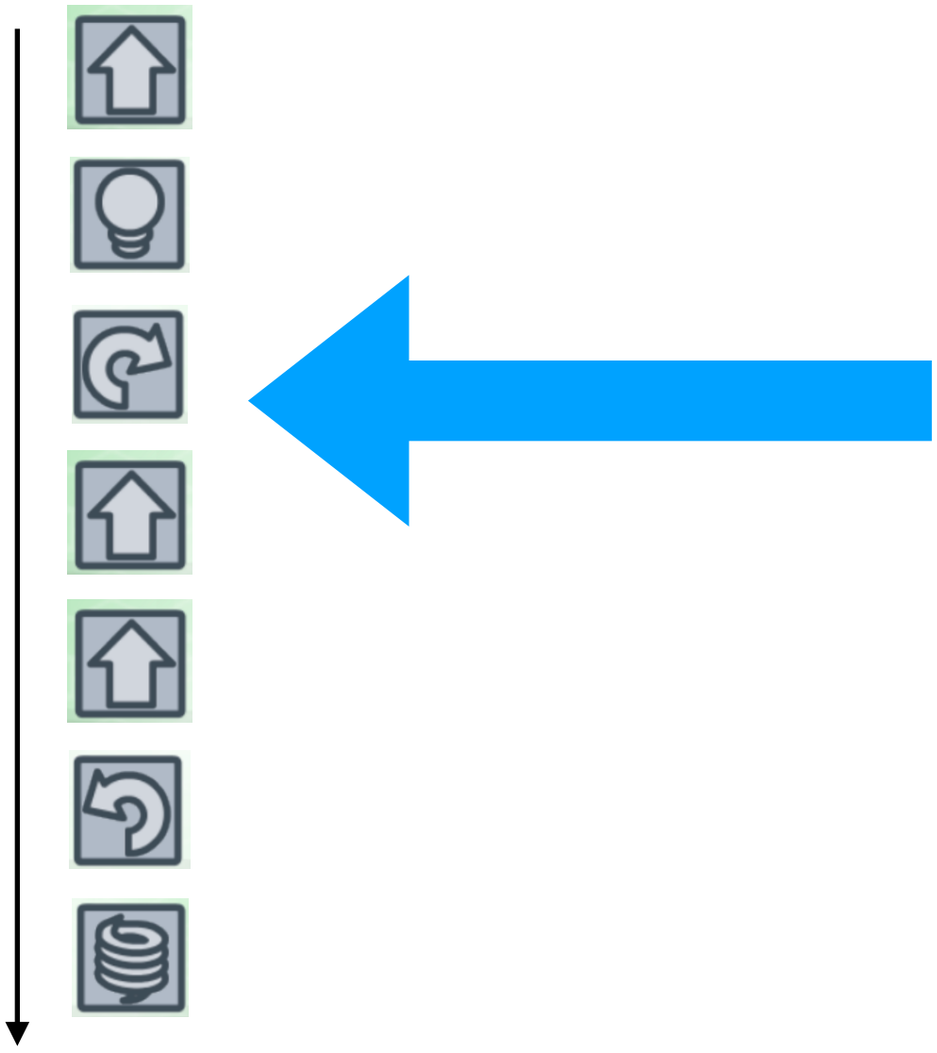
言語のルールを決めて
翻訳プログラム
も用意しておく

翻訳のやり方は大きくふた通り

1. コンパイル方式

2. インタプリタ方式

1. コンパイル方式



1. 画面を真っ黒に塗りつぶす
2. 左上から100pixel右、2000pixel下の位置に大きさ50pixelの☆を描画
3. それを黄色で塗りつぶす
4. 同じ作業を右に400pixelずつづらしながら5回繰り返す

翻訳業者方式？

- 全体を一気にまとめて訳す
- 全体の文脈がわかるので無駄がない(質の高い)コマンド列が作れる

2. インタプリタ方式

1. 画面を真っ黒に
   塗りつぶす
2. 左上から100pixel右、
   2000pixel下の位置に
   大きさ50pixelの
   ☆を描画
3. それを黄色で塗りつぶす
4. 同じ作業を右に400pixel
   ずつづらしながら5回繰
   り返す

：
：
：

同時通訳方式？

- 一まとまりづつ訳す
- 全体を作らなくても部分的に実行可能
- ちょっとづつ試しながら作れる

Python

- 非常に読みやすく書きやすい言語で最初に学ぶ言語としてもとても良い。実際海外の大学ではほぼPythonになりつつある。Webに情報や教材が多く自学しやすい。
- インタプリタ方式でちょっとずつ実行もできるし、AnacondaやJupyterなどの環境も手軽に利用可能
- 使い捨てのちょい作業から数百万行あるような大規模システムまで非常に幅広く使われており実用的な言語
- 本演習で挙げた「集計から人工知能まで」を考えると現在データ解析や人工知能実装で最も使われている。

大雑把なビジョン

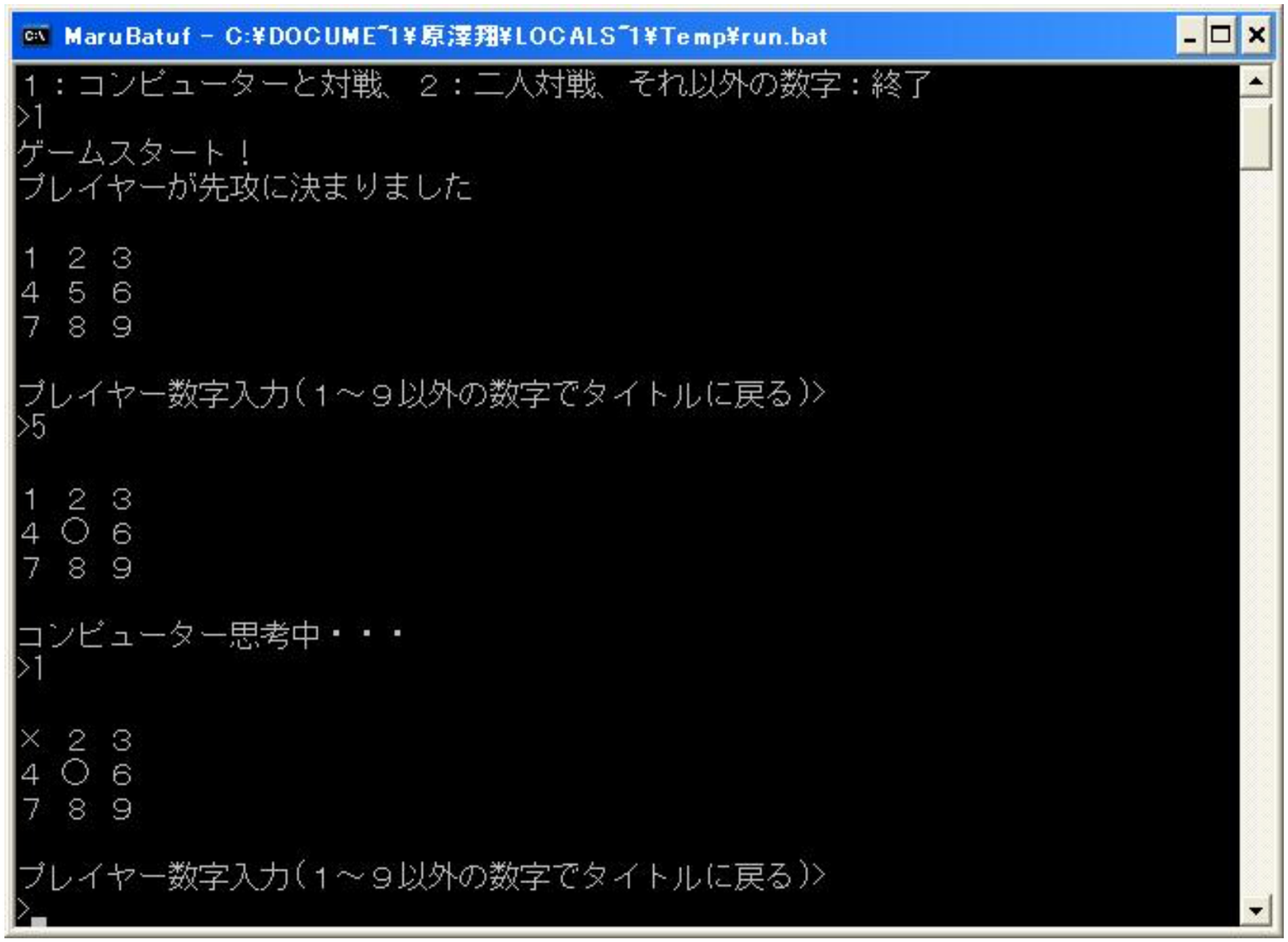
- この演習でやりたいことは大きく２つ

1. 「プログラミング」とは何かを理解しそれを
   どうやって学んでいけば良いかの指針を与える

2. 「Python」というプログラミング言語の基礎と
   実習を実際にやってみる

現代のプログラムは複雑化している

- 昔のコンピュータのオペレーティングシステムを作るのと、Windows 10やMac OSを作るのとではプログラムの複雑さがかなり異なる
- 初代ファミコンのゲーム開発 vs プレステ4のゲーム開発



良い点

- 非常に高機能で信頼性も高いソフトウェアの実現
- 機械をあまり意識しなくて良い高レベルなプログラミング言語
- コンピュータの性能の向上
- 非常に良い教材や書籍がいろいろある

悪い点

- 趣味としてのプログラミングのハードルがかなり上がったように感じてしまう
- 非常にたくさんの知識やスキルが必要でたくさんのことを勉強する必要がある
- 知識やスキルを習得してもすぐ古びてしまうため、ずっと新しい知識やスキルの勉強をし続けなくてはならない
- やれることが多すぎて何をやれば良いのか、何ができるのかの感覚が不確かになりがちである

自習のススメ (論理的思考とは？)

1. lightbot

demo版をとりあえずクリアしてみる

http://lightbot.com/hoc.html

2. Hour of Code (コードの時間)

1時間でできるらしいので下記「初めてのコンピュータープログラムを書く」をぜひ。講師ビデオにはビル・ゲイツやマーク・ザッカーバーグも登場。lightbot同様、ゲーム感覚のもの。

https://hourofcode.com/code

大雑把なビジョン

- この演習でやりたいことは大きく２つ

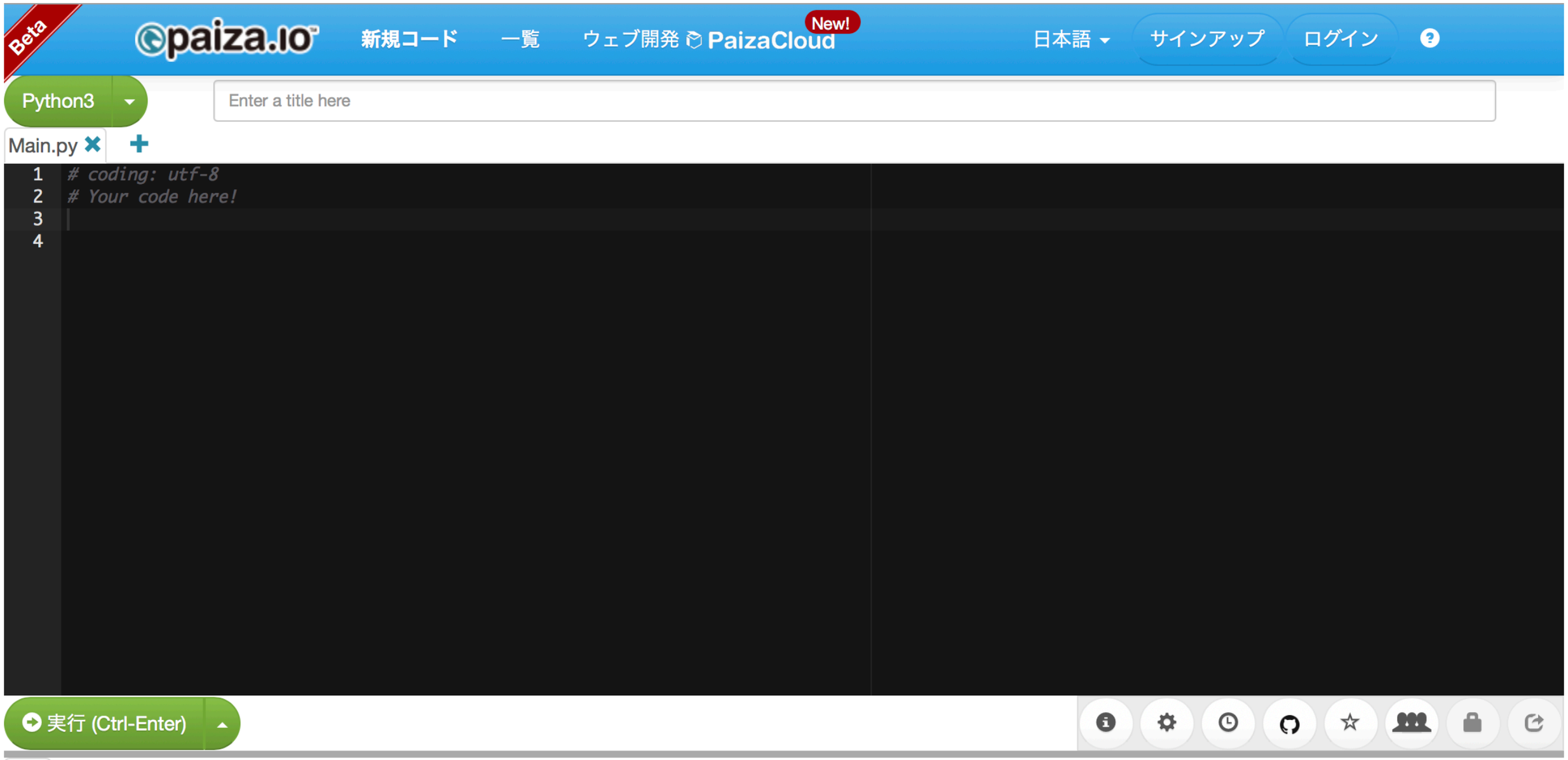
1. 「プログラミング」とは何かを理解しそれをどうやって学んでいけば良いかの指針を与える

2. 「Python」というプログラミング言語の基礎と実習を実際にやってみる

Step その1: paiza.IO

https://paiza.io/ja/


Beta
paiza.io
新規コード
一覧
ウェブ開発
PaizaCloud
New!
日本語
サインアップ
ログイン
Python3
Enter a title here
Bash
C
C#
C++
Clojure
Cobol
CoffeeScript
D
Elixir
Erlang
F#
Go
Haskell
Java
JavaScript
Kotlin
MySQL
Objective-C
Perl
PHP
Python2
Python3
R
Ruby
Rust
Scala
Scheme
Swift
VB
Ruby on Rails, Node.js, MySQL, Java(Tomcat, JSP), PHP(LAMP), Django, Jupyter Notebook, etc.
Web開発、追加インストール、ルート権限、データベース、ターミナル環境はPaizaCloudへ
PaizaCloud™
実行 (Ctrl-Enter)
入力
Leave a message



黒いところにpythonコードを書いて
「実行」をクリック

まずはprintと電卓

1. 文字を出力

   print('出力したい文字')

   print("Don't do it!") ←'があるときは"でくくる

2. 四則演算

   足し算 + 引き算 - 掛け算 * 割り算 /

   割った余り % べき乗 **

   round(1.2345, 3) ←小数点以下3桁に丸める

   (四捨五入とちょっと違う)

Python3

Enter a title here

Main.py

Success ツイート Share 0

```
1  # coding: utf-8
2  # Your code here!
3  print('テストです')
4  print(1+1)
5  print(1 + 3 * (12-2))
6  print('答えは', 1 * 2 * 3 / 4)
7  # シャープで始めるとその行は無視されるのでメモを書いておきます
8  # 「コメント」と言います。
9
10 # 空白行はOKです
11 print(1.34 * 2.2 / 7.99)
```

実行 (Ctrl-Enter)

出力 入力 コメント 0

```
テストです
2
31
答えは 1.5
0.3689612015018774
```

print文をもう少し

```
print('答えは', 3, 'です')
print('答えは', 3, 'です', sep=":")
print('答えは', 3, 'です', sep=":", end=";")
print('答えは', 3, 'です', sep=":", end=";")
print()
print('答えは', 3, 'です', sep=" ", end="\n")
```

バックスラッシュは

日本語キーボードでは「¥」

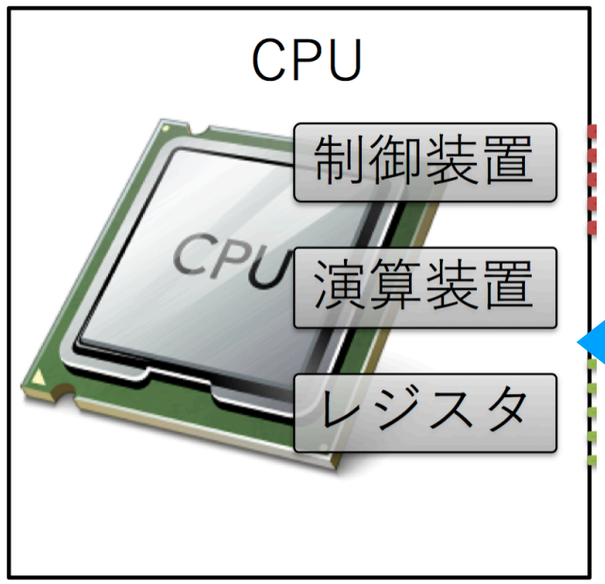
変数と代入

```
a = 3
b = 4
hoge = 5
kotae = a * b + hoge
print('答えは', kotae, 'です', sep='')
```

バックスラッシュは 

日本語キーボードでは「¥」

Step その2: python.exe



特定のコマンド(命令)

しか解さない

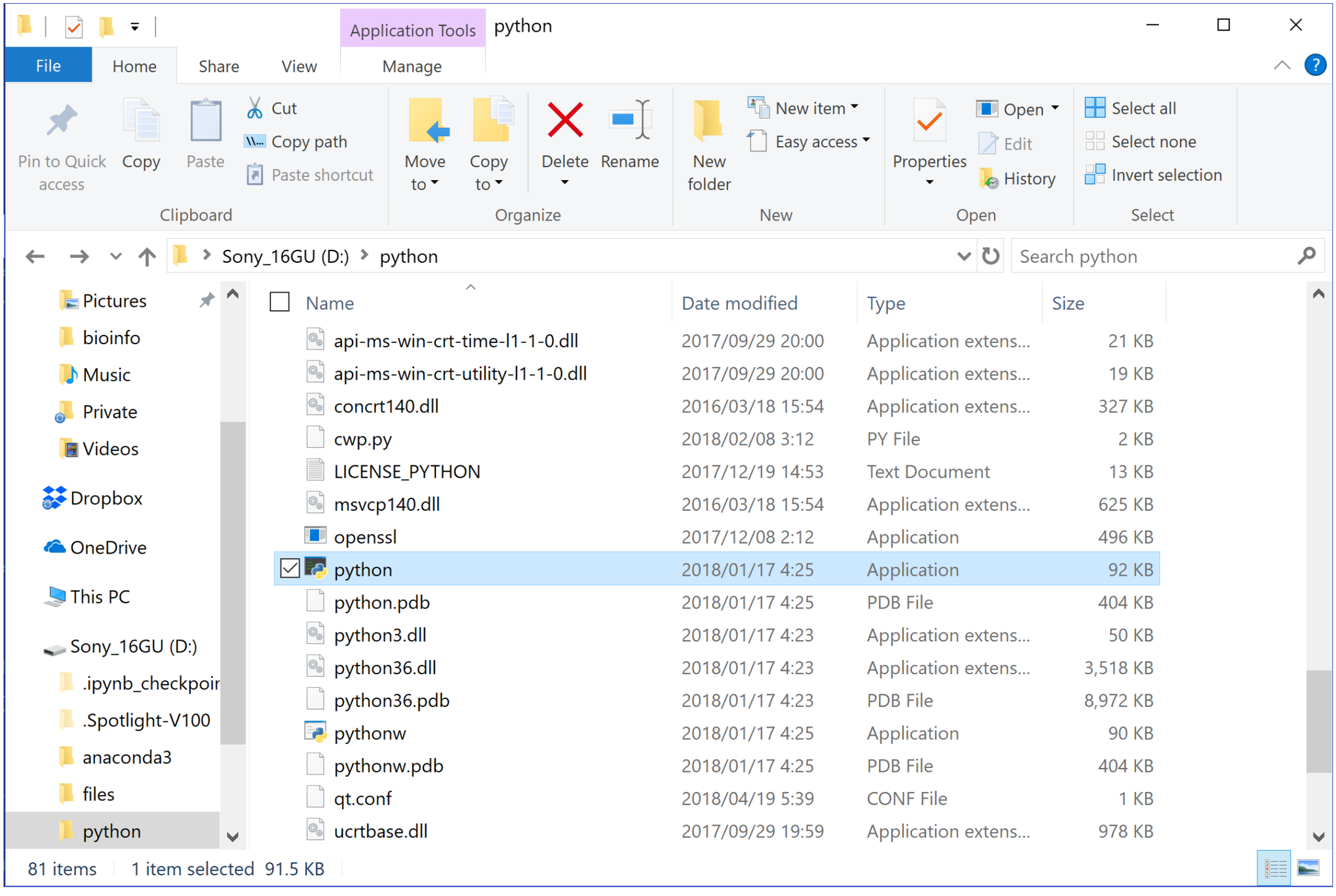
(python言語)

```
a = 3
b = 4
hoge = 5
kotae = a * b + hoge
print(kotae)
```

翻訳

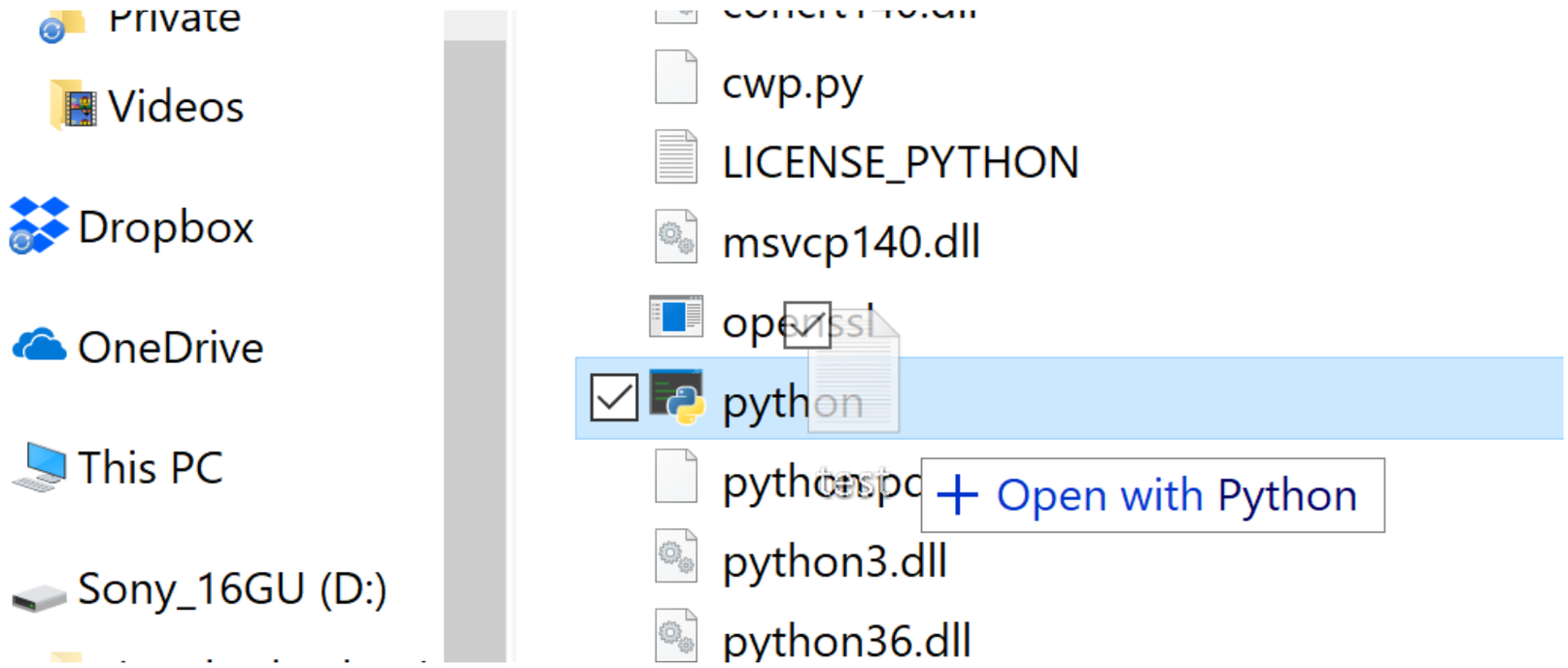
この翻訳プログラムを
行うプログラム
(インタープリタ)

Step その2: python.exe



http://art.ist.hokudai.ac.jp/~takigawa/prog/

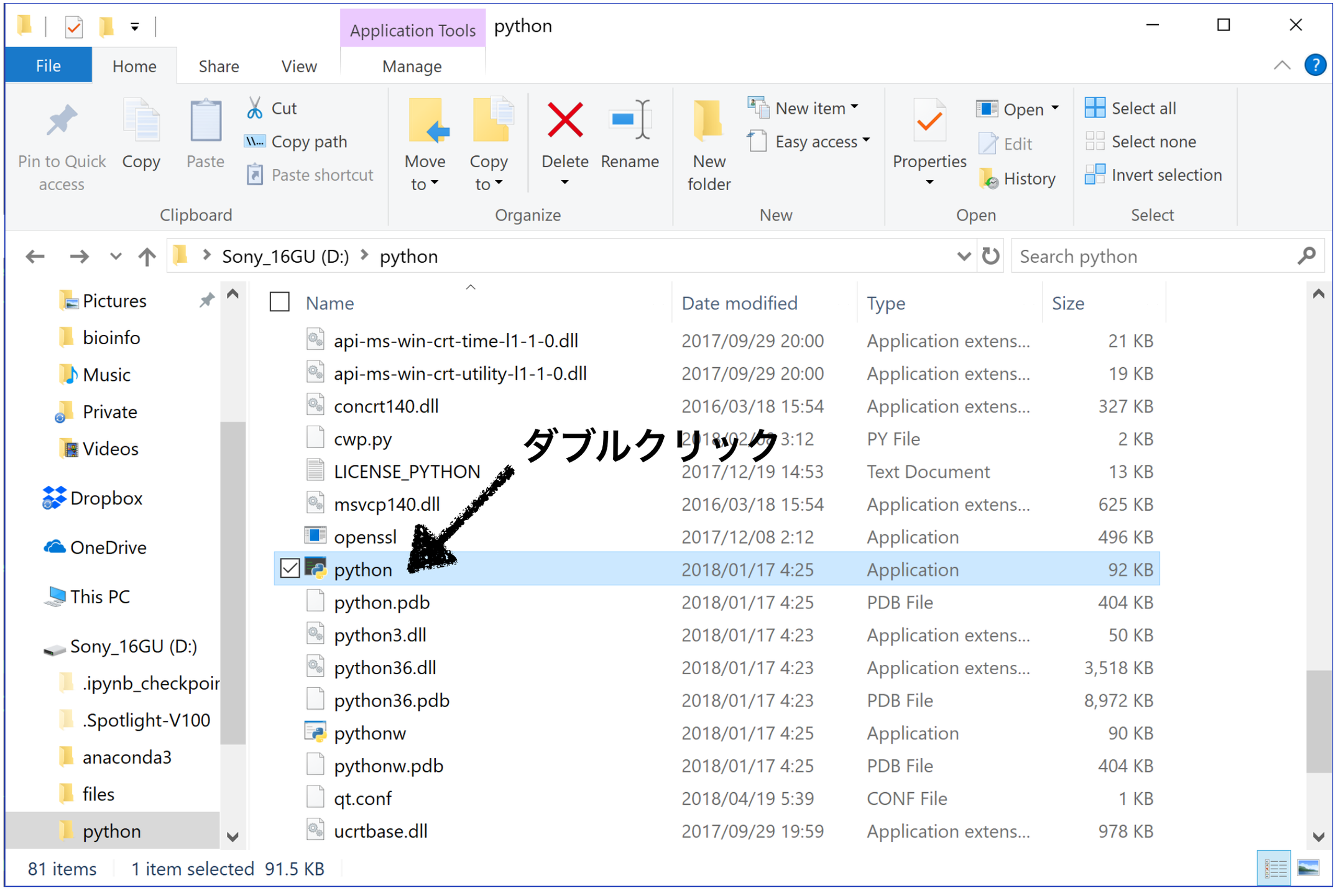
1. 講義ウェブサイトから「test.txt」を
   ダウンロード
2. そのファイルを「python」の上に
   ドラッグアンドドロップ



3.「test.txt」をダブルクリックして中身を見てみる

```
import tkinter.messagebox as mb

ans = mb.askyesno('質問', '今日は朝ごはんを食べましたか？')


if ans == True:
    mb.showinfo('メッセージ', 'ナイス朝ごはん！')
else:
    mb.showinfo('メッセージ', 'ノー朝ごはん！')
```

Step その3:対話実行モード(PEPL)



D:\python\python.exe

```
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
>>> print('こんにちは')
こんにちは
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

1 + 1

print('こんにちは')

import this

Step その2: python.exe
Step その3:対話実行モード(PEPL)

- 「test.txt」をコピーしてそれをダブルクリックし、中身を書き換えて、それをpythonへドラッグ & ドロップ
- 対話実行モードで1行づつ実行

いずれにせよ翻訳プログラム「python.exe」がPython言語で書かれたテキストを機械語に翻訳して実行

ここで大事な点：エラー

ちゃんとPython言語のルールに乗っ取っていない
テキストは翻訳に失敗するのでエラーとなる

```
>>> print('hoge'hogeho')
  File "<stdin>", line 1
    print('hoge'hogeho')
                ^
SyntaxError: invalid syntax
>>>
```

この辺が構文エラー
('の閉じ方が合ってない)

エラーの文章をよく読み、原因を探り修正する
この作業を**「デバッグ(debug)」**と呼ぶ。
エラーのことを**「バグ(bug)」**と呼ぶ。

ここで大事な点：エラー

このエラーはただ翻訳に失敗しているだけで
何もコンピュータに害はないので、最初はどんどん
適当にサンプルや例を書き換えてどうなるか
エラーを出させながら探るのが良い

コーディング時は「エラーを恐れない！」

演習環境：Jupyter Notebook

1 お絵かきしてみよう！

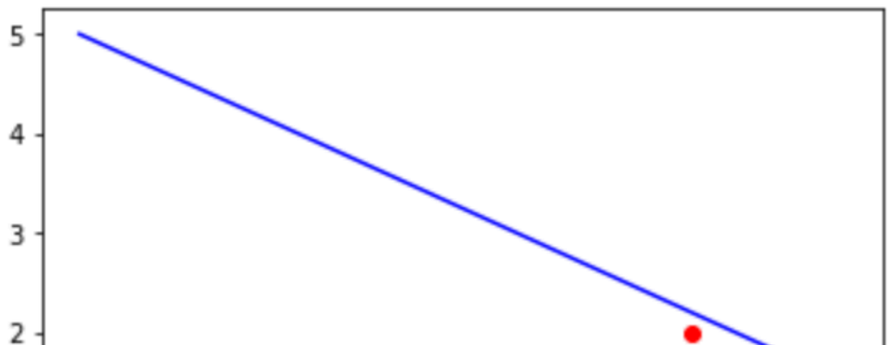
下のセルを選択して、CtrlキーとEnterキーを同時におしてください。

In [40]:
```
import matplotlib.pyplot as plt
%matplotlib inline

def line(p1, p2, c='blue'):
    plt.plot([p1[0], p2[0]], [p1[1], p2[1]], color=c)

def point(p, c='red'):
    plt.plot(p[0], p[1], 'o', color=c)
```

上はふたつの命令「line」と「point」を定義しました。 これを使ってお絵かきしてみましょう。

下記のセルを自由に編集して命令を並べてください。実行はCtrlキーとEnterキーの同時押しか、上部のRunアイコンで！

In [45]:
```
line([0, 0], [1, 1])
line([1, 1], [2, 1.5])
line([2, 1.5],[-3, 5])
point([0, 1])
point([1, 2])
```



USBメモリの環境の動かし方(改)

前提：USBメモリにAnacondaがきちんとインストールできていること(必要なファイルが全部あること)

- できてない人は後で個別対応します
- インストールできない→フォーマットを確認
- インストールできない→容量を確認
- 遅い→USBの規格を確認

USBメモリについて

正確には8G以上でUSB3.0以上のメモリが推奨
(Anaconda3だけで5Gくらいになります)

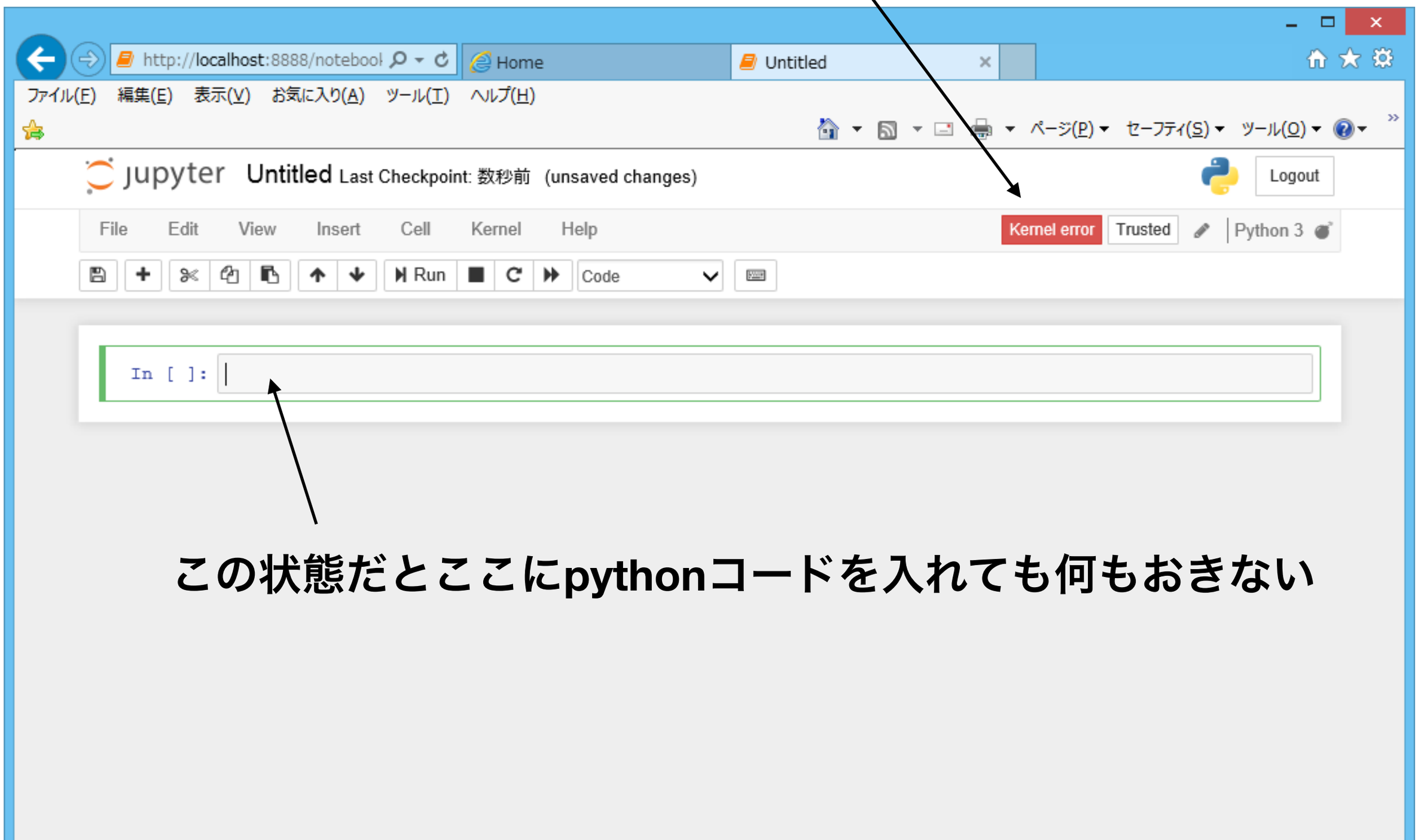
ソケットの色・端子の数が異なる。

1 B(1バイト)

= 8 bit(8ビット)

USB Standard	Max Transfer Speed	Power Output	Logo	Symbol
USB 2.0	480 Mbit/s	2.5W		
USB 3.0 (USB 3.1 Gen 1)	5 Gbit/s	4.5W		
USB 3.1 (USB 3.1 Gen 2)	10 Gbit/s	100W		

jupyter-notebook.exeを起動すると「Kernel error」が表示されてしまう



この状態だとここにpythonコードを入れても何もおきない

理由：

この演習室のPCのユーザ情報は毎回ログアウトすると消えてしまうが Jupyter Notebookはユーザ情報の領域にpython.exeの呼び出しのコピーを作ってそれを使う（毎回USBからだと遅いし）

解決策：毎回演習開始時に以下を実行

1. USBのフォルダを選択
2. コマンドプロンプトを起動
3. 「python -m ipykernel install --user」を入力

Sony_16GU (F:)

ドライブ ツール

ファイル　ホーム　共有　表示　管理

コピー　貼り付け　切り取り　パスのコピー　ショートカットの貼り付け　（クリップボード）

移動先　コピー先　削除　名前の変更　（整理）

新しいフォルダー　新しい項目　ショートカット　（新規）

プロパティ　開く　編集　履歴　（開く）

すべて選択　選択解除　選択の切り替え　（選択）

PC ▸ Sony_16GU (F:) ▸

Sony_16GU (F:)の検索

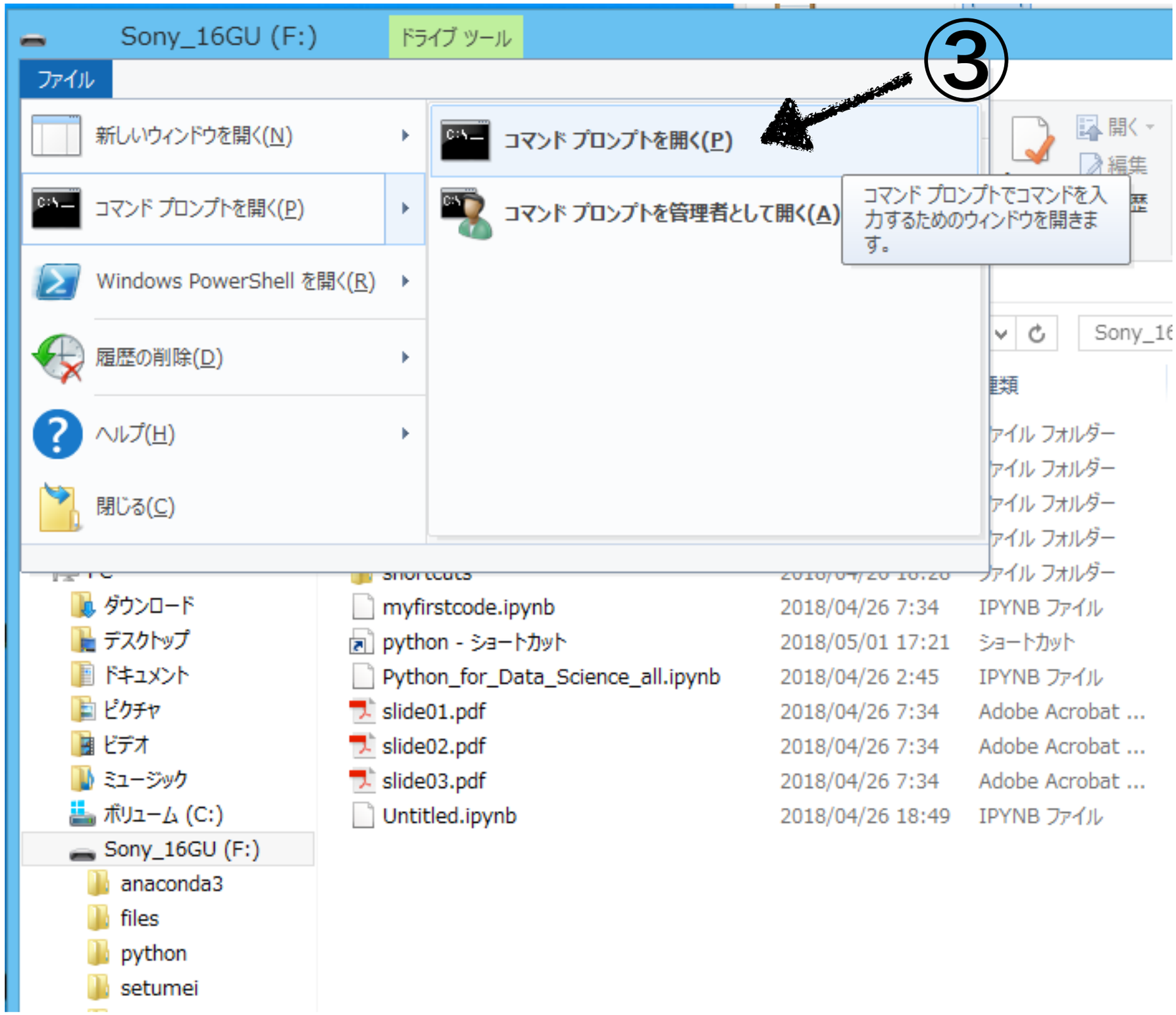
②

名前	更新日時	種類	サイズ
anaconda3	2018/04/26 19:36	ファイル フォルダー	
files	2018/04/19 7:25	ファイル フォルダー	
python	2018/04/19 4:45	ファイル フォルダー	
setumei	2018/05/08 19:02	ファイル フォルダー	
shortcuts	2018/04/26 18:28	ファイル フォルダー	
myfirstcode.ipynb	2018/04/26 7:34	IPYNB ファイル	5 KB
python - ショートカット	2018/05/01 17:21	ショートカット	1 KB
Python_for_Data_Science_all.ipynb	2018/04/26 2:45	IPYNB ファイル	219 KB
slide01.pdf	2018/04/26 7:34	Adobe Acrobat ...	36,553 KB
slide02.pdf	2018/04/26 7:34	Adobe Acrobat ...	13,253 KB
slide03.pdf	2018/04/26 7:34	Adobe Acrobat ...	8,579 KB
Untitled.ipynb	2018/04/26 18:49	IPYNB ファイル	1 KB

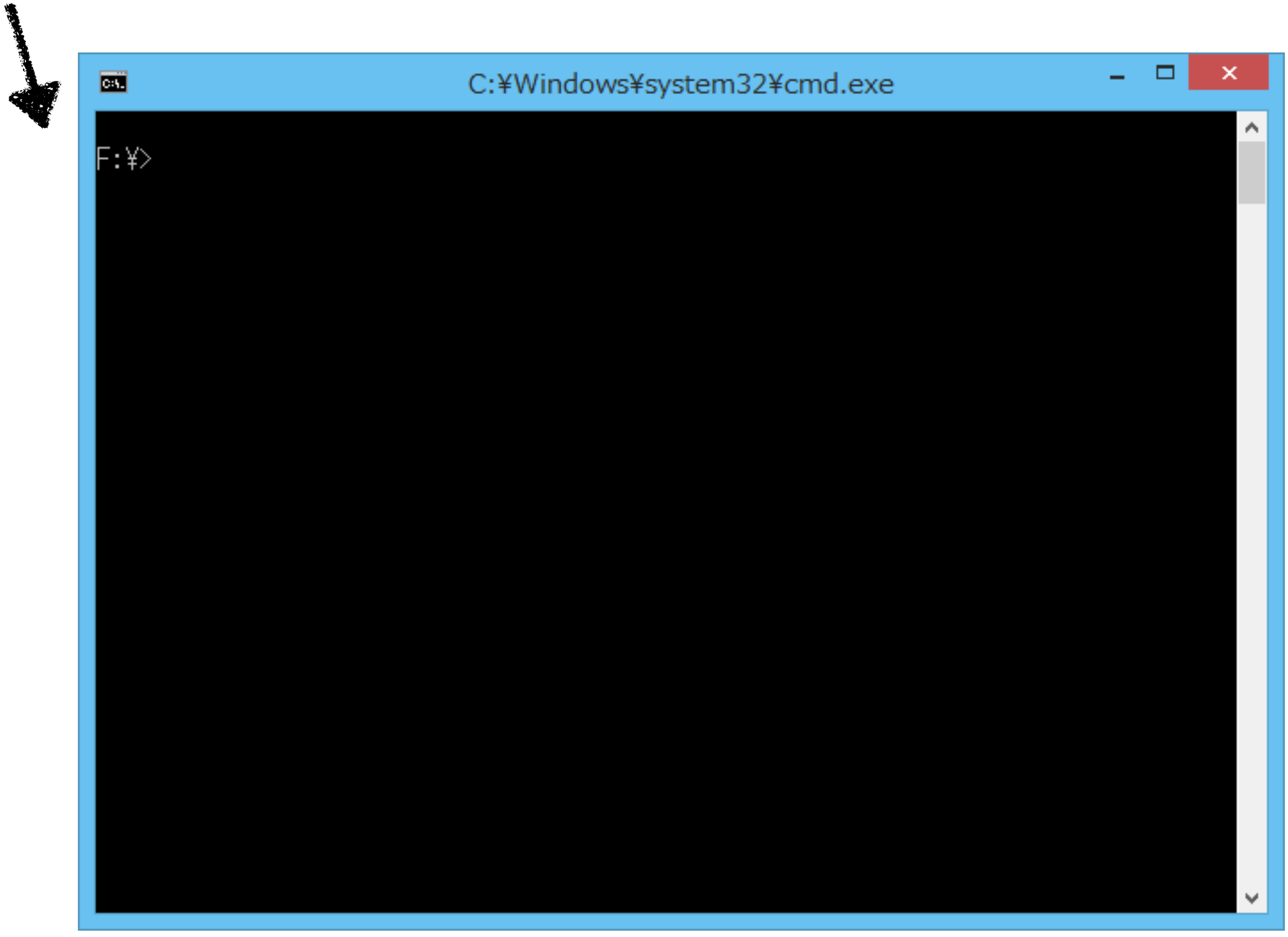
①

- お気に入り
  - ダウンロード
  - デスクトップ
  - 最近表示した場所
- PC
  - ダウンロード
  - デスクトップ
  - ドキュメント
  - ピクチャ
  - ビデオ
  - ミュージック
  - ボリューム (C:)
  - Sony_16GU (F:)
    - anaconda3
    - files
    - python
    - setumei
    - shortcuts
- ネットワーク

12 個の項目


Sony_16GU (F:)
ドライブ ツール
ファイル
新しいウィンドウを開く(N)
コマンド プロンプトを開く(P)
Windows PowerShell を開く(R)
履歴の削除(D)
ヘルプ(H)
閉じる(C)
コマンド プロンプトを開く(P)
コマンド プロンプトを管理者として開く(A)
コマンド プロンプトでコマンドを入力するためのウィンドウを開きます。
③
開く
編集
ダウンロード
デスクトップ
ドキュメント
ピクチャ
ビデオ
ミュージック
ボリューム (C:)
Sony_16GU (F:)
anaconda3
files
python
setumei
myfirstcode.ipynb 2018/04/26 7:34 IPYNB ファイル
python - ショートカット 2018/05/01 17:21 ショートカット
Python_for_Data_Science_all.ipynb 2018/04/26 2:45 IPYNB ファイル
slide01.pdf 2018/04/26 7:34 Adobe Acrobat ...
slide02.pdf 2018/04/26 7:34 Adobe Acrobat ...
slide03.pdf 2018/04/26 7:34 Adobe Acrobat ...
Untitled.ipynb 2018/04/26 18:49 IPYNB ファイル

演習室ではここがF:になっていると思いますが
家のPCではE:だったり、D:だったりかも



C:¥Windows¥system32¥cmd.exe

```
F:¥>cd python

F:¥python>python -m ipykernel install --user
Installed kernelspec python3 in C:¥Temporary¥AppData¥Roaming¥jupyter¥kernels¥pyt
hon3

F:¥python>_
```

理由：

この演習室のPCのユーザ情報は毎回ログアウトすると消えてしまうが Jupyter Notebookはユーザ情報の領域にpython.exeの呼び出しのコピーを作ってそれを使う（毎回USBからだと遅いし）

解決策：毎回演習開始時に以下を実行

1. USBのフォルダを選択
2. コマンドプロンプトを起動
3. 「python -m ipykernel install --user」を入力

http://art.ist.hokudai.ac.jp/~takigawa/prog/

講義ウェブサイトから「Jupyter_Notebook.lnk」をダウンロードし、USBに保存しておく

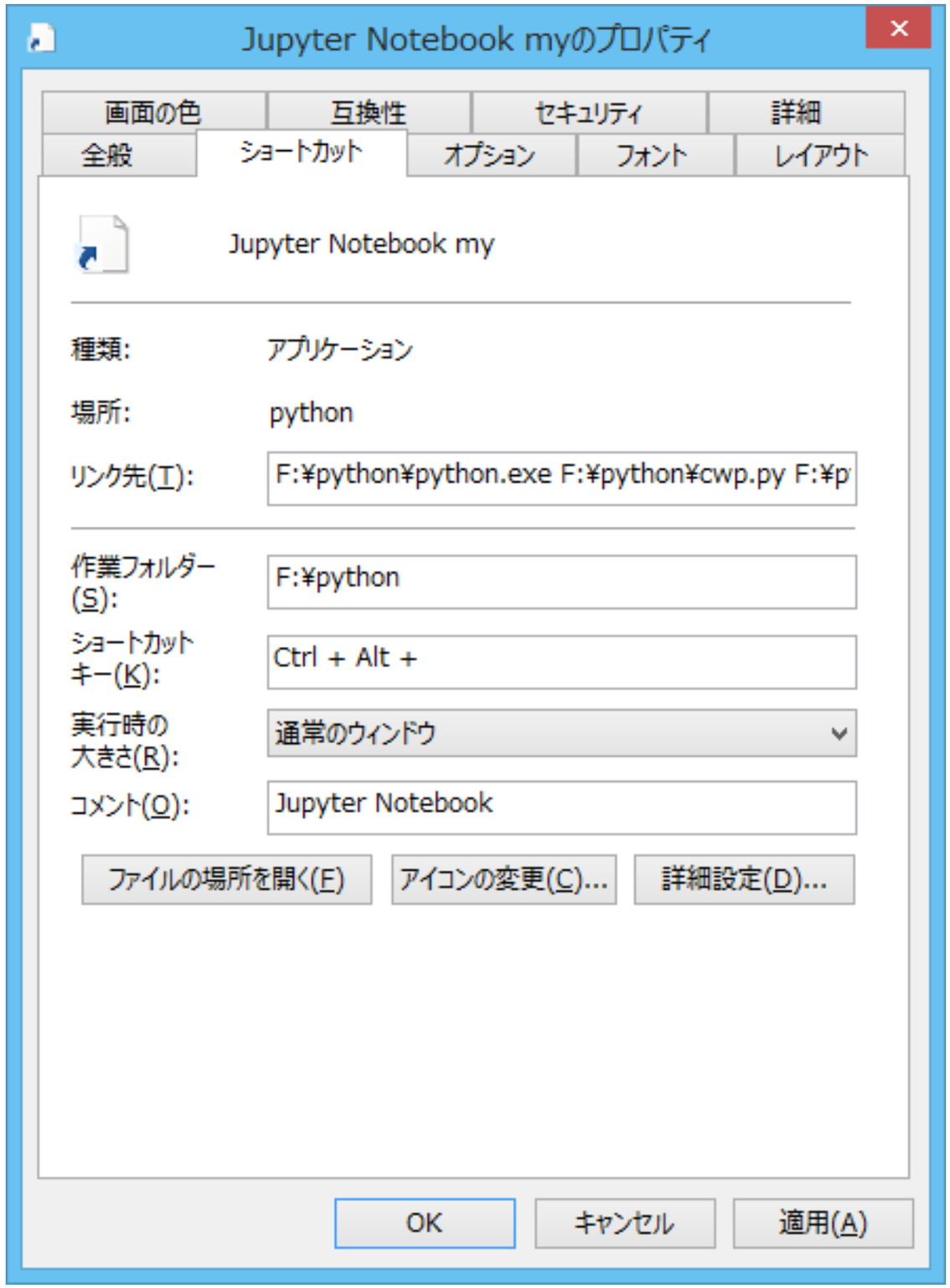
毎回講義開始時に先ほどの作業をした後、USBからこのファイルをデスクトップにコピーしダブルクリックすれば「Jupyter Notebook」が起動する

ただし先ほどの表示が「F:」で始まっていること、USBにanacondaを入れたフォルダ名が「python」であること、は必要

もし先ほどの作業をやっているのに起動しない場合：

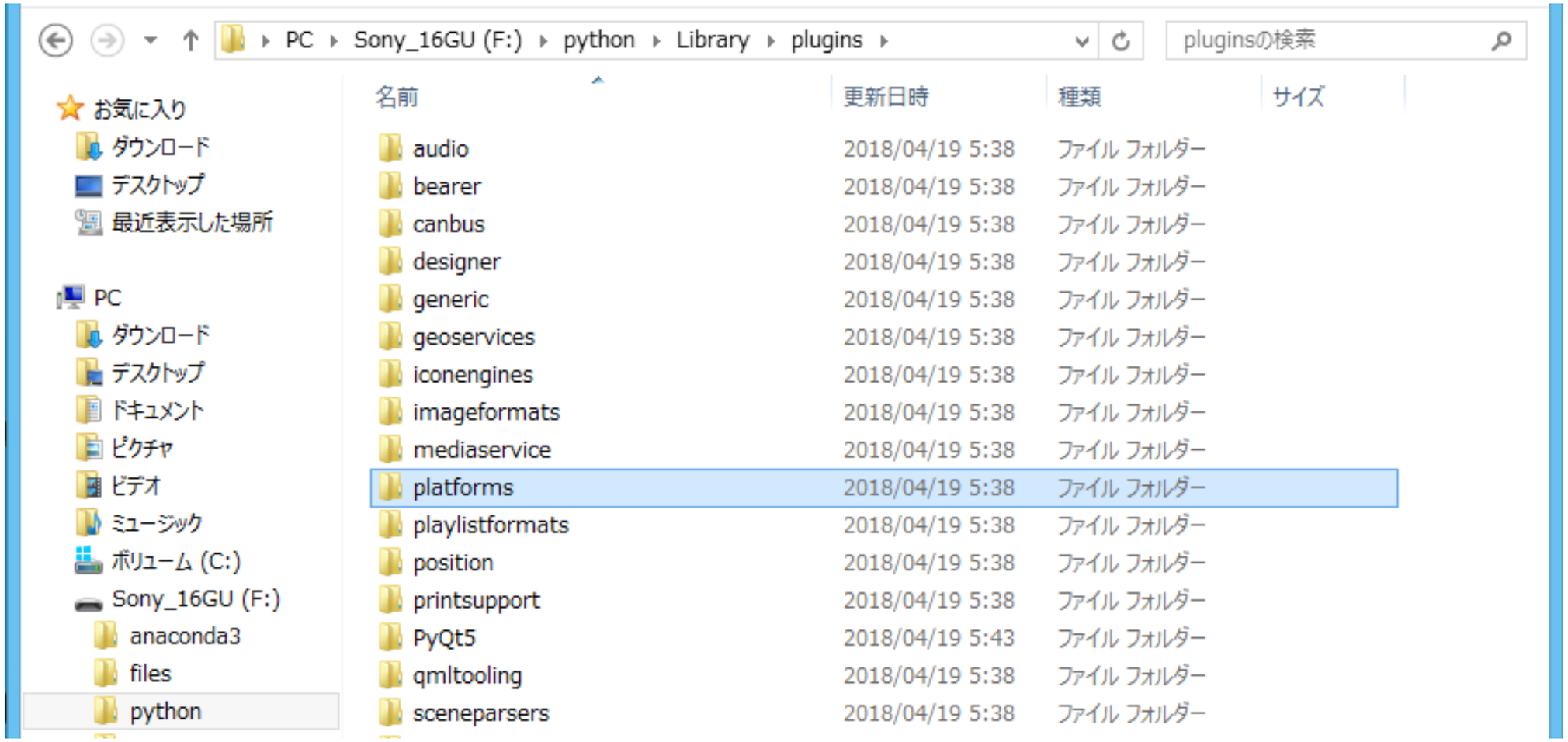
Jupyter_Notebook.lnkのアイコンを右クリックし、プロパティを選択

ショートカットを選び
右のような画面が出たら
「リンク先」の部分の
「F:\python」を**すべて**
自分の環境と同じに修正



Anaconda Navigatorが起動しない件

これもユーザ情報の領域へコピーされる情報の問題で、解決策としては
AnacondaをインストールしたフォルダにLibary→pluginで見える
「platforms」というフォルダを丸ごとコピーする。これはUSBフォル
ダ内なので一回やればOK



PC
ダウンロード
デスクトップ
ドキュメント
ピクチャ
ビデオ
ミュージック
ボリューム (C:)
Sony_16GU (F:)
anaconda3
files
python
setumei
shortcuts
ネットワーク

generic 2018/04/19 5:38 ファイル
geoservices 2018/04/19 5:38 ファイル
iconengines 2018/04/19 5:38 ファイル
imageformats 2018/04/19 5:38 ファイル
mediaservice 2018/04/19 5:38 ファイル
platforms
playlistformats
position
printsupport
PyQt5
qmltooling
sceneparsers
sensorgestures
sensors
sqldrivers

19 個の項目　1 個の項目を選択

開く(O)
新しいウィンドウで開く(E)
VLCメディアプレイヤーのプレイリストに追加
Browse in Adobe Bridge CC
VLCメディアプレイヤーで再生
共有(H)
解凍(X)
圧縮(U)
ライブラリに追加(I)
スタートにピン留め(P)
送る(N)
切り取り(T)
コピー(C)
ショートカットの作成(S)
削除(D)
名前の変更(M)
プロパティ(R)

コピー　貼り付け　パスのコピー
新しいフォルダー

展開(A)
新しいウィンドウで開く(E)
VLCメディアプレイヤーのプレイリストに追加
Browse in Adobe Bridge CC
VLCメディアプレイヤーで再生
共有(H)
解凍(X)
圧縮(U)
ライブラリに追加(I)
スタートにピン留め(P)
送る(N)
切り取り(T)
コピー(C)
貼り付け(P)
削除(D)
名前の変更(M)
新規作成(X)
プロパティ(R)

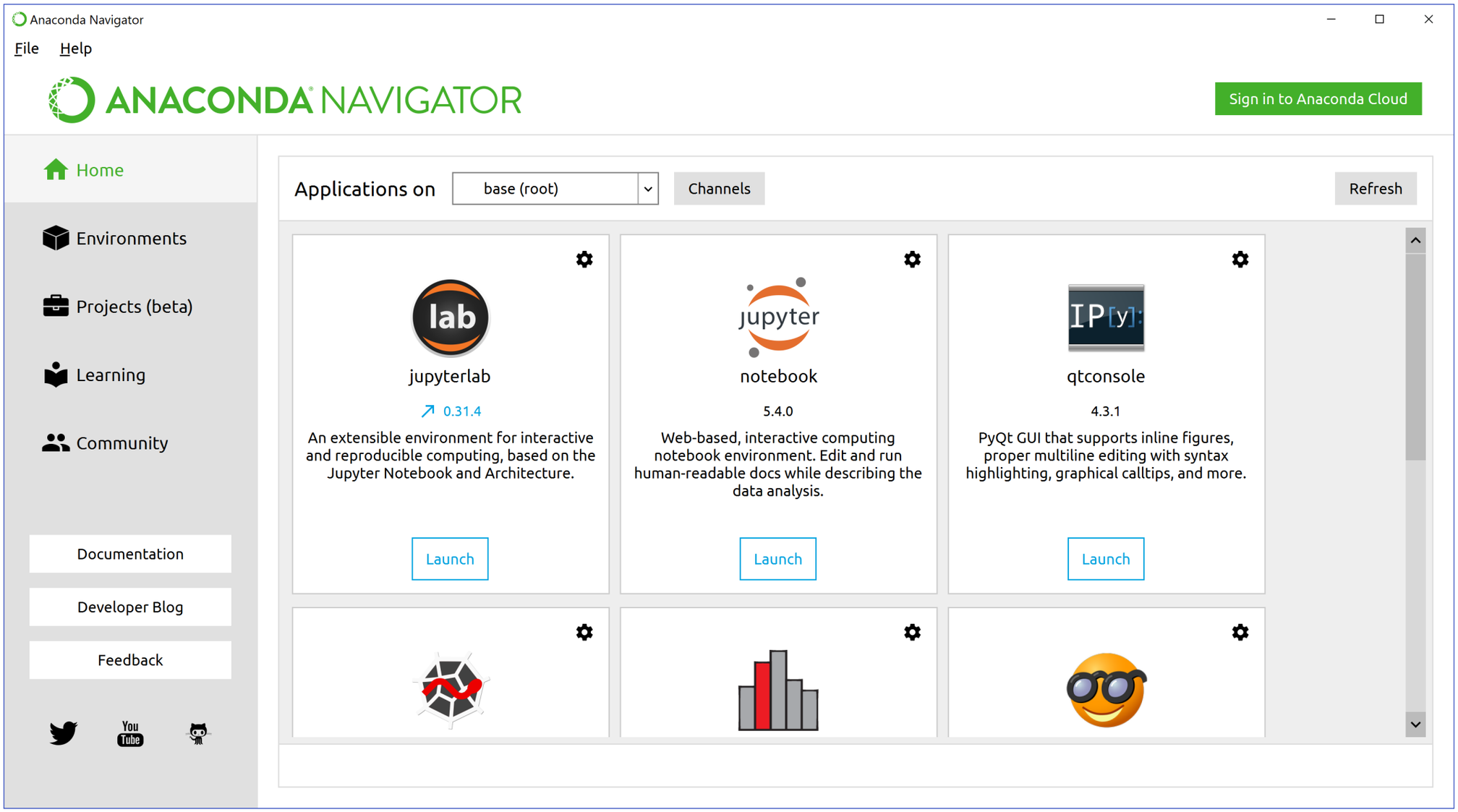
お気に入り
ダウンロード
デスクトップ
最近表示した場所
PC
ダウンロード
デスクトップ
ドキュメント
ピクチャ
ビデオ
ミュージック
ボリューム (C:)
Sony_16GU
anaconda3
files
python
setumei
shortcuts
ネットワーク

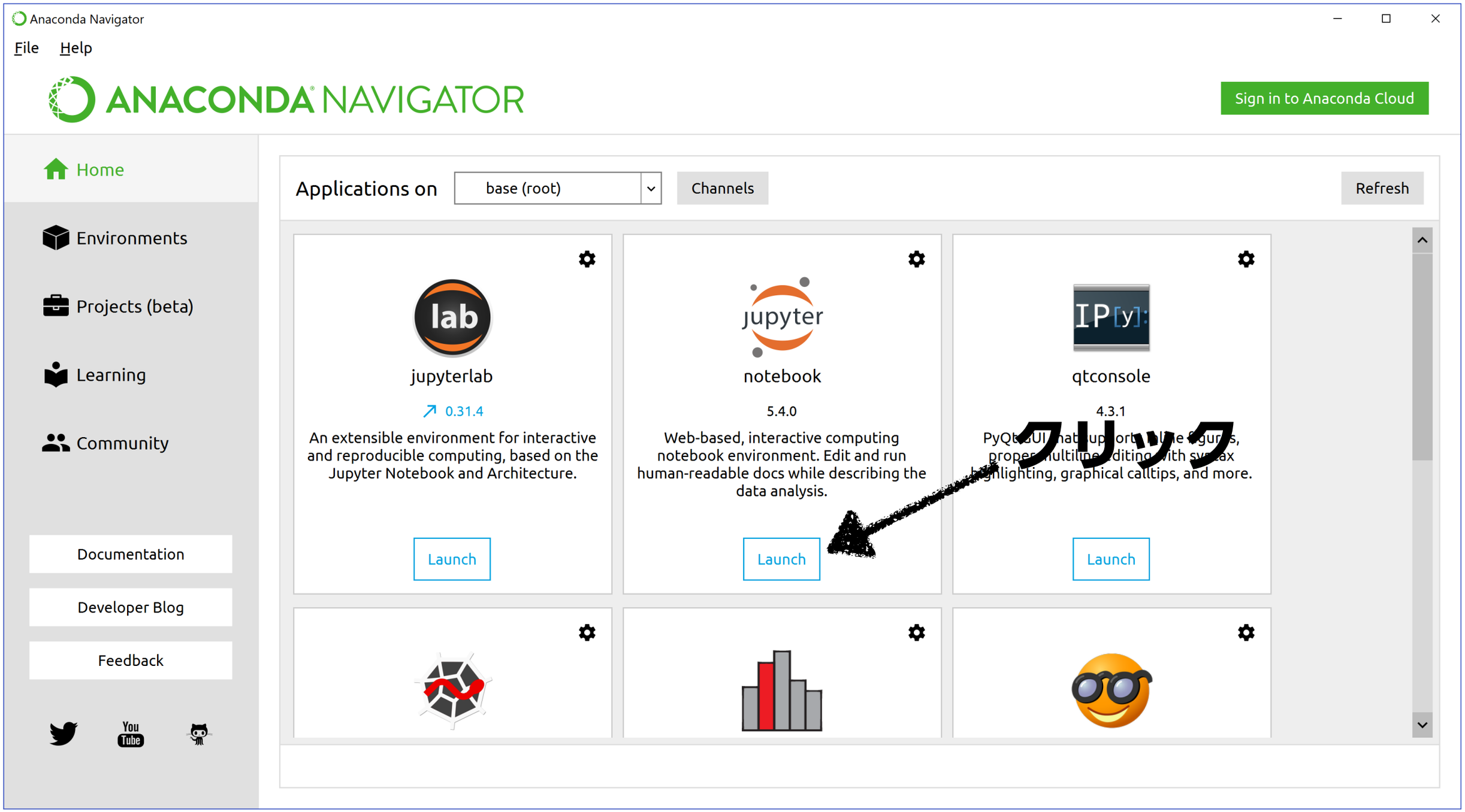
更新日時
2016/03/1
2018/02/0
2017/12/1
2016/03/1
2017/12/0
2018/01/1
2018/01/1
2018/01/1
2018/01/1
2018/01/1
2018/01/1
2018/04/1
2017/09/2
2018/04/1
2018/04/2
Untitled.ipynb
vccorlib140.dll
vcomp140.dll
vcruntime140.dll
vscode_eula.txt
xlwings32-0.11.5.dll
xlwings64-0.11.5.dll
2016/03/1
2016/03/1
2016/03/1
2018/02/1
2018/01/0
2018/01/0

80 個の項目

セットアップしたPython環境を起動してみる

インストールしたフォルダの中の「Scripts」の中にある anaconda-navigatorをダブルクリック





注：なお時期versionではJupyterLabに移行なのでそっちでもいいかも

Home

localhost:8888/tree/Desktop/mypython

jupyter

Logout

Files | Running | Clusters

Select items to perform actions on them.

Upload | New | Refresh

0 / Desktop / mypython

Name | Last Modified

.. 数秒前

The notebook list is empty.

見たいフォルダを指定できる

新しいフォルダを作成したりもできる

Home

localhost:8888/tree/Desktop/mypython

jupyter

Logout

Files | Running | Clusters

Select items to perform actions on them.

Upload | New

0 / Desktop / mypython

..

The notebook list is empty.

Notebook:

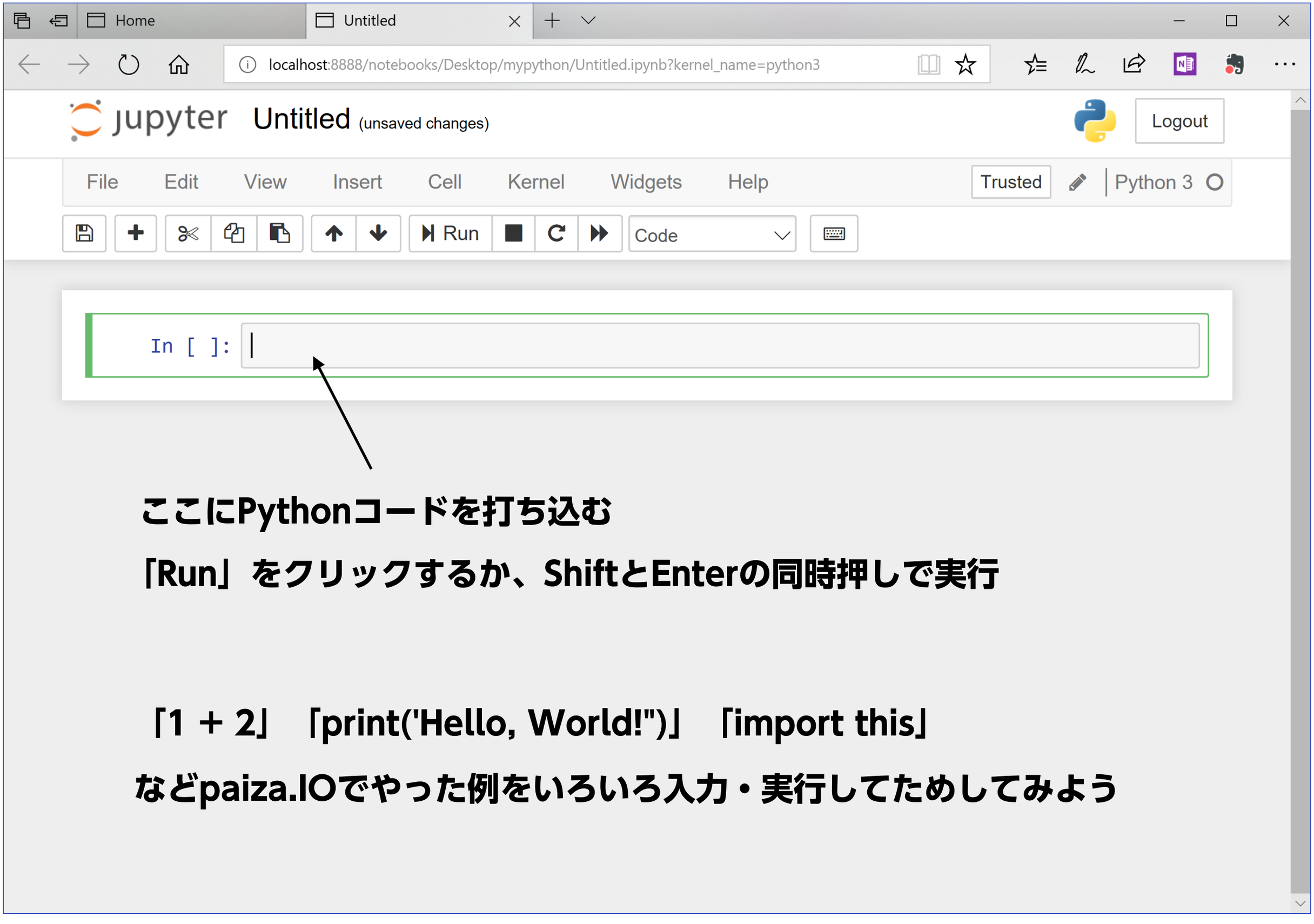
Python 3

Other:

Text File

Folder

Terminal



ここにPythonコードを打ち込む

「Run」をクリックするか、ShiftとEnterの同時押しで実行

「1 + 2」 「print('Hello, World!")」 「import this」

などpaiza.IOでやった例をいろいろ入力・実行してためしてみよう

作業：授業のホームページから「myfirstcode.ipynb」や「ex1.ipynb」をダウンロードして実行してみよう。

http://art.ist.hokudai.ac.jp/~takigawa/prog/

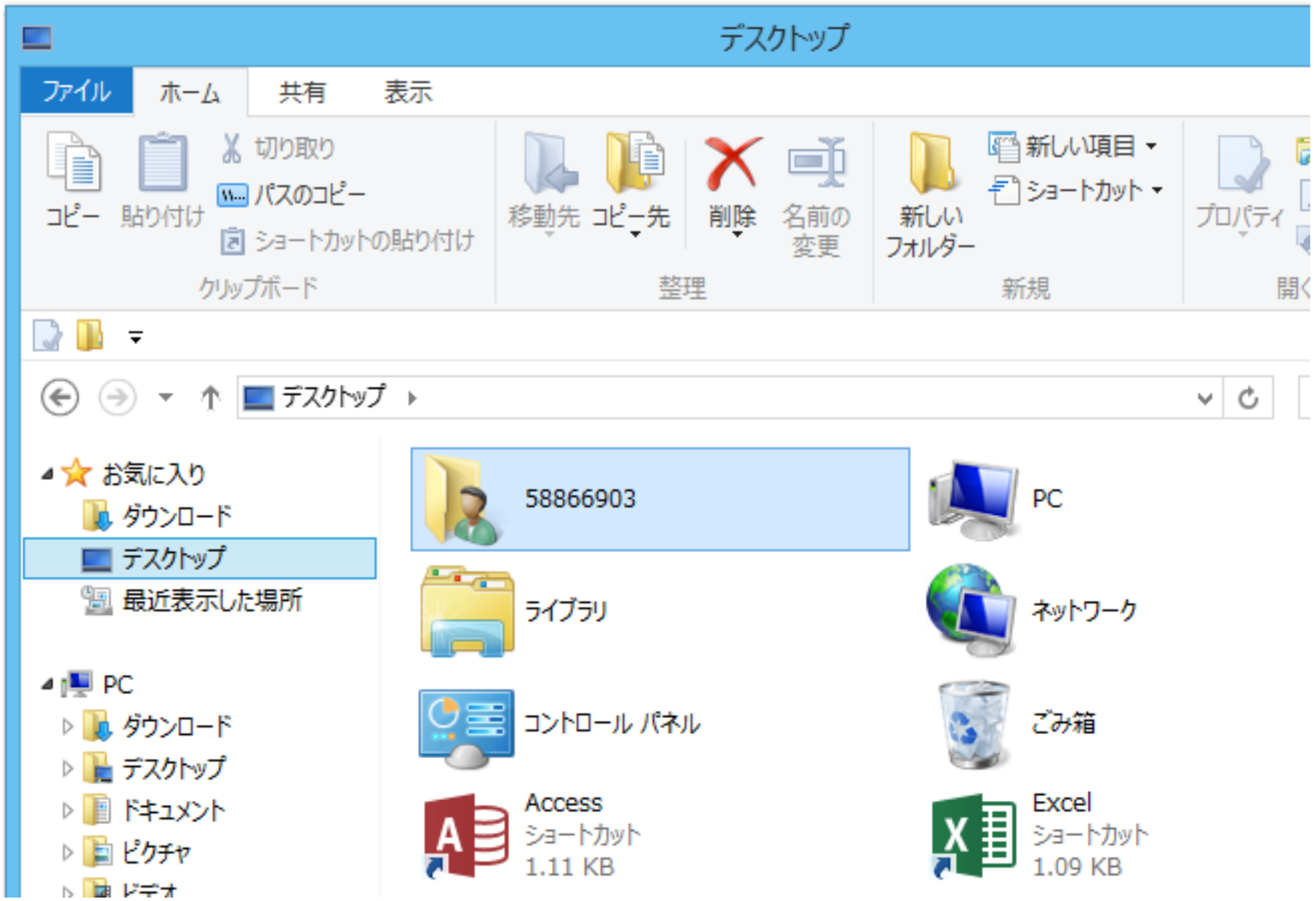
Pythonが起動しているフォルダにコピーし、

クリックして開く！

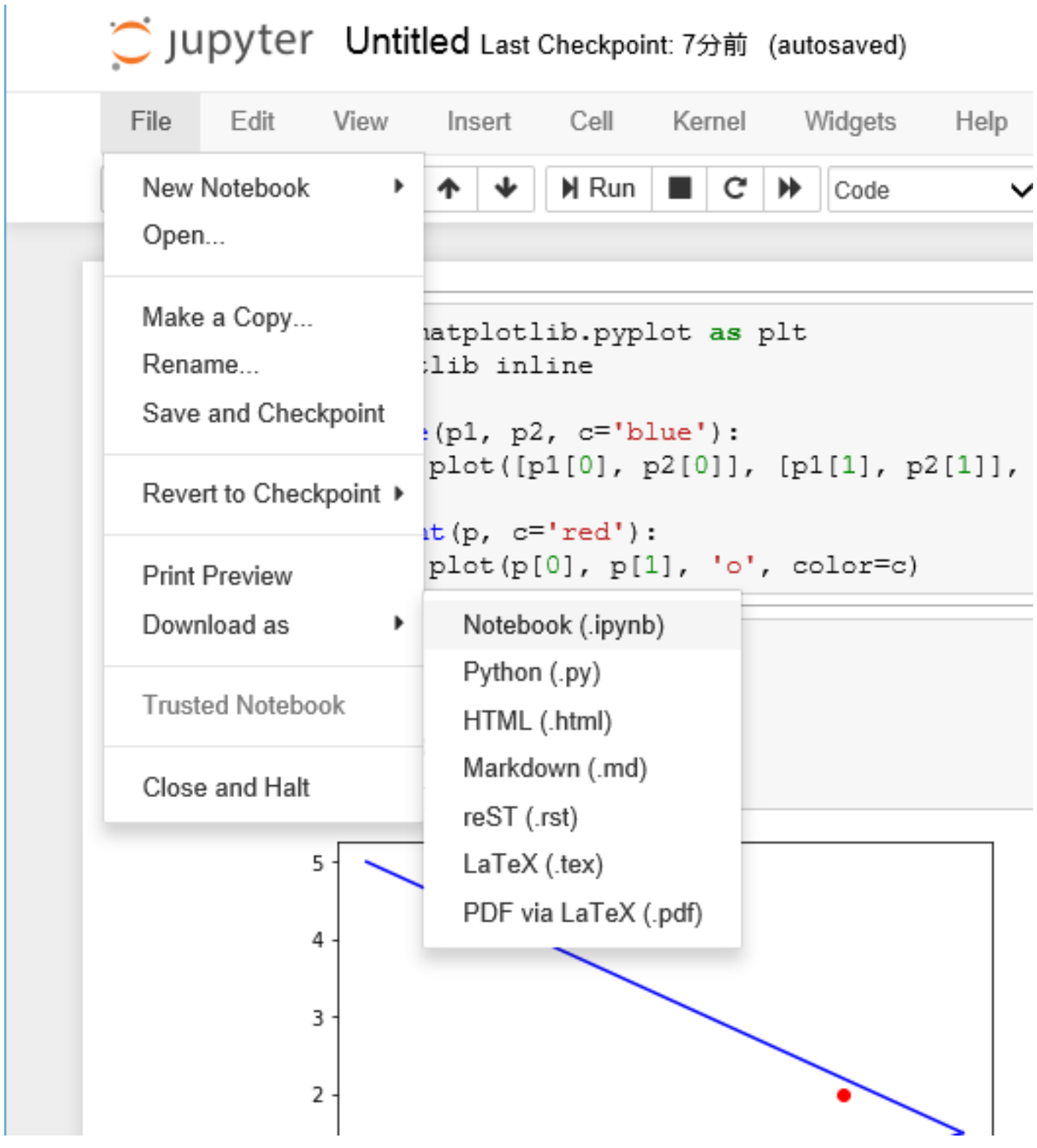
「Pythonが起動しているフォルダ」

Anacondaを入れたフォルダのScriptsのjupyter-notebookを実行して起動した場合 → そのScriptsフォルダ

先ほどの「Jupyter_Notebook.lnk」から起動した場合



実行できたらそのJupyter Notebookを保存してみよう！

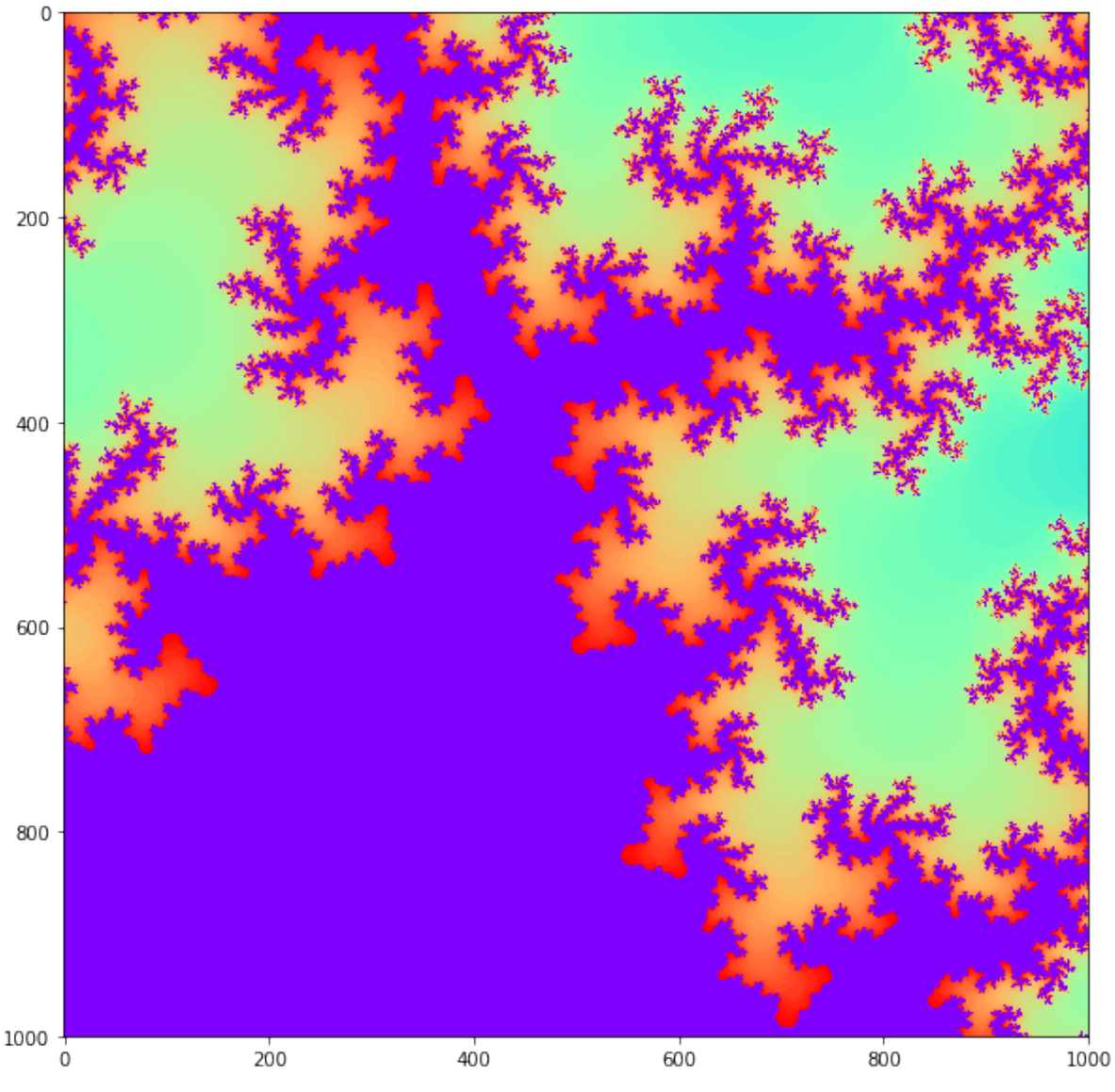


Jupyter Notebookが起動し動いて入れば、この環境でかなり高度なこともできます。またネットに転がっているNotebookを実行して見ることもできます!

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

mpl.rcParams['figure.figsize'] = 10, 10
```

```
def mandelbrot(c, maxiter):
    z = c
    for n in range(maxiter):
        if abs(z) > 2:
            return n
        z = z*z + c
    return 0

def mandelbrot_set(xmin, xmax, ymin, ymax, w, h, maxiter):
    a = np.linspace(xmin, xmax, w)
    b = np.linspace(ymin, ymax, h)
    s = np.empty((w, h))
    for i in range(w):
        for j in range(h):
            s[i, j] = mandelbrot(a[i] + 1j*b[j], maxiter)
    return s
```

```
m = mandelbrot_set(-0.56, -0.55, -0.56, -0.55, 1000, 1000, 80)
plt.imshow(m, cmap='rainbow')
```



Pythonの学習について

- Python公式チュートリアル
  https://docs.python.jp/3/tutorial/
- LearnPython.org interactive Python tutorial (英語)
  https://www.learnpython.org
- CodeAcademy (英語)
  https://www.codecademy.com/learn/learn-python
- Intro to Python for Data Science (英語)
  https://www.datacamp.com/courses/intro-to-python-for-data-science/
- Python tutorials for beginners (英語)
  http://thepythonguru.com
- Progate (プロゲート)
  http://prog-8.com/languages/python
- ドットインストール
  https://dotinstall.com/lessons/basic_python_v3

オンラインで無料の良い講義も多数！

https://wiki.python.org/moin/BeginnersGuide/NonProgrammers

edX

Introduction to Python: Absolute Beginner

In this course that's perfect for true beginners, learn Python basics and start coding right away.

Microsoft

Introduction to Python: Fundamentals

Build on what you learned in the "Introduction to Python: Absolute Beginner" course, and dig into data structure basics.

Microsoft

UDACITY

FREE COURSE

Introduction to Python

Starting Out in Python 3

START FREE COURSE

- LearnPython.org interactive Python tutorial (英語)

Indentation

Python uses indentation for blocks, instead of curly braces. Both tabs and spaces are supported, but the standard indentation requires standard Python code to use four spaces. For example:

script.py | IPython Shell

```
1  x = 1
2  if x == 1:
3      # indented four spaces
4      print("x is 1.")
```

Run

Powered by DataCamp

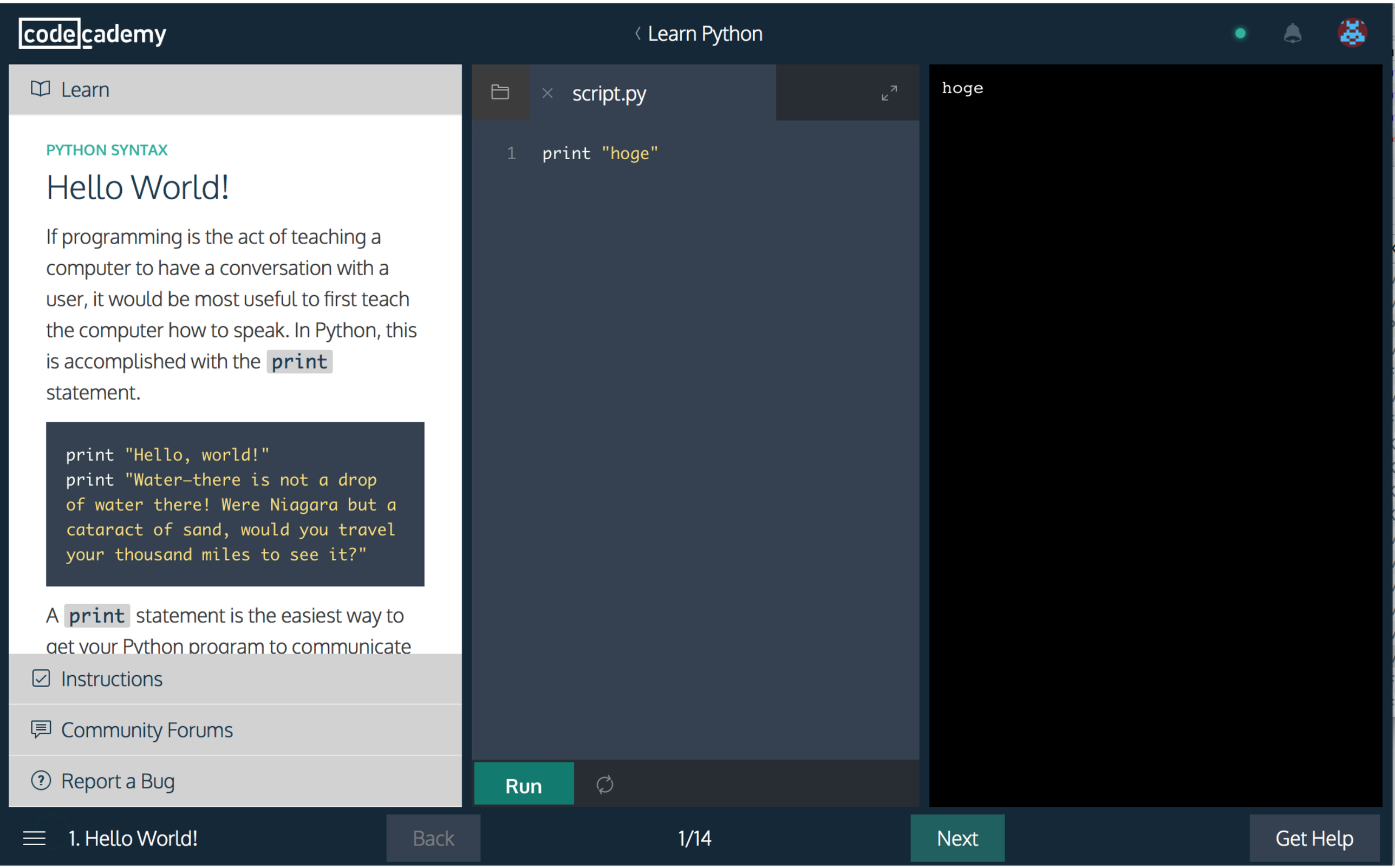
Exercise

Use the "print" command to print the line "Hello, World!".

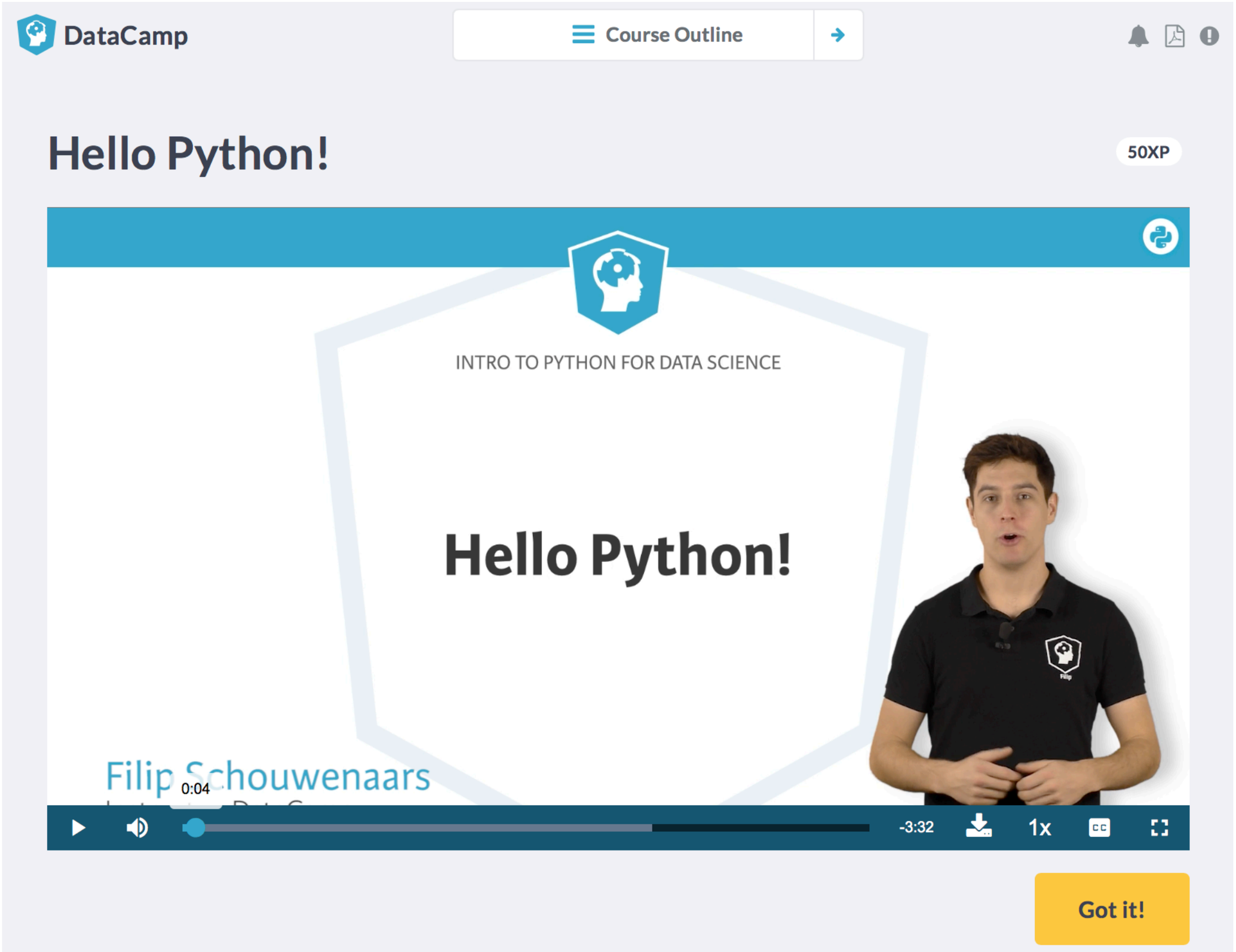
script.py | IPython Shell

```
1  print("Goodbye, World!")
```

- CodeAcademy (英語)



- Intro to Python for Data Science (英語)



- Progate (プロゲート)

文字列

文字列とは？

・文字列　・クォーテーション

先ほどの例で用いた「Hello Python」という文字は、プログラミングの世界では「文字列」と呼ばれます。
文字列はシングルクォーテーション「'」またはダブルクォーテーション「"」で囲む必要があります。どちらで囲んでも出力結果は同じとなります。
どちらかで囲んでいない場合、コードは動かなくなります。

シングルクォーテーション (')
または、ダブルクォーテーション (") で囲む

script.py

```
print('Hello Python')
print("Hello Python")
```

script.py

```
# エラー発生
print(Hello Python)
```

出力結果

コンソール

```
Hello Python
Hello Python
```

出力結果は同じ

コンソール

```
SyntaxError: invalid syntax
```

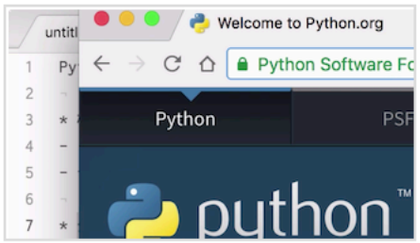
✕ エラー！！：必ず文字列はクォーテーションで囲む！

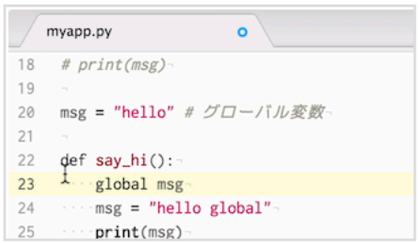
- ドットインストール

Python 3入門 (全31回) PREMIUM

データ解析や機械学習などにも利用される、シンプルなオブジェクト指向型言語であるPythonについて見ていきます。

このレッスンでは **Python 3.5.2** を使用しています。





動画レッスン一覧 (31) | よくある質問一覧 (9)

#01 Pythonを使ってみよう (01:58) 無料公開中 — 未完了

- 概要
- 公式サイト
- 必要となる知識
- レッスンにおける環境

#02 はじめてのPythonプログラム (02:45) 無料公開中 — 未完了

- myapp.pyの作成
- 実行方法
- コメントの書き方

3分動画でマスターする プログラミング学習サイト

初心者向け

すべてのレッスンを見る

ご利用はこちらから

新規登録（無料）

ログイン

Sign in with Google

Python 3入門 、HTML入門 、iPhoneアプリ開発入門 など、**349** レッスンを **5,186** 本の **3分動画** にて提供中

人気のレッスン

» すべてのレッスンを見る (349)

HTML入門 (全15回)
簡単なプロフィールサイトを作りながらHTMLについて学んでいきます。

新着レッスン NEW !!

CSS入門 (全17回)
ウェブページの見た目を整えるための言語であるCSSについて学んでいきます。

HTML入門 (全15回)
簡単なプロフィールサイトを作りながらHTMLについて学んでいきます。

HTML/CSSの学習環境を整えよう [macOS編] (全4回)
初めてHTML/CSS入門を受講する方に向けて、必要なツールや設定について解説していきます。

HTML/CSSの学習環境を整えよう [Windows編] (全4回)

レッスン一覧 349レッスンを5,186本の動画で提供中

目的別で探す　絞り込み検索をする

ホームページを作れるようになろう

ホームページを作るための基礎知識をマスターできるレッスンパックです。まずはここから始めてみましょう。

- HTML/CSSの学習環境を整えよう [Windows編] (全4回)
- HTML/CSSの学習環境を整えよう [macOS編] (全4回)
- HTML入門 (全15回)
- CSS入門 (全17回)
- CSSレイアウト入門 (全15回)
- 実践！ウェブサイトを作ろう (全17回)
- CSSで吹き出しを作ろう (全8回)
- レスポンシブウェブデザイン入門 (全14回)
- JavaScriptでモーダルウィンドウを作ろう (全8回) PREMIUM

JavaScriptから始めるお手軽プログラミング

ブラウザとテキストエディタがあればすぐに始められるJavaScriptをマスターするためのレッスンパックです。

- JavaScript入門 (全24回)
- JavaScriptでおみくじを作ろう (全9回)
- JavaScriptで5秒当てゲームを作ろう (全9回)
- JavaScriptで割り勘電卓を作ろう (全12回)
- JavaScriptでパスワードジェネレータを作ろう (全8回)
- JavaScriptで文字数チェッカーを作ろう (全8回)
- JavaScriptでストップウォッチを作ろう (全13回)
- JavaScriptでカウントダウンタイマーを作ろう (全12回)
- JavaScriptでスロットマシンを作ろう (全11回) PREMIUM

iOSアプリを作れるようになろう

iPhoneやiPad向けアプリの作り方をマスターできるようになるためのレッスンパックです。

- iPhoneアプリ開発入門 (全13回)
- iOSレイアウト入門 (全14回) PREMIUM
- Swift 3入門 (全36回) PREMIUM
- iOSでおみくじアプリを作ろう (全6回) PREMIUM
- iOSで姓名診断アプリを作ろう (全12回) PREMIUM
- iOSでストップウォッチを作ろう (全11回) PREMIUM
- iOSでブラウザを作ろう (全17回) PREMIUM
- iOSで一行メモアプリを作ろう (全18回) PREMIUM

9,476 人が学習中です

Androidアプリを作れるようになろう

Androidのスマートフォンやタブレット向けのアプリを作れるようになるためのレッスンパックです。

- Androidアプリ開発入門 (全11回)
- Androidでおみくじアプリを作ろう (全8回) PREMIUM

1,903 人が学習中です

仕事で使える技術に挑戦してみよう

仕事で使えるさまざまな技術やツールを学習していくためのレッスンパックです。

- WordPress入門 (全23回)
- Java 8入門 (全43回) PREMIUM

ゲームプログラミングに挑戦してみよう

ゲームを作りながらプログラミングを楽しく学習していくためのレッスンパックです。

- Unity入門 (全26回)
- Scratch 2.0入門 (全19回)

- Progate (プロゲート)

文字列

文字列とは？

・文字列　・クォーテーション

先ほどの例で用いた「Hello Python」という文字は、プログラミングの世界では「文字列」と呼ばれます。
文字列はシングルクォーテーション「'」またはダブルクォーテーション「"」で囲む必要があります。どちらで囲んでも出力結果は同じとなります。
どちらかで囲んでいない場合、コードは動かなくなります。

シングルクォーテーション (')
または、ダブルクォーテーション (") で囲む

script.py

```
print('Hello Python')
print("Hello Python")
```

script.py

```
# エラー発生
print(Hello Python)
```

出力結果

コンソール

```
Hello Python
Hello Python
```

出力結果は同じ

コンソール

```
SyntaxError: invalid syntax
```

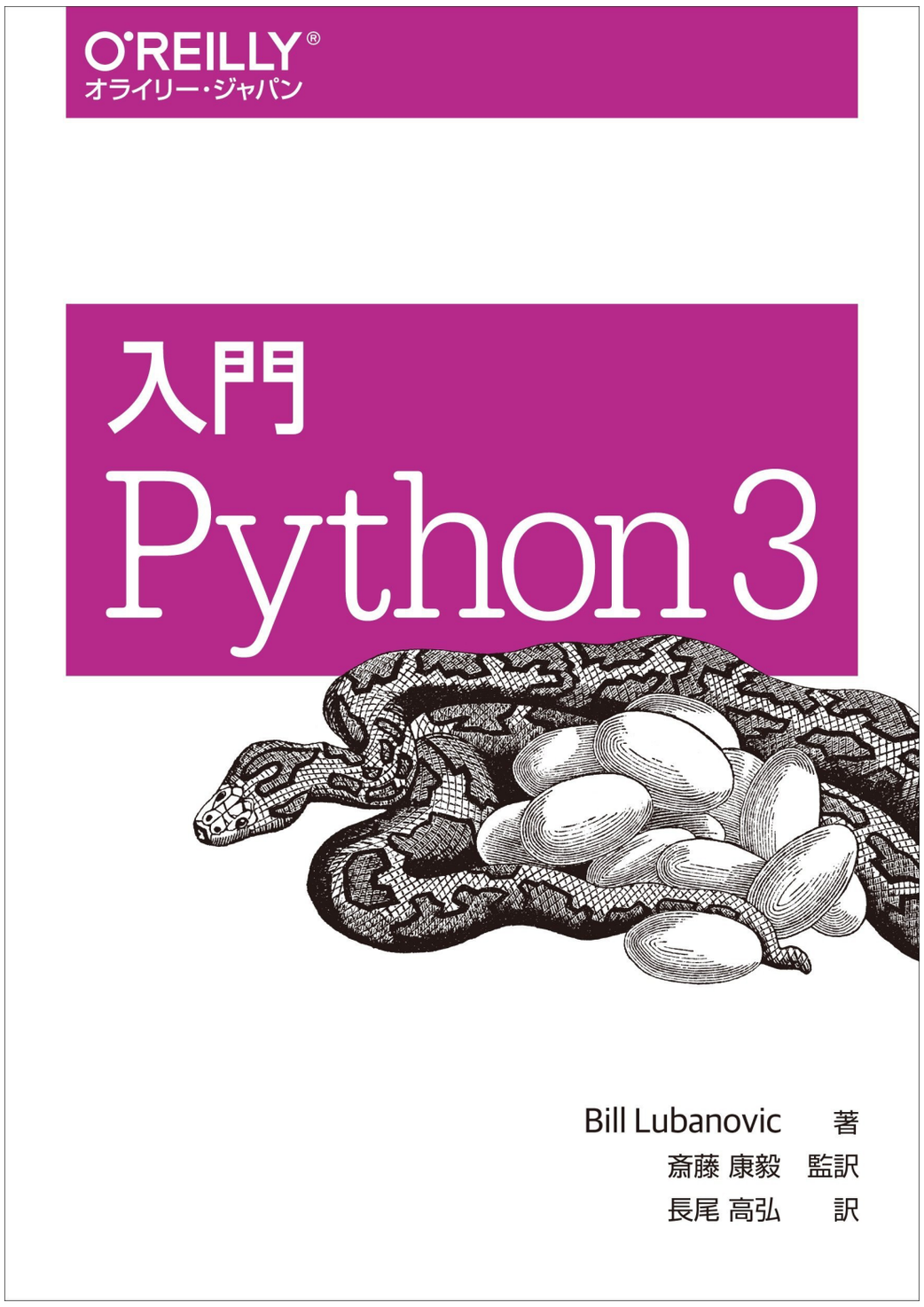
✕ エラー！！：必ず文字列はクォーテーションで囲む！

Pythonの学習について

- Python公式チュートリアル
  https://docs.python.jp/3/tutorial/
- LearnPython.org interactive Python tutorial (英語)
  https://www.learnpython.org
- CodeAcademy (英語)
  https://www.codecademy.com/learn/learn-python
- Intro to Python for Data Science (英語)
  https://www.datacamp.com/courses/intro-to-python-for-data-science/
- Python tutorials for beginners (英語)
  http://thepythonguru.com
- Progate (プロゲート)
  http://prog-8.com/languages/python
- ドットインストール
  https://dotinstall.com/lessons/basic_python_v3

本: 本当の本当に０スタート向け

本: もう少し広いこともある入門



まとめ：

- 基本は「公式チュートリアル」
  https://docs.python.jp/3/tutorial/
- 本はものすごくいろいろ出ていて個人によって難しすぎたり、簡単(冗長)すぎてつまらなかったりするので、本屋や図書館で手にとって自分の好みか確認するのが良い
- 入門むけのinteractiveなWeb教材は無料でもできが良い。英語もシンプル。ただし発展になると有料になったりするので注意

次回からPythonを掘り下げたい

宿題：

- USBメモリからJupyter Notebookを起動する手順を確認復習しておくこと
- 特に毎回必要となるコマンドプロンプト作業も確認復習しておくこと
- Jupyter Notebookにいろいろ打ち込んでエラー出してみたり遊んでみること
- 余裕ある人はPythonの教材もチェック

今日のお題：Pythonを始めよう

- 前回の復習とふりかえり
- paiza.IOでPythonに触れてみる
- Python言語の基本を学ぶ + 最初はどうやって勉強すればいいか？
- Pythonインタプリタと実行環境について
- USBメモリのPython環境の使い方 (改)
- Jupyter Notebookの動かしかた
- (補足) なぜ前回動かなかったか?
- (補足) USBメモリについて
- ミニレポートタイム