
Introduction to python

— 数值計算特論(1) —

Why python?

First, take a look at “Why python?” in “Scipy Lecture Notes”:

(ぼくも「科学者」であったので、これがしっくりきた。この授業を始めた 2016年頃)

<http://www.turbare.net/transl/scipy-lecture-notes/intro/intro.html>

original page: <http://www.turbare.net/transl/scipy-lecture-notes/intro/intro.html>

今はそれに加えて、

データサイエンス関係ライブラリ利用の最もアクティブな言語環境として急速に普及

My experiences

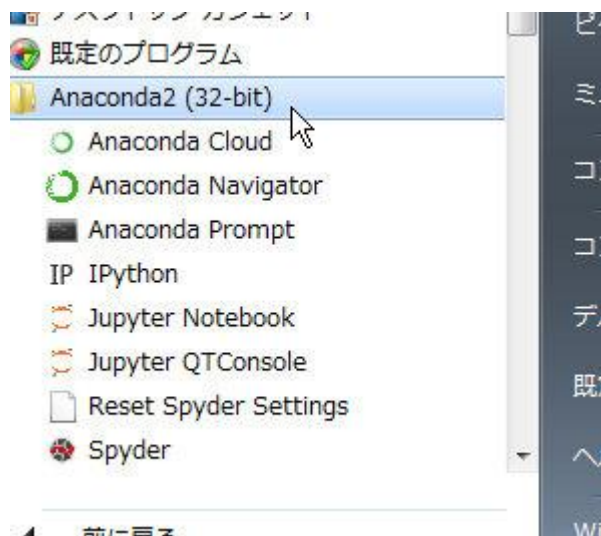
- FORTRAN, mid-1980s - 2000 for simulations
- Turbo Pascal, Turbo C, mid-1990's for classes
- Perl, mid-1990s - early 2000's for data analysis
- PHP, 2000 - for Web applications (+ MySQL)
- C on UNIX, 2000 - for simulations and classes
- JAVA, early-2000s - for simulations
- JavaScript, late-2000s for Web applications
- python, 2016 - for class, data analysis and controlling RasPi
- Others: HTML, shell script, awk, basic, ...

Trends: <https://www.tiobe.com/tiobe-index/>

<https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>

<http://redmonk.com/sograzy/2017/06/08/language-rankings-6-17/>

pythonの開発・実行環境(ツール)



1. 統合開発環境 **Spyder**
 - a. エディタ、コンソール、デバッガなど
2. 対話型コンソール**IPython** + 外部エディタ
 - a. **Jupyter QtConsole**も同様
3. ノートブック **Jupyter Notebook**
4. 外部エディタで編集 → python.exeのコマンドライン引数にプログラムを渡す
(Windowsではファイルとアプリの関連付けができれば、ダブルクリックで実行可)

Spyder

プログラムエディタ

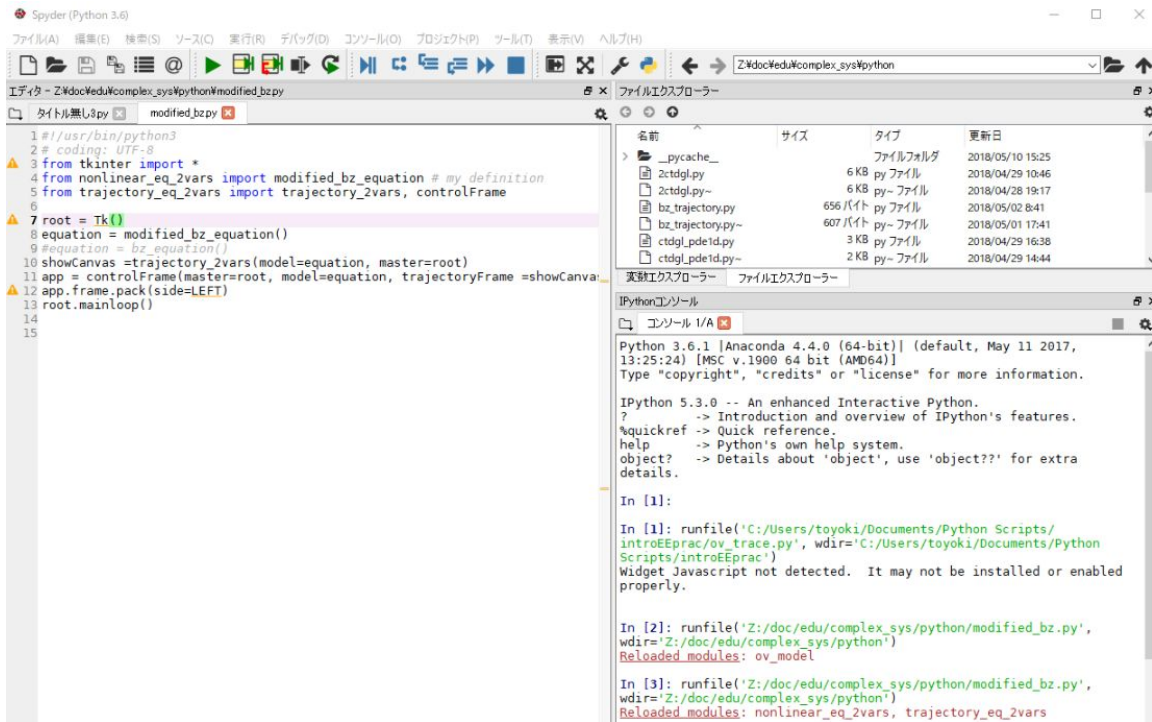
コンソール (IPython)

プロファイラーなどの様々な高機能がある

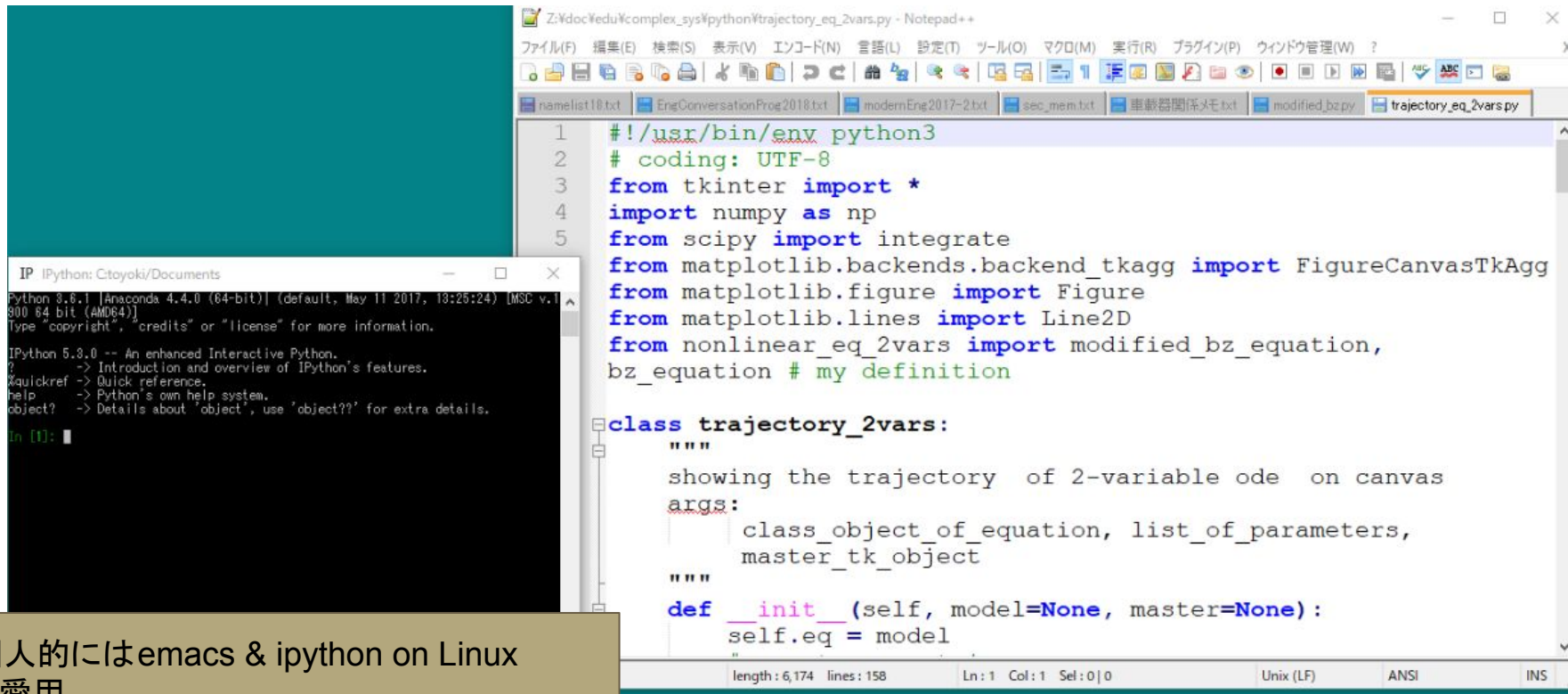
[短所]

立ち上がりが遅い

キーレイアウト(ショットカットキー)のカスタマイズが困難



外部エディタ & IPython: Notepad++の場合



The image shows a Notepad++ window with a Python script named `trajectory_eq_2vars.py`. The script uses `#!/usr/bin/env python3` and imports `tkinter`, `numpy`, `scipy.integrate`, `matplotlib.backends.backend_tkagg`, `matplotlib.figure`, `matplotlib.lines`, and `nonlinear_eq_2vars`. It defines a `trajectory_2vars` class with a docstring and an `__init__` method that takes `model` and `master` as arguments.

```
1 #!/usr/bin/env python3
2 # coding: UTF-8
3 from tkinter import *
4 import numpy as np
5 from scipy import integrate
6 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
7 from matplotlib.figure import Figure
8 from matplotlib.lines import Line2D
9 from nonlinear_eq_2vars import modified_bz_equation,
10 bz_equation # my definition
11
12 class trajectory_2vars:
13     """
14     showing the trajectory of 2-variable ode on canvas
15     args:
16         class_object_of_equation, list_of_parameters,
17         master_tk_object
18     """
19     def __init__(self, model=None, master=None):
20         self.eq = model
```

An IPython terminal window is also visible, showing the IPython 5.3.0 prompt and help text.

個人的にはemacs & ipython on Linux
を愛用
VisualStudioとの連携も進んでいるようだ

jupyter notebook

Mathematicaライクなノートとプログラムエディタ、実行環境がWebブラウザの中で動く

さまざまなプラグイン開発がアクティブ

この講義で使用

個人的には、

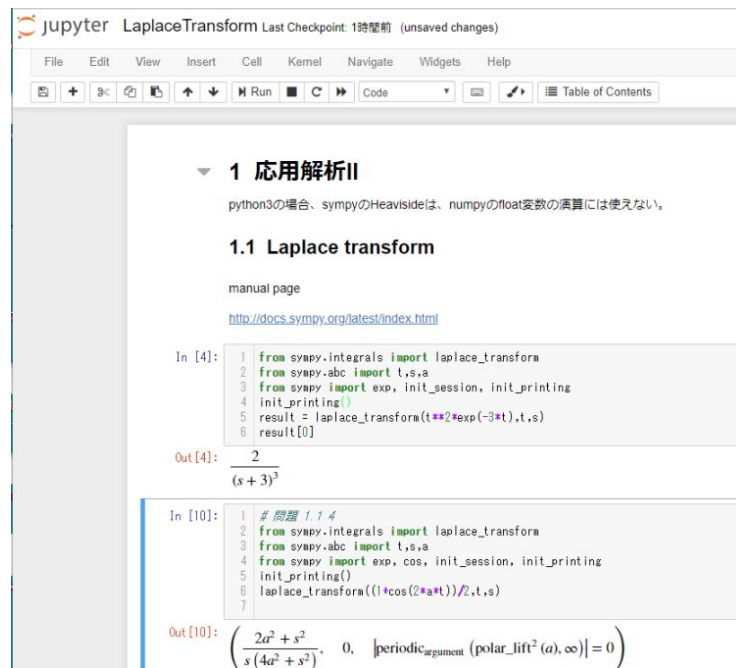
- ・授業用資料作成
- ・短いコードの実行 (右図は数式処理利用の例)
- ・各種コードのメモ用

に利用

簡単なGUIを作りたいときも便利

[欠点]

今のところインラインのアニメーションが作れない



The screenshot shows a Jupyter Notebook window titled "LaplaceTransform" with a "Last Checkpoint: 1時間前 (unsaved changes)" status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Navigate, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook content is organized into sections:

- 1 応用解析II**
 - python3の場合、sympyのHeavisideは、numpyのfloat変数の演算には使えない。
 - 1.1 Laplace transform**
 - manual page
 - <http://docs.sympy.org/latest/index.html>

Two code cells are shown:

In [4]:

```
1 from sympy.integrals import laplace_transform
2 from sympy.abc import t,s,a
3 from sympy import exp, init_session, init_printing
4 init_printing()
5 result = laplace_transform(t**2*exp(-3*t),t,s)
6 result[0]
```

Out[4]:

$$\frac{2}{(s+3)^3}$$

In [10]:

```
1 # 問題 1.1 4
2 from sympy.integrals import laplace_transform
3 from sympy.abc import t,s,a
4 from sympy import exp, cos, init_session, init_printing
5 init_printing()
6 laplace_transform((1+cos(2*a*t))/2,t,s)
7
```

Out[10]:

$$\left(\frac{2a^2 + s^2}{s(4a^2 + s^2)}, 0, \left| \text{periodic_argument}(\text{polar_lift}^2(a), \infty) \right| = 0 \right)$$

VScodeもお薦め

様々なプログラム開発環境として、人気急上昇中

The screenshot displays the Visual Studio Code interface with several Python files open. The Explorer sidebar on the left shows a project structure with folders for 'PYTHON' and 'PYCACHE'. The main editor area shows the code for 'ctdgLpde1d.py' and 'nonlinear_eq_2vars.py'. The code in 'ctdgLpde1d.py' includes imports for numpy, matplotlib, and animation, and defines a class 'TDGL' with methods for initialization and field updates. The code in 'nonlinear_eq_2vars.py' defines a 'modified_bz_equation' class with parameters for a BZ model.

```
1 #!/usr/bin/python3
2 ...
3 Calculation and Animation for One-dimensional complex TDGL equation
4 (solved by simple Euler method.)
5
6 ...
7 import numpy as np
8 from matplotlib import pyplot as plt
9 from matplotlib import animation
10 import sys
11 import math
12
13
14 size = 400
15 amp = np.zeros(size) # used in the computing process as amplitudes
16 diff = np.array(amp, dtype=np.complex128) # used as the Laplacian term
17
18 x = np.random.rand(size)
19 x = np.array(x, dtype=np.complex128)
20 y = np.array(x, dtype=np.complex128)
21 site_list = range(size)
22 # parameters of the differential equation
23 c0 = 1.0
24 c1 = -1.0
25 c2 = 0.6
26 c0i = 1.0 + c0*1.0j
27 Di = 1.0 + c1*1.0j
28 ci = 1.0 + c2*1.0j
29 boundary = "f"
30
31 def updateField(x_in, x_out, c0i = 0.0 + 0.0j, ci = 1 + 0.6j, Di = 1 - 1.0j, boundary):
32     DT = 0.002 # time step
33     amp = np.absolute(x_in)
34     amp = np.power(amp, 2)
35     diff[1:-1] = x_in[0:-2,] + x_in[2:] - 2.0* x_in[1:-1]
36     x_out[1:-1] = x_in[1:-1] + DT*(c0i - ci* amp[1:-1])*x_in[1:-1] + DT* Di * diff
37     # periodic boundary condition
```

```
1 # coding: UTF-8
2 ...
3 To inheritance
4
5 "bz_equation" could be the parent class for the model
6 | including functions adjusting the x and y widths
7 "modified_bz_equation" could be the parent without adjust function
8
9 set_parameters() and get_parameters() can be inherited,
10 | and other functions should be overridden.
11
12 ...
13 import numpy as np
14
15 class modified_bz_equation:
16     ...
17     modified BZ model showing excitability similar to neural membrane model
18     du/dt = u(1-u) - bv(u-c)/(u+c)
19     dv/dt = u - v
20     ...
21     def __init__(self, a=0.25, epsilon=0.002, gamma=14.0, I=0.4):
22         self.ini_params = []
23         self.ini_params.append({"name": "a", "min": 0.0, "max": 0.5,
24                                 "ini_val": 0.25, "resolution": 0.01})
25         self.ini_params.append({"name": "epsilon", "min": 0.001, "max": 0.05,
26                                 "ini_val": 0.002, "resolution": 0.001})
27         self.ini_params.append({"name": "gamma", "min": 0.1, "max": 20,
28                                 "ini_val": 14.0, "resolution": 0.1})
29         self.ini_params.append({"name": "I", "min": 0.01, "max": 1.0,
30                                 "ini_val": 0.4, "resolution": 0.01})
31
32         self.x_min = -0.3
33         self.x_max = 1.1
34         self.y_min = -0.2
35         self.y_max = 0.4
36         self.params = {}
37         for i, item in enumerate(self.ini_params):
38             self.params[item["name"]] = item["ini_val"]
```

クをe-Learningのページへ
8月8日)

[view.php?id=2167](#)

ids
//feedback/analysis.php?id

(including "Why python?")

きには、IEではなく、Chrom

fail to download jquery
file type and so add som
I recommend you to use a

この授業の教材及び実習について

- 教材はWebページで行う
- スクリプトを含む教材は基本的にはjupyter notebookを用いて作成
 - 演示もjupyter notebookにて示す
- スクリプトの実行により体験的に学ぶことを基本とする
 - 積極的な質問を歓迎する
- 実習(演示プログラムの実行や独自のプログラム作成)はどの環境で行ってもよい
 - 統合環境spyderを立ち上げ、notebookにある個々のスクリプトをspyder内に新しいファイルを作成し、そこにコピー&ペーストして実行してみるのがよい かもしれない
 - その際、notebookのスクリプト中
#matplotlib inline
のような行頭の#で始まる文はnotebook独自のものであるので、除くこと
- python3環境を作って、試してみてほしい。だめなら情報処理教室の端末で
(python3とpython2.7それぞれのAnacondaパッケージが入っている)

python文法をざっ と

少なくとも1つのプログラム言語を習ったことがあることを想定

ここから後の説明は[python notebook](#)で

1. 変数
 - a. 文字列
 - b. 配列(リストと辞書)
2. 制御構造 (if, for, while, ...)
3. 関数定義

以下は後で

4. 複数ファイルのコード利用
 5. オブジェクト指向
-