Python Cheat Sheet

青: このゼミで重要; 赤: 間違えやすい注意が必要; ∗: マスターの必要性はないが, わかっとく必要あり; †: 飛ばして良い

1 Jupyter 環境

1.1 実行

セル上で Shift + Enter

1.2 既存セルの編集

セルをマウスクリックもしくは Enter

1.3 作業の保存

Ctrl-s または保存ボタン

1.4 ファイル名の変更

画面上部のファイル名 (初期状態は Untitled) 部をクリック

1.5 その他の操作

Help → Keyboard Shortcuts を参照

2 式

2.1 数リテラル

```
3
3.45
1.0e-5
```

2.2 文字列リテラル

```
"hello"
'hello'
'he said "hi" and I said "bye"'
"y = f'(x)"
"""string with
multiple lines"""
'''another string with
multiple lines '''
```

2.3 リスト ∗

```
[]
[1]
[ 1, 2, 3]
["one", "two", 'three']
[ 1, 2, "san!" ]
[ [ 1, 2 ], [ 2, 1 ] ]
```

2.4 タプル ∗

```
()
(1,)      # (1) ではない
(1,2)
(1,2,)    # (1,2) と同じ
(1,2,3)
("one", "two", 'three')
(1, [2,3], (4,5,6))
```

2.5 辞書 †

```
{}
{ "one" : 1, "two" : 2, "three" : 3 }
```

2.6 変数参照

変数代入 (3.2 節) も参照. (x = 10 とする)

```
x          # ⇒ 10
x + 3      # ⇒ 13
```

代入されてない変数の参照 (エラー)

```
y
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'y' is not defined
```

2.7 演算

2.7.1 数の演算

(x = 2, y = 100 とする)

```
1 + 2 * 3    # ⇒ 7
4 / 5        # ⇒ 0.8 (上との違いに注意!)
40 // 7      # ⇒ 5 (商を整数とする割り算)
2 ** 100     # ⇒ 1267650600228229401496703205376
x + y / 3    # ⇒ 35.333333333333336
x ** y       # ⇒ 1267650600228229401496703205376
```

2.7.2 文字列の演算

(x = 2, y = 100) とする

```
"hello" + " and good bye"   # ⇒ 'hello and good bye'
"超" * 5 + "かわいい"        # ⇒ '超超超超超かわいい'
"x = %s" % x                # ⇒ 'x = 5'
"x = %s, y = %s" % (x, y)   # ⇒ 'x = 2, y = 100'
s = "abcdefghijklmnopqrstuvwxyz"
len(s)                      # ⇒ 26
s[0]                        # ⇒ 'a'
s[3:7]                      # ⇒ 'defg'
s[:7]                       # ⇒ 'abcdefg'
s[3:]                       # ⇒ 'defghijklmnopqrstuvwxyz'
for c in s:                 # ⇒ c = 'd', 'e', 'f', ...
    ...
```

2.7.3 タプルの演算 ∗

```
t = ("abc", "def", "ghi", "jkl")
len(t)
t[0]          # ⇒ 26
t[1:3]        # ⇒ ('def', 'ghi')
t[:3]         # ⇒ ('abc', 'def', 'ghi')
t[1:]         # ⇒ ('def', 'ghi', 'jkl')
t + t         # ⇒ ("abc", "def", "ghi", "jkl", "abc", "def", "ghi", "jkl")
t * 3         # ⇒ ("abc", "def", "ghi", "jkl", "abc", "def", "ghi", "jkl", "abc", "def", "ghi", "jkl")
for s in t:   # ⇒ s = 'abc', 'def', 'ghi', 'jkl'
    ...
```

2.7.4 リストの演算

```
L = [ 1, 4, 9, 16 ]
M = [ "a", "b", "c", "d" ]
len(L)          # ⇒ 4
L[0]            # ⇒ 1
L[1:3]          # ⇒ [ 4, 9 ]
L[:3]           # ⇒ [ 1, 4, 9 ]
L[1:]           # ⇒ [ 4, 9, 16 ]
L + L           # ⇒ [ 1, 4, 9, 16, 1, 4, 9, 16 ]
L * 3           # ⇒ [ 1, 4, 9, 16, 1, 4, 9, 16, 1, 4, 9, 16 ]
L[1] = 10
L               # ⇒ [ 1, 10, 9, 16 ]
L.append(25)
L               # ⇒ [ 1, 10, 9, 16, 25 ]
del L[2]
L               # ⇒ [ 1, 10, 16, 25 ]
for x in L:     # ⇒ x = 1, 10, 16, 25
    ...
enumerate(L)    # ⇒ [ (0,1), (1,4), (2,9), (3,16) ]
zip(L, M)       # ⇒ [ (1,"a"), (4,"b"), (9,"c"), (16,"d") ]
```

2.7.5 辞書の演算 †

```
D = { "one" : 1, "two" : 2, "three" : 3, (0,1) : "zero one" }
len(D)              # ⇒ 3
D["one"]            # ⇒ 1
D[0,1]              # ⇒ 'zero one'
D[2,3] = 10
D                   # ⇒ { "one" : 1, "two" : 2, "three" : 3, (0,1) : "zero one", (2,3) : 10
                    }
del D[0,1]
D                   # ⇒ { "one" : 1, "two" : 2, "three" : 3, (2,3) : 10 }
for k,v in D.items(): # ⇒ (k,v) = ("one",1), ("two",2), ("three",3), ((2,3), 10)
    ...
```

2.8 関数呼び出し

x = -7.29 とする

```
abs(x)          # 7.29
min(3, 4, 8)    # 3
max(3, 4)       # 4
3 / float(4)    # 0.75
2 + int(7.9)    # 9
...
```

3 文

3.1 print 関数

x = -7.29, y = 9.18 とする.

```
print(x)                    # 7.29
print(x, y)                 # −7.29 9.18
print("x=", x, " y=", y)    # x=7.29 y=9.18
print("x=%s y=%s" % (x,y))  # x=7.29 y=9.18
```

3.2 代入文

```
x = -7.29
y,z = (9.18, -x)          # y=9.18, z=7.29
t = (9.18, -x)
p,q = t                   # p=9.18, q=7.29
```

3.3 関数定義

文法:

```
def 名前 (名前, 名前, ...):
    文
    文
    ...
```

例:

```
def f(x):
    print("x = %s" % x)
    print("x * x = %s" % (x * x))

f(3) # x = 3
     # x * x = 9
```

以下は, 関数にローカルな変数と言い, 関数呼び出しごとに別々に記憶される.

- 関数の入力変数 (引数)
- 関数内で代入される変数

例:

```
# g の x, y と f の x, y は別物
def g(x):
    y = x + 1
    return y

def f(x):
    y = x + 2
    z = g(x + 3)
    return x + y + z

f(0)  # ⇒ 6 (0 + 2 + 4)
```

例:

```
def f(x):
    y = x + 1
    z = y * y
    return z

f(3)
# f の x と外側の x は別物
x       # ⇒ 10
# f 内の y, z は f 終了後には存在しない
y       # ⇒ error
```

3.4 return 文

文法:

```
return 式
```

- 式を計算 → R
- 現在の関数実行を終了してその関数呼び出しの値を R とする
- (関数実行中でなかったらエラー)

例:

```
def f(x):
    print("x = %s" % x)
    return x * x

f(3)  # x = 3
      # ⇒ 9
```

3.5 import

文法:

```
import モジュール名
import モジュール名 as 名前
from モジュール名 import 名前,名前,...
from モジュール名 import *
```

- 指定された「モジュール」の機能 (名前) を使えるようにする
- 通常ファイルの先頭にまとめて書く

例:

```
import random
x = random.random()
```

```
from vpython import *
sphere()
```

```
from math import pi,cos
cos(pi)    # ⇒ −1.0
```

```
import matplotlib.pyplot as plt
plt.plot(range(0,100), range(3,303,3))
plt.show()
```

3.6 if

文法

```
if 式:
    文
    ...
elif 式:
    文
    ...
...
else:
    文
    ...
```

- elif は 0 個以上
- else は最後にひとつ (か無し)

3.7 for

文法:

```
for 名前,名前,... in 式:
    文
    文
    ...
```

3.8 while

文法:

```
while 式:
    文
    文
    ...
```

3.9 break

文法:

```
break
```

- 現在実行中のループ (for, whille) の実行を即座に終了する
- ループ実行中でなければエラー

3.10 continue

文法:

```
continue
```

- 現在実行中のループ (for, whille) の, 「現在の繰り返し」の実行を即座に終了する (次の繰り返しのための条件判定に直ちに移行)
- ループ実行中でなければエラー

4 リスト内包表記

必須ではないが色々なものを「一撃で」書くのにマスターする価値あり

文法:

```
[ 式 for 名前,名前,... in 式 ]
[ 式 for 名前,名前,... in 式 if 式 ]
```

- [E for x in L] は, L 中の各 x に対して E を計算したものを要素とするリスト
- [E for x in L if C] はその中から C を満たすものだけを残したリスト

```
[ x * x for x in range(5) ]              # [0,1,4,9,16]
[ x * x for x in range(5) if x % 2 == 0 ] # [0,4,16]
sum([ x * x for x in range(5) ])         # 30
n = 10000
X = [ i / float(n) for i in range(n) ]   # 0.0001,0.0002,...
sum([ x * x / n for x in X ])            # 0.33328333...
```

5 クラス定義 †

文法:

```
class 名前:
    関数定義

    関数定義

    ...
```

例:

```
class point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def dist2(self):
        return self.x ** 2 + self.y ** 2
    def move(self, dx, dy):
        self.x = self.x + dx
        self.y = self.y + dy

p = point(3, 4)
p.dist2()
p.move(1, 2)
p.x
p.y
```

例:

```
from vpython import *
s = sphere()
s.radius = 5
s.pos = (1,2,3)
```

6 dir

オブジェクトやモジュールの属性を調べる

```
import math
dir(math)
from vpython import *
s = sphere()
dir(s)
```

7 文法上の諸注意

7.1 字下げ

Python において字下げの位置は重要で，def, for, while, if などにおいてその本体がどこまでかを明示するのに字下げが使われる

```
for x in range(0,10):
    print(x)        # for 文の中
    print(x * x)    # for 文の中
print x             # 外
```

7.2 コメント

```
# から行末までがコメント
```

7.3 日本語

- コメントや文字列の中に日本語が使える
- 使いたい場合以下をファイルの冒頭に入れる

```
#! −*− coding: utf−8 −*−
```

8 Visual Python

`https://vpython.org/`

8.1 import

```
from vpython import *
```

8.2 描画の例

```
from vpython import *
s = sphere()
s.pos = vector(1,2,3)
s.color = color.yellow
```

- オブジェクト: arrow (矢印), box (箱), cone (円錐), curve (線), cylinder (柱), ellipsoid (楕円体), faces (多面体), helix (らせん), label (文字) , points (点をいっぱい), pyramid (四角錐), ring (輪), sphere (球), text (立体文字), . . .
- 一 覧: `http://vpython.org/contents/docs/primitives.html`
- 属性: pos, color, axis, radius, make_trail, . . .

8.3 色

```
from vpython import *
s = sphere(color=color.white)
b = box(color=color.gray(0.3))
r = ring(color=color.red, opacity=0.8)
c = cone(color=(0.4, 0.2, 0.8)) # RGB
```

- 他の色 (color.xxx): red, yellow, black, green, orange, white, blue, cyan, magenta
- color.hsv_to_rgb, color.rgb_to_hsv

8.4 アニメーション

```
for i in range(n):
    rate(30) # 30フレーム/秒
    ...
    s.pos = new position
```

8.5 カメラアングルなど

自動追従の抑止

```
scene.autocenter = 0
scene.autoscale = 0
```

`http://vpython.org/contents/docs/options.html`

8.6 軌跡を残す

```
from vpython import *
trail = curve()
for i in range(n):
    rate(30)
    pos = ...
    trail.append(pos)
```

`http://vpython.org/contents/docs/trail.html` にある例はうまくいかない

8.7 vector

```
from vpython import *
u = vector(1,2,3)
v = vector(9,8,7)
u + v, u − v, 3 * u
u.dot(v), u.cross(v)
u.mag, u.mag2
```

`http://vpython.org/contents/docs/vector.html`

9 Matplotlib

`http://matplotlib.org/`

9.1 import

```
import matplotlib.pyplot as plt
```

9.2 グラフ描画 (1 変数)

x, y 座標それぞれをリストまたは numpy 配列で与える

```
import matplotlib.pyplot as plt
plt.plot([0,1,2,3], [0,1,4,9]) # (0,0),(1,1),(2,4),(3,9)
plt.show()
```

9.3 データの作り方様々

リスト:

```
import math
import matplotlib.pyplot as plt
X = range(100) # リスト [0,1,..,99]
# リスト内包表記 [ √0, √1, . . . , √99 ]
Y = [ math.sqrt(x) for x in X ]
plt.plot(X, Y)
plt.show()
```

numpy:

```
import matplotlib.pyplot as plt
import numpy as np
X = np.arange(0, 7, 0.01) # numpy 配列 0,0.01,0.02,...,6.99
Y = np.sin(X)             # numpy universal 関数
plt.plot(X, Y)
plt.show()
```

9.4 複数グラフ描画

- 本当はこちらだけを覚えれば良い
- 注: 以下の 3D 表示時も必須

```
import matplotlib.pyplot as plt
import numpy as np
X = np.arange(0, 7, 0.01) # numpy 配列 0,0.01,0.02,...,6.99
# ウインドウ全体
fig = plt.figure()
# 2x3 のタイルに二つのグラフ領域
ax0 = fig.add_subplot(2,3,1)
ax1 = fig.add_subplot(2,3,6)
# 描画
ax0.plot(X, np.sin(X))
ax1.bar(X, np.cos(X))
fig.show()
```

9.5 2 変数データの描画

基本: x, y, z 座標それぞれを, 形の同じ 2 次元 numpy 配列で与える

```
import matplotlib.pyplot as plt
import numpy as np
# [0,2]x[0,3]
X = np.array([[0,0,0,0],[1,1,1,1],[2,2,2,2]])
Y = np.array([[0,1,2,3],[0,1,2,3],[0,1,2,3]])
Z = X ** 2 - Y ** 2 # array([[0,−1,−4,−9],[1,0,−3,−8]])
plt.pcolor(X, Y, Z)
plt.show()
```

格子点を生成するには `np.meshgrid` が有用

```
import matplotlib.pyplot as plt
import numpy as np
X = np.linspace(0,1,11) # array([0,0.1,0.2,...,1.0])
Y = np.linspace(0,1,11)
X,Y = np.meshgrid(X, Y)
Z = X ** 2 - Y ** 2 # array([[0,−1,−4,−9],[1,0,−3,−8]])
plt.pcolor(X, Y, Z)
plt.show()
```

9.6 3D 描画

```
import matplotlib.pyplot as plt
import numpy as np
import mpl_toolkits.mplot3d.axes3d

fig = plt.figure()
ax = fig.add_subplot(1,1,1, projection='3d')
X = np.linspace(0,1,11) # array([0,0.1,0.2,...,1.0])
Y = np.linspace(0,1,11)
X,Y = np.meshgrid(X, Y)
Z = X ** 2 - Y ** 2 # array([[0,−1,−4,−9],[1,0,−3,−8]])
ax.plot_surface(X, Y, Z)
fig.show()
```

9.7 ギャラリー

https://matplotlib.org/gallery/index.html

10 Numpy

10.1 import

```
import numpy as np
```

10.2 配列 (array)

```
import numpy as np
np.array([2,0,1,4])
np.array([[1,2,3],[4,5,6]])
```

10.3 配列の作り方様々

```
import numpy as np
np.arange(3.0,4.0,0.2) # array([ 3., 3.2, 3.4, 3.6, 3.8])
np.linspace(3.0,4.0,5) # array([ 3., 3.25, 3.5, 3.75, 4. ])
np.eye(50)             # 50x50 の単位行列(I)
np.zeros((50,100))     # 50x100 の 0
np.ones((50,100))      # 50x100 の 1
np.random.random((50,100)) # 50x100 のランダムな行列
# 汎用的・地道な作り方 (1)
a = np.zeros((50,100))
for i in range(50):
  for j in range(100):
    a[i,j] = i + j
# 汎用的・地道な作り方 (2)
def aij(i, j): return i + j
np.fromfunction(aij, (50,100))
np.fromfunction(lambda i,j: i + j, (50,100))
np.array(range(6)).reshape((2,3)) # array([[0,1,2],[3,4,5]])
np.linspace(0.0,1.0,6).reshape((2,3)) # array([[0., 0.2, 0.4],[0.6, 0.8, 1.]])
```

10.4 配列のアクセス

```
a = np.array(range(6)).reshape((2,3))
a[0,0]   # 0
a[0]     # array([0, 1, 2])
a[0,:]   # array([0, 1, 2])
a[:,1]   # array([1, 4])
a[0,1:3] # array([1, 2])
a[:,1:3] # array([[1, 2],[4, 5]])
a[:,1:3] = np.array([[10,20],[40,50]])
a        # array ([[0,10,20],[3,40,50]])
```

10.5 配列の演算

```
import numpy as np
x = np.array([2,0,1,4])
y = np.array([5,6,7,8])
x + y    # array([ 7,  6,  8, 12])
x * y    # array([10,  0,  7, 32])
x.dot(y) # 49
```

10.6 配列を行列として使う

```
import numpy as np
a = np.array([[1,2,3],[0,4,5],[0,0,6]])
a.T # 転置
x = np.array([100,200,300])
a * x # 行列 x ベクトル積じゃない!
a.dot(x) # array([1400, 2300, 1800])
```

10.7 matrix

```
import numpy as np
a = np.matrix([[1,2,3],[0,4,5],[0,0,6]])
x = np.array([[100],[200],[300]])  # 縦ベクトルは,n x 1の行列
a * x
```

10.8 線形代数の計算

```
import numpy as np
a = np.array([[1,2,3],[0,4,5],[0,0,6]])
x = np.array([100,200,300])
np.linalg.solve(a, x) # Ax = b
np.linalg.eig(a)      # 固有値と固有ベクトル
np.linalg.inv(a)      # 逆行列
m = np.matrix(a)
x / m
```

10.9 Universal 関数

多くの実数に対する演算が, 自動的に配列に対する演算に拡張される

```
import numpy as np
X = np.linspace(0,4,5) # ⇒ array([0, 1, 2, 3, 4])
X + 3                  # ⇒ array([3, 4, 5, 6, 7])
4 * X ** 2             # ⇒ array([0, 4, 16, 36, 64])
np.sin(X)              # ⇒ array([0., 0.84147098, 0.90929743, 0.14112001, −0.7568025 ])
```

A Emacs の操作 †

A.1 キー操作の記法

- C- は Ctrl キーを押しながら
- M- は Meta (Alt) キーを押しながら

例:

- C-x C-f ⇒「Ctrl キーを押したまま x, f と連打」
- C-x 1 ⇒「Ctrl キーを押して x, Ctrl キーを離してから 1」

A.2 ファイル操作

メニュー (Files) もある

操作	キー
開く	C-x C-f
保存	C-x C-s
ファイル一覧	C-x d

A.3 コマンドを途中で抜ける

操作	キー	備考
コマンドを途中で抜ける	C-g	

A.4 カーソル移動

操作	キー	備考
1 ページスクロール	C-v	
1 ページ逆スクロール	M-v	
カーソル移動 右	C-f	forward
カーソル移動 左	C-b	backward
カーソル移動 下	C-n	next
カーソル移動 上	C-p	previous
カーソル移動 行頭	C-a	
カーソル移動 行末	C-e	

A.5 コピペ

操作	キー
行末までカット	C-k
1 行全部カット	C-a C-k C-k
ペースト	C-y
広い範囲をカット	C-space; カーソル移動; C-w
広い範囲をコピー	C-space; カーソル移動; M-w

- 範囲選択はマウスでも可能
- C-space ではなく，C-@ の場合も有る

A.6 検索

操作	キー	備考
検索 (前向き)	C-s	
検索 (後ろ向き)	C-r	

A.7 バッファ (窓) 切り替え

メニュー (Buffers) もある

操作	キー
今いるバッファを見えなくする	C-x 0
今いるバッファだけにする	C-x 1
今いるバッファを閉じる	C-x k
他のバッファを表示する	C-x b