

プログラミング応用 第1回

河瀬 康志

2018 年 6 月 11 日

担当教員： 河瀬康志

TA: 徐安洋（松井研 M2）

python3 を用いての応用的な技術の習得を目指す.

python3 の実行, for 文, if 文などについて基礎は前提.

参考書：

- <http://docs.python.jp/3.6/tutorial/index.html>
- Mark Lutz (著), 夏目 大 (訳), 「初めての Python」
- John V. Guttag (著), 久保幹雄 (訳), 「Python 言語によるプログラミングイントロダクション」

「プログラミング基礎」の履修を強く推奨

	日程	内容
第1回	6/11	ガイダンス・復習
第2回	6/18	文字列操作（文字列整形，パターンマッチ，正規表現） 平面幾何（線分の交差判定，点と直線の距離，凸包）
第3回	6/25	乱数（一様分布，正規分布への変換，乱数生成） 統計（データ処理，フィッティング）
第4回	7/2	計算量（オーダー表記） スタックとキュー（幅優先探索，深さ優先探索）
第5回	7/9	ソートアルゴリズム バックトラック（Nクイーン問題，数独）
第6回	7/23	動的計画法（ナップサック問題） 最短経路探索（Warshall-Floyd, Bellman-Ford, Dijkstra）
第7回	7/30	巡回セールスマン問題
期末試験	8/6?	—

- Python 3.6.x
 - <https://conda.io/miniconda.html> からインストール
- PC: 備え付け持ち込みどちらでも. Windows, Mac, Linux 全て可
- 開発環境
 - Python IDLE:
 - Python と一緒にインストールされます
 - File/New File から新しいスクリプトファイルを作成
 - F5 を押すと実行
 - PyCharm: <https://www.jetbrains.com/pycharm/>
 - Atom: <https://atom.io>
 - サクラエディタ: <http://sakura-editor.sourceforge.net>
 - Emacs
 - vi/vim
 - Jupyter Notebook
 - Google Colaboratory <https://colab.research.google.com/>

演習問題の提出状況 + 期末試験

- 演習

- 途中まででも提出すれば部分点はつけます
- ただし、現状でどこまで実装できていて、
どういう場合にうまく動かないかをコメントでつけてください
- 相談推奨 (コピペ不可)

- 期末試験

- 持ち込み可，記述式を予定
- ただしインターネットの使用は不可

- 対話環境

```
% python3  
(中略)  
>>> 1+2  
3
```

- ワンライナー実行

```
% python3 -c "print('Hello world!')"
```

- スクリプトの実行

```
% python3 hoge.py
```

```
>>> 2*3 # かけ算
6
>>> 7/2 # 割り算
3.5
>>> 7//2 # 整数に切り捨てたいときはスラッシュ2つ
3.0
>>> 11 % 4 # 剰余
3
>>> 2 ** 3 # 累乗
8
>>> divmod(14,3)
(4, 2)
>>> (1+2j)**2 # 複素数
-3+4j
>>> 2.7182818285**(3.1415926536j)
(-1-5.753918418796315e-11j)
```

```
>>> n=3
>>> n
3
>>> n**2
9
>>> n=n+1
>>> n
4
>>> m # 未定義の変数
Trace back (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'm' is not defined
```


文字列

```
>>> s='spam' # 文字列はシングルクォートで囲んで表す
>>> t="ham" # ダブルクォーテーションでも良い
>>> s+t # 連結
'spamham'
>>> s*3 # 繰り返し
'spamspamspam'
>>> s[2] # インデクシング
'a'
>>> s[-1] # インデクシング
'm'
>>> s[1:3] # スライシング
'pa'
>>> s[:3] # 最初の 3 文字
'spa'
>>> s[2:] # 最初の 2 文字以外
'am'
>>> len(s) # 長さ
4
>>> int('42') # 整数に変換
42
```

エスケープシーケンス

<code>\</code>	改行	行の継続
<code>\n</code>	改行	
<code>\t</code>	タブ	
<code>\b</code>	バックスペース	
<code>\\</code>	バックスラッシュ	
<code>\'</code>	シングルクォーテーション	
<code>\"</code>	ダブルクォーテーション	
<code>\\</code>	バックスラッシュ	

```
>>> s='it\'s\tab'
>>> s
"it's\tab"
>>> print(s)
it's    a tab
```

リスト

- オブジェクトを一定の順序で並べたもの
- 変更を加えることができる

```
>>> l = [0,1,2,'three',[4]] # 型が違うものも並べられる, ネストも可
>>> l[3]
'three'
>>> l[1:3]
[1,2]
>>> l+[5,6,7] # 連結
[0, 1, 2, 'three', [4], 5, 6, 7]
>>> l*2 # 繰り返し
[0, 1, 2, 'three', [4], 0, 1, 2, 'three', [4]]
>>> len(l) # 長さ
5
>>> sum([1,2,3]) # リストの要素和
6
>>> list('abcd')
['a', 'b', 'c', 'd']
```

リストの操作

```
>>> a = [0,1,2,3]
>>> a
[0, 1, 2, 3]
>>> a[2] = 8 # 値の置き換え
>>> a
[0, 1, 8, 3]
>>> a[1:3] = [5,5,5] # 置き換え
>>> a
[0, 5, 5, 5, 3]
>>> a[2:2] = [3,4] # 挿入
>>> a
[0, 5, 3, 4, 5, 5, 3]
>>> a[::-1] # 逆順
[3, 5, 5, 4, 3, 5, 0]
>>> a.append(4) # 末尾に追加
[0, 5, 3, 4, 5, 5, 3, 4]
>>> a.extend([5,6,7]) 末尾に追加
[0, 5, 3, 4, 5, 5, 3, 4, 5, 6, 7]
>>> a.pop(), a # 末尾を消去して, 末尾の値を返す
(7, [0, 5, 3, 4, 5, 5, 3, 4, 5, 6])
```

リストの変更

```
>>> a = [2,7,3,4,5,3,8]
>>> sorted(a), a
([2, 3, 3, 4, 5, 7, 8], [2, 7, 3, 4, 5, 3, 8])
>>> a.sort() # 昇順に並べ替え
>>> a
[2, 3, 3, 4, 5, 7, 8]
>>> sorted(a,reverse=True) # 逆順
[8, 7, 5, 3, 3, 2]
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(zip([1,2,3],[4,5,6])) # 同じ長さのリストをくっつける
[(1, 4), (2, 5), (3, 6)]
>>> list(enumerate(['a','b','c'])) # 何番目かとペアにする
[(0, 'a'), (1, 'b'), (2, 'c')]
>>> list(reversed([1,2,3]))
[3, 2, 1]
```

注: range, enumerate, reversed は主に for 文で使う (iterator)

リスト内包表記

```
>>> [x**2 for x in range(10)] # 9までの整数の平方のリスト
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [x for x in range(10) if x % 2 == 0] # 9までの偶数のリスト
[0, 2, 4, 6, 8]
>>> [x+y for x in [0,1,2] for y in [100,200,300]] # ネストさせた使い方
[100, 200, 300, 101, 201, 301, 102, 202, 302]

>>> a = [50,60,70,70,100]
>>> ave = sum(a)/len(a) # 平均
>>> ave
70.0
>>> sum([(x-ave)**2 for x in a])/len(a) # 分散
280.0
>>> sum([x**2/len(a) for x in a])-(sum(a)/len(a))**2 # 別の計算法
280.0
```

```
>>> x = set('abcda')
>>> y = set('bdxyz')
>>> x
set(['a', 'c', 'b', 'd'])
>>> 'b' in x
True
>>> 'e' in x
False
>>> x - y # 集合差
set(['a', 'c'])
>>> x | y # or
set(['a', 'c', 'b', 'd', 'y', 'x', 'z'])
>>> x & y # and
set(['b', 'd'])
>>> x ^ y # xor
set(['a', 'c', 'y', 'x', 'z'])
```

変更不能な集合は frozenset

- オブジェクトを一定の順序で並べたもの
- 変更を加えることができない

```
>>> t = (0,1,2,'three',(4,)) # (4) だと単なる整数となることに注意
>>> t[3]
'three'
>>> t[1:3]
(1,2)
>>> t[0]=1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> t*2 # 繰り返し
(0, 1, 2, 'three', (4), 0, 1, 2, 'three', (4,))
>>> len(t) # 長さ
5
>>> list(t) # リストに変換
[0, 1, 2, 'three', (4,)]
>>> tuple([0,1,2,3,4]) # リストをタプルに変換
(0, 1, 2, 3, 4)
```


key と value のペアの集合

```
>>> score = {'tanaka':62, 'matsui': 100, 'sato': 83, 'yamada': 62}
>>> score['matsui'] # [] を使ってアクセス（存在しない key だとエラー）
100
>>> score.get('matsui') # get メソッド使っても値を取得できる
100
>>> score.get('suzuki') # 存在しない場合は None が返り値
>>> score.get('suzuki',0) # いない場合の値を指定できる
0
>>> list(score.keys()) # key のリストを取得
['tanaka', 'matsui', 'sato', 'yamada']
>>> list(score.values()) # value のリストを取得
[62, 100, 83, 62]
>>> 'suzuki' in score # key の存在確認
False
>>> score['takahashi']=78 # 追加
>>> del score['yamada'] # key の削除
>>> score
{'tanaka':62, 'matsui': 100, 'sato': 83, 'takahashi': 78}
```

```
>>> # key と value のペアのリストからも作成できる
>>> score2 = dict([('tanaka',62), ('matsui', 100),
                  ('sato', 83), ('yamada', 62)])
>>> score2
{'tanaka':62, 'matsui': 100, 'sato': 83, 'yamada': 62}
>>> list(score2.items()) # .items でペアのリストにもできる
[('tanaka', 62), ('matsui', 100), ('sato', 83), ('yamada', 62)]
>>> {k:v+10 for k,v in score2.items()} # 辞書の内包表記
{'tanaka':72, 'yamada': 102, 'matsui': 110, 'sato': 93}
```

辞書の key としてリストや set は使えない．タプルや frozenset を使う．

- False: 0 や空のリスト, タプル, 辞書
- True: その他

```
>>> bool(0), bool(1), bool([]), bool([1,2])
(False, True, False, True)
>>> 2 and 3 # x and y: x が偽なら x, そうでなければ y
3
>>> 2 or 3  # x or y:  x が偽なら y, そうでなければ x
2
>>> not [] # not x: x が偽なら True, そうでなければ False
True
```

数学関数

```
>>> import math
>>> math.cos(math.pi/3)
0.50000000000000001
>>> math.log(1024,2)
10.0
>>> math.atan2(3,4) # atan2(y,x) は y/x の逆正接をラジアンで返す
0.6435011087932844
```

有理数

```
>>> from fractions import Fraction
>>> Fraction(5,6)+Fraction(7,4)
Fraction(31,12)
>>> r=Fraction(5,3)
>>> r.numerator # 分子
5
>>> r.denominator # 分母
3
>>> float(r)
1.6666666666666667
```

```
>>> import random
>>> random.random() # 値域 [0.0,1.0) のランダムな小数
0.48559804457756095
>>> random.uniform(-2.0,5.0) # 値域 [-2.0,5.0) のランダムな小数
1.5409831624784305
>>> random.randint(1,6) # 1,2,...,6 からランダムに選択
4
>>> l = [2,3,5,7,11,13]
>>> random.choice(l) # リストなどからランダムに 1 つ
7
>>> random.shuffle(l) # リストなどをシャッフル
>>> l
[5, 2, 13, 7, 11, 3]
>> random.sample(l,3) # 重複なくランダムサンプリング
[3, 5, 13]
```

if 文

```
x = 4
if x < 0:
    print('-')
elif x==0:
    print('0')
elif x>0:
    print('+')
```

インデントは半角空白 4 つ

#階乗計算

```
res = 1
```

```
for i in range(1,10): # 1,2,...,9
```

```
    res *= i
```

```
print(res)
```

```
# 362880
```

#素数列举

```
for n in range(2,10):
```

```
    for x in range(2,n):
```

```
        if n % x == 0:
```

```
            print(n, 'equals', x, '*', n//x)
```

```
            break
```

```
    else:
```

```
        print(n, 'is a prime number')
```

```
# 2 is a prime number ...
```

while 文

```
a, b = 0, 1
while b < 10:
    print(b, end=' ')
    a, b = b, a+b

# 1 1 2 3 5 8
```


関数の定義

```
def isprime(n):  
    for i in range(2,n):  
        if n%i ==0:  
            return False  
    return True
```

```
>>> isprime(7)  
True  
>>> isprime(8)  
False  
>>> p=isprime  
>>> p(5)  
True
```

関数名を付けることなく関数を作れる

```
>>> (lambda x: x**2+1)(3)
10
>>> f=(lambda a,b: 2*a+b)
>>> f(2,3)
7
>>> l=[1, 3, 4, 6]
>>> list(map(lambda x:x**2, l)) # l の各要素を 2 乗
[1, 9, 16, 36]
>>> list(filter(lambda x:x%2==0, l)) # l で偶数の要素だけを取り出す
[4, 6]
```

ファイルの入出力

読み込み

```
f = open('test.txt','r') # test.txt を読み込みモードとして開く
for line in f: # 1行ずつ読み込み
    print(line,end='') # 出力
f.close() # test.txt を閉じる
```

書き込み

```
f = open('test.txt','w') # test.txt を書き込みモードとして開く
f.write('hoge!\n') # test.txt に書き込み
f.close() # test.txt を閉じる
```

コマンドライン引数

test.py

```
import sys
```

```
param = sys.argv
```

```
print(param)
```

```
% python3 test.py hoge fuga
```

```
['test.py', 'hoge', 'fuga']
```

matrix.txt の形式

```
N M
a11 a12 a13 ... a1M
a21 a22 a23 ... a2M
a31 a32 a33 ... a3M
...
aN1 aN2 aN3 ... aNM
```

matrix.txt

```
3 4
1 2 3 4
3 4 7 8
0 2 3 9
```

matrix.py

```
N, M = map(int, input().split())
a = []
for i in range(N):
    a.append(list(map(int, input().split())))
print(a)
```

```
% python3 matrix.py < matrix.txt
[[1, 2, 3, 4], [3, 4, 7, 8], [0, 2, 3, 9]]
```

- try 節の中で例外が発生すると、その節の残りは飛ばされる
- 例外型が except の後で指定されているものに一致すると、except 節の中が実行される
- 一致しない場合は例外となり実行停止

```
while True:
    try:
        x = int(input('Please enter a number: '))
        break
    except ValueError:
        print('not a valid number: Try again')
print(x)
```

```
class Person:
    def __init__(self,name,age): # コンストラクタ
        self.name = name
        self.age = age
    def __str__(self): # 文字列化
        return self.name + ' is ' + str(self.age) + ' years old'
    def older(self):
        self.age+=1

k = Person('kawase',31)
print(k) # kawase is 31 years old
k.older()
print(k) # kawase is 32 years old
for i in range(100): k.older()
print(k) # kawase is 132 years old
```

解答プログラムをまとめたテキストファイルを作成して、OCW-iで提出

- ファイル名は `practice1.txt`
- 次回授業の開始時間が締め切り
(登録が間に合わないなどの場合は別途相談)
- ファイルの最初に学籍番号と名前を書く
- どの演習問題のプログラムかわかるように記述
- 出力結果もつける
- 途中までしかできなくても、どこまでできてどこができなかったかを書けば部分点を付けます

問 1

$1^2, 2^2, 3^2, \dots, 100^2$ の和を求めよ

問 2

$1, 2, 3, \dots, 1000$ のうち、3 か 5 の倍数になっているものの和を求めよ.

問 3

“abracadabra” と “bibbidi-bobbidi-boo” のどちらにも現れるアルファベットを全て答えよ.

問 4

各桁を 4 乗した数の和が元の数と一致する自然数を全て求めよ.

(ヒント : $9^4 = 6561$ なので 100000 より小さい数だけ考えればよい)