Pythonの組み込み関数をもっと知ろう！

辻 真吾

@tsjshg

自己紹介

- 1975年生まれ
- 都内のとある大学で研究やってることになってます
  - 研究室のテーマは癌とゲノム
  - 私がやっているのはPythonで主に機械学習を使ったデータ解析
- Udemyで「実践Pythonデータサイエンス」やってます
- AI活用時代にPythonで見る夢@CTC教育サービス
  - http://www.school.ctc-g.co.jp/columns/tsuji/
- http://www.tsjshg.info/

今日は新年1回目

- 年が明けると、毎年考えていることがあります
- 2017は素数（1とその数自身以外の約数がない数）か？
- 2017は素数（prime number）ちなみに29も
- 後ほど、これを1行で確認するPythonコードを

組み込み関数

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

https://docs.python.org/3/library/functions.html

len, open, print, range, str

まずは基本から

```
for i in range(4):
    print(i)
```

```
0
1
2
3
```

0から3まで（4つ）の数字を順次画面に出力

0から3までの数字を、output.txtに連続して書き込み

```
with open('output.txt', 'w') as f:
    for i in range(4):
        f.write(str(i))
```

書き込んだデータを読み込んで長さを表示

```
with open('output.txt') as f:
    data = f.read()
print(len(data))
```

```
4
```

ちなみに

```
data
```

```
'0123'
```

all, any

全部あるall, 何かあるany

```
all([0,0,1])
```

False

```
any([0,0,1])
```

True

```
all(['', 'a', ''])
```

False

```
any(['', 'a', ''])
```

True

```
all([[], ['a', 'b'], []])
```

False

```
any([[], ['a', 'b'], []])
```

True

```
all([1, 'a', ['a', 'b']])
```

True

全部入っている

```
any([0, '', []])
```

False

何も入っていない

繰り返しの時に便利

zip, enumerate

両方、タプルを返します

```
for u, v in zip([0,1,2], [2,1,0]):
    print('u = {}, v = {}'.format(u, v))
```

```
u = 0, v = 2
u = 1, v = 1
u = 2, v = 0
```

3つ以上引数を並べてもOK

```
for i, v in enumerate([9,8,7]):
    print('{} at index={}'.format(v, i))
```

```
9 at index=0
8 at index=1
7 at index=2
```

繰り返し処理で、
リストの場所が欲しいときに便利

オブジェクト操作系 その1

type, isinstance

typeとisinstance

```
type('python')
```

```
str
```

```
isinstance('python', str)
```

```
True
```

‘python’のデータ型を調べる
外部ライブラリを使っていて、「なんだっけ、これ？」という時に便利

```
class MyClass:
    pass
```

```
c = MyClass()
isinstance(c, MyClass)
```

```
True
```

データの型に応じて、処理を分けたりするときに便利

もちろん、自分で作ったデータ型にも使えます。

オブジェクト操作系 その2

dir, hasattr, getattr

属性を調べたり、使ったり

```
>>> s = 'python'
>>> dir(s)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__ge
tattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__iter__', '__le__', '__len__', '_
_lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
 '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', '
endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isid
entifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip'
, 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitl
ines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

```
s = 'python'
```

```
hasattr(s, 'split')
```

```
True
```

```
getattr(s, 'split')('t')
```

```
['py', 'hon']
```

```
s.split('t')
```

```
['py', 'hon']
```

文字列で、クラスの属性にアクセスできる

ちょっと、一息

割り算と小数

divmod, round

```
divmod(10, 3)
```

```
(3, 1)
```

10を3で割ると、商は3で剰余が1

```
divmod(10.1, 3)
```

```
(3.0, 1.0999999999999996)
```

余り1.1のはずが・・・

floatは、テキトーな小数

```
round(2.685, 2)
```

```
2.69
```

```
round(2.675, 2)
```

```
2.67
```

```
from decimal import Decimal
```

```
round(Decimal('2.675'), 2)
```

```
Decimal('2.68')
```

ちょっと関数型っぽいのを

mapとfilter

```
list(map(str, [1,2,3]))
```

```
['1', '2', '3']
```

リストとして表示するために、
組み込み関数listを使っています。

```
list(map(lambda x:'No. {}'.format(x), [1,2,3]))
```

```
['No. 1', 'No. 2', 'No. 3']
```

```
def no_form(x):
    return 'No. {}'.format(x)

list(map(no_form, [1,2,3]))
```

```
list(filter(lambda x: x%2==0, [0, 1, 2, 3, 4]))
```

```
[0, 2, 4]
```

条件に合う要素だけが残る

コードを動的に実行する

eval, exec

```
a = 1
```

```
eval('a + 1')
```

```
2
```

a + 1という式（expression)を評価して、2を得る

```
exec('a + 1')
```

a + 1というコードを実行する（だけ）

```
exec('a += 1')
```

aの値を変更するコードなので、これなら意味がある

```
a
```

```
2
```

コードの断片をメッセージとして受け取って、それを動的に実行することなどが可能

辞書のキーに出来るset

list, dict, setではなくfrozenset

```
s = set([1,1,2,3,4])
```

```
s
```

```
{1, 2, 3, 4}
```

普通のsetを用意

```
d = dict()
d[s] = 'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-57-f07f96906c49> in <module>()
      1 d = dict()
----> 2 d[s] = 'a'

TypeError: unhashable type: 'set'
```

setは中身を変更できる（mutable）
ので、辞書型のキーにはできない

```
fs = frozenset([1,1,2,3,4])
```

```
fs
```

```
frozenset({1, 2, 3, 4})
```

```
d = dict()
d[fs] = 'a'
```

```
d[fs]
```

```
'a'
```

辞書型のキーにできる

最後は、並べ替え

sorted

```
sorted(['2', '1', '10'])
```

```
['1', '10', '2']
```

文字列としてソート

```
sorted(['2', '1', '10'], key=int)
```

```
['1', '2', '10']
```

整数に変換してソート

```
sorted([(2, 'c'), (1, 'a'), (3, 'b')])
```

```
[(1, 'a'), (2, 'c'), (3, 'b')]
```

タプルは、頭でソートしてくれる

```
sorted([(2, 'c'), (1, 'a'), (3, 'b')], key=lambda x:x[1])
```

```
[(1, 'a'), (3, 'b'), (2, 'c')]
```

場所の指定も簡単

まとめ

2017が素数かどうか？

```
all([2017 % v for v in range(2, 2017)])
```

```
True
```

便利な組み込み関数を使っていきましょう！