1 もくじ

2 型変換系

- float
- list
- set
- dict
- int
- str
- bool
- tuple
- complex

3 進数変換

- oct
- bin
- hex

4 要素の真偽

全ての要素が真なら真 (all)

```
1 import random
2
3 a = [True] * 10
```

```
print(a)
print(all(a))

a = [True] * 5 + [False] + [True] * 4
print(a)
print(all(a))
```

```
[True, True, True, True, True, True, True, True, True, True]
True
[True, True, True, True, True, False, True, True, True, True]
False
```

いずれかの要素が真なら真 (any)

```
import random

a = [False] * 10
print(a)
print(any(a))

a = [True] * 5 + [False] + [True] * 4
print(a)
print(any(a))
```

```
[False, False, False, False, False, False, False, False, False, False]
False
[True, True, True, True, True, False, True, True, True, True]
True
```

5 サイズを図る

`len(iterator)`

引数のサイズを返す.
多次元配列であれば一番外側の次元数を返す.

6 算術系

divmod

`divmod(a, b)`

a / b の除法を行った時の商と剰余からなる対を返す.

sum

配列の総計を算出

```
1  import random
2
3  a = random.choices(range(100), k=15)
4  print(a)
5  print(sum(a))
```

```
[70, 80, 83, 43, 20, 85, 27, 50, 65, 94, 83, 45, 78, 23, 91]
937
```

abs

数値の絶対値を返す.
`complex` 型でも適用できる.

```
1  a = -2.1
2  print(abs(a))
3
4  c = 2.1 - 1.3j
5  print(abs(c))
```

```
2.1
2.4698178070456938
```

min/max

配列の最小/最大を返す.

```
1  import random
2
3  a = random.sample(range(100), k=15)
4  print(a)
```

```
print(min(a))
print(max(a))
```

```
[46, 76, 28, 30, 23, 31, 18, 84, 58, 43, 82, 59, 60, 61, 15]
15
84
```

pow

`pow(a, n)` で, a^nを返す.
もちろん, `a ** n` でもOK

round

`round(number[, ndigits])`

`number` の小数部を `ndigits` 桁に丸めた値を返す. `ndigits` が省略されていたり, `None` であったとき, 入力値に最も近い整数を返す.

```
print(round(2.4))
print(round(2.347081348, 0))
print(round(2.347081348, 1))
print(round(2.347081348, 2))
print(round(2.347081348, 3))
print(round(2.347081348, 4))
```

```
2
2.0
2.3
2.35
2.347
2.3471
```

7 利用しているオブジェクトについて

dir

引数がない場合, 現在のローカルスコープにある名前のリストを返す.
引数があれば, そのオブジェクトの有効な属性のリストを返そうと試みます.

locals

その位置にスコープを有するローカルオブジェクトの名前と, その内容の辞書を得ることができます.

以下は実行例です. (少し表示がバグってますが, スルーしてください.)

```
def func(x):
        return 3*(x - 2) + 4

def main():
        a = 10
        print(func(a))

        b = 20
        print(func(b))

        print("\n-- 以下, main最下部におけるlocalオブジェクト --")
        print(*locals().items(), sep='\n')


if __name__ == '__main__':
        main()
        print("\n-- 以下, if以下のlocalオブジェクト --")
        print(*locals().items(), sep='\n')
```

```
28
58


-- 以下, main最下部におけるlocalオブジェクト --
('b', 20)
('a', 10)

-- 以下, if以下のlocalオブジェクト --
('__name__', '__main__')
('__doc__', None)
('__package__', None)
('__loader__', <_frozen_importlib_external.SourceFileLoader object at 0x7fef3fe1be80>)
('__spec__', None)
('__annotations__', {})
('__builtins__', )
('__file__', '/home/haru/python_ws/document/matplotlib/pythonについて/C言語ならわかる人向け/6zcg3ovkn_code_chunk')
```

```
('__cached__', None)
('func', )
('main', )
```

globals

グローバルオブジェクトを取得し, その名前を `key` とし, 内容を `value` とした辞書を得ることができます.

```python
def func(x):
        return 3*(x - 2) + 4

def main():
        a = 10
        print(func(a))

        b = 20
        print(func(b))

        print("\n-- 以下, main最下部におけるlocalオブジェクト --")
        print(*locals().items(), sep='\n')
        print("\n-- 以下, main最下部におけるglobalオブジェクト --")
        print(*globals().items(), sep='\n')


if __name__ == '__main__':
        main()
```

```
28
58


-- 以下, main最下部におけるlocalオブジェクト --
('b', 20)
('a', 10)

-- 以下, main最下部におけるglobalオブジェクト --
('__name__', '__main__')
('__doc__', None)
('__package__', None)
('__loader__', <_frozen_importlib_external.SourceFileLoader object at 0x7fa976313e80>)
('__spec__', None)
```

```
('__annotations__', {})
('__builtins__', )
('__file__', '/home/haru/python_ws/document/matplotlib/pythonについて/C言語ならわかる人向け/dkl74i0x6_code_chunk')
('__cached__', None)
('func', )
('main', )
```

8 イテレータに対する操作

next

`next(iterator[, default])`

引数にあるイテレータの次の要素を取得する. もしイテレータが尽きていれば, `default` に設定した値を返す. もし `default` が設定されていなければ, `StopIteration` が返される.

zip

複数のイテレータをまとめます.

enumerate

イテレータの要素と同時に, インデックスを返します.

map

`map(function, iterable)`

各要素に `function` の操作を行い, 新たに配列を作成する.

```
1 a = [3, 1, 10, 4]
2 print(*map(lambda x: "hoge を {:2} 回 => ".format(x) + "hoge" * x, a), sep="\n")
```

```
hoge を  3 回 => hogehogehoge
hoge を  1 回 => hoge
hoge を 10 回 => hogehogehogehogehogehogehogehogehogehoge
hoge を  4 回 => hogehogehogehoge
```

filter

`filter(function, iterable)`

各要素に対して, `function` の操作を行い, 真を返す要素だけを取り出し, 新たな配列

sorted

`soerted(iterable)`

要素を並び替えた新たなリストを返す.

メソッドとして, `list.sort()` のような方法もあるので, 元のとは別に新たなリストを利用したいときは, こちらを利用する.

9 標準入出力

input

`input([prompt])`

引数 `prompt` が存在する場合, それが末尾の改行を除いて, 標準出力に書き出される.
その後, 入力から一行を読み込み, 文字列に変換(末尾の改行を除く)して, 返す.

print

`print(*objects, sep='', end='\n', file=sys.stdout, flush=False)`

`objects` を `sep` で区切りながら, テキストストリーム `file` に表示し, 最後に `end` を表示します.

10 Unicode文字

ord

1文字のUnicode文字を表す文字列に対し, その文字のUnicodeコードポイントを表す整数を返します.

```
1 print(ord('A'))
2 print(ord('a'))
```

```
65
97
```

chr

ordの逆. コードポイントを表す整数から, Unicode文字を返す.

```
1 print(chr(65))
2 print(chr(97))
```

```
A
a
```

11 ファイル操作

open

12 その他

hash

オブジェクトのハッシュ値を返す.

id

引数に与えられたオブジェクトの識別値を返します.
このオブジェクトの有効期間中は一意かつ定数であることが保証されています.

help

組み込みのヘルプシステムを起動します.

range

いつものです.

type

- `type(object)`
  引数が1つだけの場合, `object` の型を返す. `object.__class__` によって返されるものと同じオブジェクト.

- `type(name, bases, dict)`
  引数が3つの場合, 新しい型オブジェクトを返す. 本質的には `class` 文の動的な形式.

13 (自分は)使わない/知らん/説明がめんどい

- delattr
- memoryview
- setattr
- slice
- ascii

- object
- staticmethod
- eval
- breakpoint
- exec
- isinstance
- bytearray
- issubclass
- super
- bytes
- iter
- callable
- format
- property
- frozenset
- vars
- classmethod
- getattr
- repr
- compile
- reversed
- import
- hasattr