1 もくじ

2 型とクラス

先ほどまでの章では, 型について見てきました.
そのとき, 以下のようなプログラムが散見されたと思います.

```
hoge = 43

print( hoge.__class__.__name__ )
```

```
fuga = "string"
print( fuga.__class__.__name__ )
```

```
int
str
```

「`.__class__.__name__`」はフィールドですね. しかも「クラス.名前」と成ってます.
実は「型」と言ってきたものも, 実はクラスとして実装されています.

そして, これら各型にはとっても便利なメソッドがたくさん用意されています.
以下では各方における便利なメソッド/フィールドを紹介していきます.

3 数値

数値型はあまりメソッド/フィールドを使う印象ないので特に紹介するものはありませんが, 複素数だけいくつか紹介しておきます.

実部/虚部の取得

`real` (実部)と `imag` (虚部, imaginary)というフィールドを参照すれば良い.

```
c = 13.4 + 20.3j
print(c.real, c.imag)
```

```
13.4 20.3
```

複素共役を取得

`.conjugate()` メソッドを使う.

```
c = 12.3 + 32.3j
print(c.conjugate())
```

```
(12.3-32.3j)
```

複素数の大きさ(絶対値)を取得

`abs()` を使う. こちらは組み込みの関数で, メソッドではない.
一応複素数でよく使うので, メソッド/フィールドではないが, 載せておく.

```
c = 1. + 1.j
print( abs(c) )
```

```
1.4142135623730951
```

複素数の回転角を求める.

こちらは `math` モジュールを使う必要がある.

4 リスト

シーケンスは用意されているメソッドがかなり豊富にある.

リストへ新しい要素を入れる (append)

```
a = [0, 1, 2, 3, 4, 5]; print(a)
a.append(99); print(a)
```

```
[0, 1, 2, 3, 4, 5]
[0, 1, 2, 3, 4, 5, 99]
```

リストの中の要素を全て消す (clear)

```
a = [0, 1, 2, 3, 4, 5]; print(a)
a.clear(); print(a)
```

```
[0, 1, 2, 3, 4, 5]
[]
```

引数と一致する要素の個数を数える (count)

```
a = [0, 1, 2, 3, 3, 4]; print(a)
print(a.count(0))
print(a.count(3))
print(a.count(10))
```

```
[0, 1, 2, 3, 3, 4]
1
2
0
```

リストの拡張 (extend)

```
a = [0, 1, 2, 3, 4]; print(a)
a.extend([5, 6, 7]); print(a)
```

```
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 5, 6, 7]
```

一致する要素のインデックスの取得 (index)

`.index(探したい要素, start, end)` 一番最初に一致する要素のインデックスを取得できます.

```
import random
a = random.choices([0, 1, 2, 3], k=23); print(a)

print(a.index(0))
print(a.index(2))

b = [0, 1, 2, 3, 0, 1, 2, 3]
print(b.index(0, 2, 6))
```

```
[0, 3, 1, 0, 3, 2, 2, 0, 3, 0, 0, 1, 0, 1, 2, 0, 1, 1, 2, 3, 3, 1, 3]
0
5
4
```

ちなみに一致する要素がリスト中に存在しない場合, エラーがでます.

要素の挿入をする (insert)

`.insert(挿入する場所, 要素)` で, 要素を挿入できる.

```
a = ['a', 'b', 'c', 'd', 'e']
a.insert(3, 'hoge')
print(a)
```

```
['a', 'b', 'c', 'hoge', 'd', 'e']
```

要素のpop (pop)

`.pop(index)` で取り出す要素を指定します.「取り出し」なので, その値はリストからなくなります.

```
a = ['a', 'b', 'c', 'd', 'e']; print(a)
print(a.pop(2)); print(a)
print(a.pop(1)); print(a)
```

```
['a', 'b', 'c', 'd', 'e']
c
['a', 'b', 'd', 'e']
b
['a', 'd', 'e']
```

指定した要素の削除 (remove)

```
a = ['a', 'b', 'c', 'd', 'e']
a.remove('c'); print(a)
```

```
['a', 'b', 'd', 'e']
```

指定した要素が見つからない場合, エラーが発生する.

リストのソート (sort)

`.sort(key, reverse)` でソートできる.
`key` では比較の際に呼び出される関数を指定でき,
`reverse` は `True` だと逆順にソートされる.

```
import random
a = random.sample(range(0, 10), k = 10); print(a)
a.sort(); print(a)

print()
```

```
a = random.sample(range(0, 10), k = 10); print(a)
a.sort(reverse=True); print(a)
```

```
[8, 2, 1, 0, 9, 7, 6, 5, 4, 3]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[6, 3, 1, 7, 8, 9, 4, 5, 0, 2]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

5 文字列

文字列もなかなか良いメソッドがたくさんあるので, こういうやつあったな〜くらいに覚えておくと役立つ.

文字置換

`.replace("置換対象", "置換後")` のように指定する.

```
a = "hogefugapiyo"
a = a.replace("fuga", ":-)")
print(a)
```

```
hoge:-)piyo
```

文字検索

`.find("検索したい文字列")` で, 検索したい文字列があればその先頭位置を, なければ `-1` が返ってくる.

ちなみに複数個合っても一番先に見つかったもののインデックスのみ.

```
a = "hogefugapiyo"
print(a.find("gap"))
```

```
print(a.find("pot"))
```

```
6
-1
```

文字カウント

```
a = "hogefugapiyo"
print(a.count("o"))
```

```
2
```

その他

- 両端の文字列を消す(`strip()` , `lstrip()` , `rstrip()`)
- 全て大文字/小文字にする(`upper()` , `lower()`)

6 辞書

値の取得 (get)

`.get(key)` で `key` がその辞書に存在していればその値をとってくる.
もしない場合は `None` を返してくれる.
`[]` を利用した参照方法ではエラーが出るが, こちらではエラーにはならない.

```
a = {"up":10, "down":39, "right":30, "left":29}
print(a.get('up'))
print(a.get('lost'))
```

```
10
None
```

要素, キー, (キー, 要素)のビューを得る

```
a = {"up":10, "down":39, "right":30, "left":29}
print(a.values())
print(a.keys())
print(a.items())
```

```
dict_values([10, 39, 30, 29])
dict_keys(['up', 'down', 'right', 'left'])
dict_items([('up', 10), ('down', 39), ('right', 30), ('left', 29)])
```

その他

他にも色々ありますが, たぶんそこまで使わないので, 省略します.