オブジェクト

- 1-3. 論理・比較演算と条件分岐の基礎 — オブジェクト

オブジェクト指向

- **データと、そのデータを使った処理を一つのオブジェクトにまとめておこうという発想**
- データじたいが、自分自身は何をすることが出来るか（=**メソッド**）を持つイメージ
- 一番のメリット：**データを作ってしまえば、そのデータに対して出来ることを一つ一つコーディングする必要が無い**

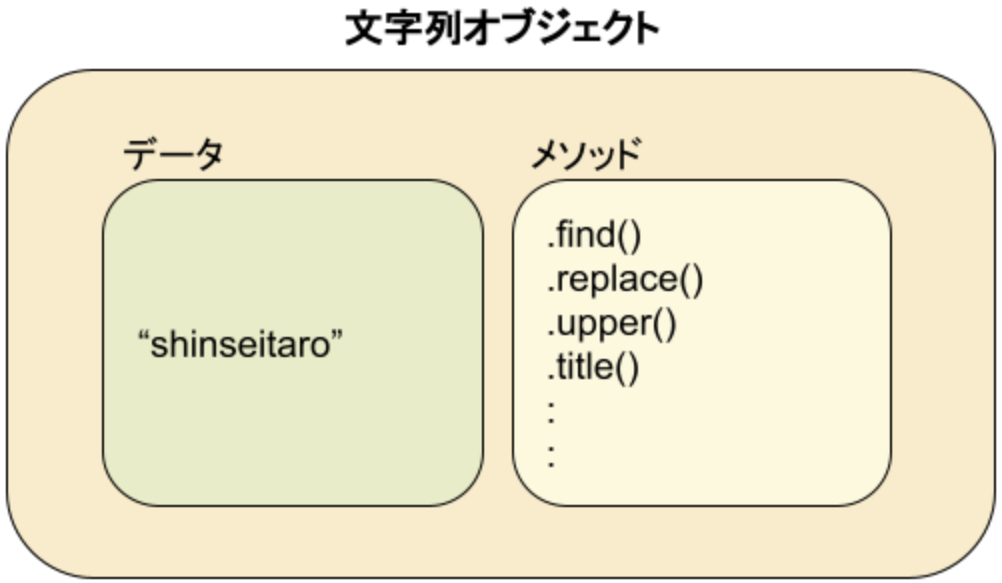
```
# s に文字列を格納
>>> s = "shinseitaro"
```

- s という変数に入れたのは "shinseitaro"という文字列をpython が評価したタイミングで、文字列オブジェクトが作成され、データは "shinseitaro", メソッドに文字列に対して出来る色々なことが実装された状態になる

```
# type関数で、s の型を調べる
>>> type(s)
<class 'str'>
```

```
# テキストシーケンス型 s が持つ アトリビュートを調べる.(長いので一部省略しています)
>>> dir(s)
['__add__',...,
'capitalize', 'casefold', 'center', 'count', ...,
'find', 'format', 'format_map', 'index', ...,
'isdigit', 'isidentifier', 'islower', 'isnumeric', ...,
'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition',
'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

- https://docs.python.org/ja/3/library/stdtypes.html#string-methods



- `s` は "shinseitaro" というデータを持つ
- データに対する処理は `.` で呼び出せる
- . で呼び出せる処理のことを **メソッド** という

```
# オブジェクトのデータを大文字で返す .upper() メソッド
>>> s.upper()
'SHINSEITARO'
```

メソッドと関数

- オブジェクトのデータに対して、オブジェクトが持つ処理命令を「**メソッド**」
- 引数として外から渡されたデータの処理命令を「**関数**」

見た目の違い

- メソッド
  - `データ.メソッド()`

```
s = "shinseitaro"
s.upper() # オブジェクトのデータを大文字で返すメソッド
s.replace("taro", "jiro") # オブジェクトのデータを置換して値を返すメソッド
```

- 関数
  - `関数名(引数)`

```
max([2,5,8]) # 渡されたデータで最大値を返す関数
len([2,5,8]) # 渡されたデータの個数を返す関数
str(12345) #  渡されたデータを文字列で返す関数
```

文

- 通常1行で書く命令

```
import os
return v
```

- メソッドや関数のように呼び出しための `()` がついていない命令
- メモ
  - python2系では print は文
  - python3系では print は関数

```
# 2.x
print "Hello"
```

```
# 3.x以降
print("Hello")
```

まとめ

- オブジェクトとは
  - データとメソッドをひとかたまりで持っている構造
  - 文字列や数値など python で評価(eval)したタイミングで全てオブジェクトに変わる
  - だから、ユーザーが定義しなくても組み込まれたメソッドが使える