第３回 Python勉強会

中沢研究室

2014/04/23

目次

- 前回分の振り返り +α
  - 座学 + 演習
  - 第1回でやるべきだったこと
- オブジェクト指向
  - オブジェクト指向とは
  - やってみよう
    - コードの解説
    - 詳細なコードの解説
- 演習
  - Even the last

前回分の振り返り +α

座学 + 演習

- FizzBuzz
- 関数の書き方・デフォルト引数・キーワード引数
- マジックコメント・if __name__ == '__main__':
- CheckiOに登録，使い方（FizzBuzz）

+α

- Pythonの仕様
  - 定数はありません

オブジェクト指向

オブジェクト指向とは

- オブジェクト　＝　車
- 関数　＝　車を構成する部品群

オブジェクト指向とは

- 【ゆっくり解説】 クラスの作り方 【プログラミング】

- http://www.nicovideo.jp/watch/sm23316188

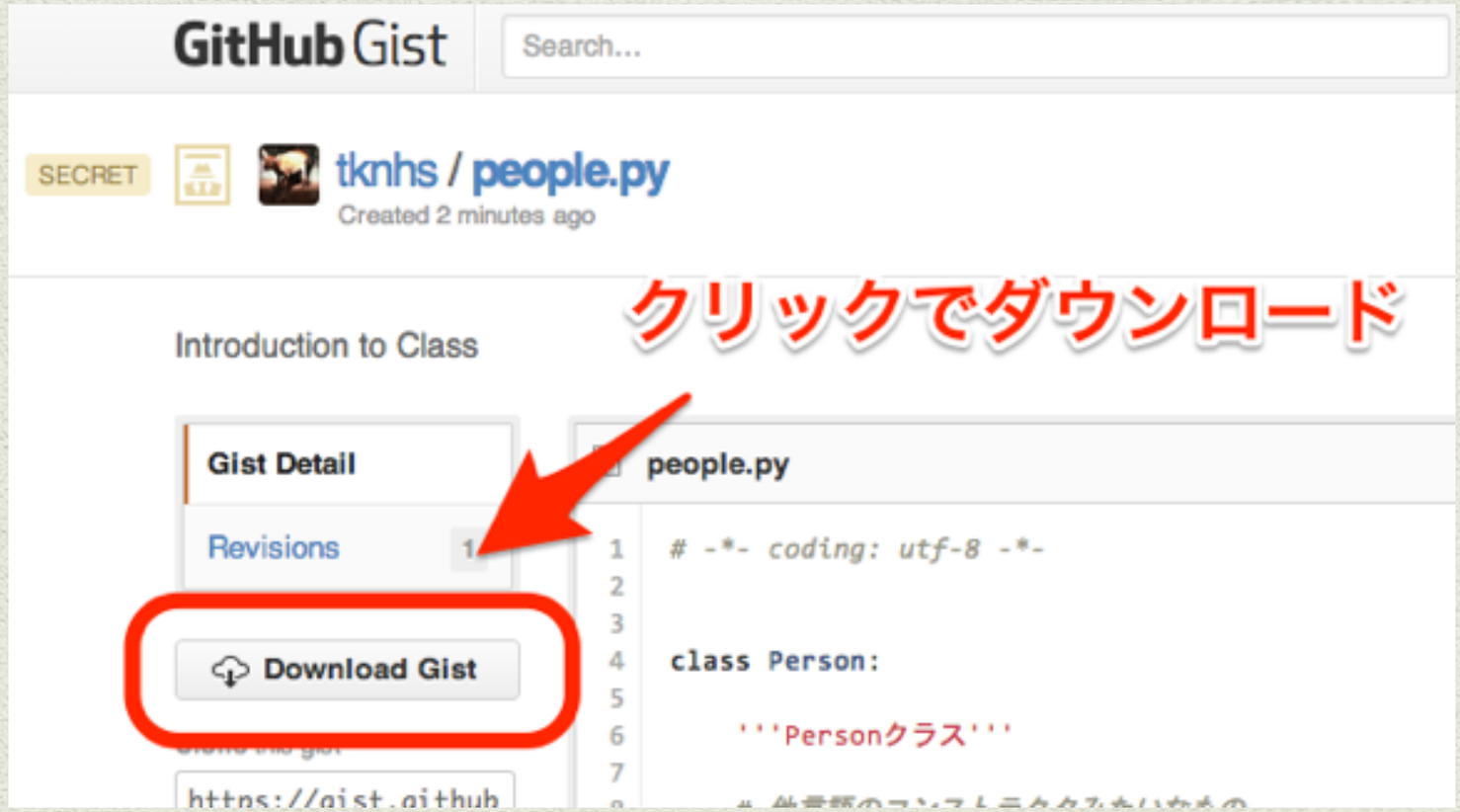
オブジェクト指向とは

- プログラミング勉強中の人にオブジェクト指向とは何なのかを何となく伝えたい話

- http://tdak.hateblo.jp/entry/20140406/1396773476

やってみよう

ダウンロード



- https://gist.github.com/tknhs/17d20eace59a28cedb10

コードの解説

クラス

```python
# -*- coding: utf-8 -*-

class Person:
    '''Personクラス'''
    def __init__(self, name, age, sex):
        self.name = name
        self.age = age
        self.__sex = sex

    def greet(self):
        print('Hello, my name is ' + self.name +
              ' and I am ' + str(self.age) + ' years old.')

if __name__ == '__main__':
    taro = Person('Taro', 25, 'Man')
    taro.greet()
```

クラス

インスタンス化

```
1 # -*- coding: utf-8 -*-
2
3 class Person:
4     '''Personクラス'''
5     def __init__(self, name, age, sex):
6         self.name = name
7         self.age = age
8         self.__sex = sex
9
10    def greet(self):
11        print('Hello, my name is ' + self.name +
12              ' and I am ' + str(self.age) + ' years old.')
13
14 if __name__ == '__main__':
15    taro = Person('Taro', 25, 'Man')
16    taro.greet()
17
```

インスタンス化

メソッド呼び出し

```
1 # -*- coding: utf-8 -*-
2
3 class Person:
4     '''Personクラス'''
5     def __init__(self, name, age, sex):
6         self.name = name
7         self.age = age
8         self.__sex = sex
9
10    def greet(self):
11        print('Hello, my name is ' + self.name +
12              ' and I am ' + str(self.age) + ' years old.')
13
14 if __name__ == '__main__':
15     taro = Person('Taro', 2
16     taro.greet()
17
```

メソッドを呼び出す

簡単でしょ？

詳細なコードの解説

クラス

```
1 # -*- coding: utf-8 -*-
2
3 class Person:
4     '''Personクラス'''
5     def __init__(self, name, age, sex):
6         self.name = name
7         self.age = age
8         self.__sex = sex
9
10    def greet(self):
11        print('Hello, my name is ' + self.name +
12              ' and I am ' + str(self.age) + ' years old.')
13
14 if __name__ == '__main__':
15     taro = Person('Taro', 25, 'Man')
16     taro.greet()
17
```

コンストラクタみたいなもの

クラス

```python
# -*- coding: utf-8 -*-

class Person:
    '''Personクラス'''
    def __init__(self, name, age, sex):
        self.name = name
        self.age = age
        self.__sex = sex

    def greet(self):
        print('Hello, my name is ' + self.name +
              ' and I am ' + str(self.age) + ' years old.')

if __name__ == '__main__':
    taro = Person('Taro', 25, 'Man')
    taro.greet()
```

インスタンス変数

プライベート
インスタンス変数

クラス

```
1 # -*- coding: utf-8 -*-
2
3 class Person:
4     '''Personクラス'''
5     def __init__(self, name, age, sex):
6         self.name = name
7         self.age = age
8         self.__sex = sex
9
10    def greet(self):
11        print('Hello, my name is ' + self.name +
12              ' and I am ' + str(self.age) + ' years old.')
13
14 if __name__ == '__main__':
15     taro = Person('Taro', 25, 'Man')
16     taro.greet()
17
```

メソッド

クラス

```
1 # -*- coding: utf-8 -*-
2
3 class Person:
4     '''Personクラス'''
5     def __init__(self, name, age, sex):
6         self.name = name
7         self.age = age
8         self.__sex = sex
9
10    def greet(self):
11        print('Hello, my name is ' + self.name +
12              ' and I am ' + str(self.age) + ' years old.')
13
14 if __name__ == '__main__':
15    taro = Person('Taro', 25, 'Man')
16    taro.greet()
17
```

インスタンス化

```
1 # -*- coding: utf-8 -*-
2
3 class Person:
4     '''Personクラス'''
5     def __init__(self, name, age, sex):
6         self.name = name
7         self.age = age
8         self.__sex = sex
9
10    def greet(self):
11        print('Hello, my name is ' + self.name +
12              ' and I am ' + str(self.age) + ' years old.')
13
14 if __name__ == '__main__':
15    taro = Person('Taro', 25, 'Man')
16    taro.greet()
```

変数 taro の属性

self.name = 'Taro'
self.age = 25
self.__sex = 'Man'

メソッド呼び出し

```
1 # -*- coding: utf-8 -*-
2
3 class Person:
4     '''Personクラス'''
5     def __init__(self, name, age, sex):
6         self.name = name
7         self.age = age
8         self.__sex = sex
9
10    def greet(self):
11        print('Hello, my name is ' + self.name +
12              ' and I am ' + str(self.age) + ' years old.')
13
14 if __name__ == '__main__':
15     taro = Person('Taro', 25, 'Man')
16     taro.greet()
```

変数 taro の属性

self.name = 'Taro'
self.age = 25
self.__sex = 'Man'

メソッドを呼び出す際は，初期化した属性を使う

オブジェクト

```
14 if __name__ == '__main__':
15     taro = Person('Taro', 25, 'Man')
16     print(taro)
17
18     import inspect
19     for data in inspect.getmembers(taro):
20         print(data)
21
22
23
24
```

変数 taro には オブジェクトが入っている

<__main__.Person object at 0x1098b8cf8>

オブジェクト

```
14 if __name__ == '__main__':
15     taro = Person('Taro', 25, 'Man')
16     print(taro)
17     print(dir(taro))
18
19
20
21
22 ['_Person__sex', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__',
23  '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__',
24  '__init__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__',
25  '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__',
    '__subclasshook__', '__weakref__', 'age', 'greet', 'name']
```

いろんな属性名が格納されている

オブジェクト

```
['_Person__sex', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__',
'__init__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', '__weakref__', 'age', 'greet', 'name']
```

インスタンス変数	メソッド
'_Person__sex'	'greet'
'age'	
'name'	

オブジェクト

```
14 if __name__ == '__main__':
15     taro = Person('Taro', 25, 'Man')
16     import inspect
17     for data in inspect.getmembers(taro):
18         print(data)
```

属性名と値

```
('_Person__sex', 'Man')
('__class__', <class '__main__.Person'>)
('__dict__', {'name': 'Taro', '_Person__sex': 'Man', 'age': 25})
('__doc__', 'Personクラス')
('age', 25)
('greet', <bound method Person.greet of <__main__.Person object at 0x107b62cf8>>)
('name', 'Taro')
```

オブジェクト

```
14 if __name__ == '__main__':
15     taro = Person('Taro', 25, 'Man')
16     print(taro.name)
17     print(taro.age)
18     print(taro.__sex)
19     print(taro._Person__sex)
20
21 taro.name          → Taro
22 taro.age           → 25
23 taro.__sex         → エラー
24 taro._Person__sex  → Man
```

クラスメソッド

```
3  class Person:
4      '''Personクラス'''
5      def __init__(self, name, age, sex):
6          self.name = name
7          self.age = age
8          self.__sex = sex
9
10     def greet(self):
11         print('Hello, my name is ' + self.name +
12               ' and I am ' + str(self.age) + ' years old.')
13
14     @classmethod
15     def pub_class_method(cls):
16         print('this is a public class method')
17
18     pub_class_value = 'this is a public value'
19
20
21 if __name__ == '__main__':
22     Person.pub_class_method()
23     print(Person.pub_class_value)
```

クラスメソッド

クラス変数

```
3  class Person:
4      '''Personクラス'''
5      def __init__(self, name, age, sex):
6          self.name = name
7          self.age = age
8          self.__sex = sex
9
10
11
12
13
14     @classmethod
15     def pub_class_method(cls):
16         print('this is a public class method')
17
18     pub_class_value = 'this is a public value'
19
20
21 if __name__ == '__main__':
22     Person.pub_class_method()
23     print(Person.pub_class_value)
```

クラス変数

継承

```
15 class SuperPerson(Person):
16     '''Personクラスの継承'''
17     def shout(self):
18         print('Hi, I am ' + self.name + '!!!!!!!!!!')
19
20
21 if __name__ == '__main__':
22     taro = Person('taro', 25, 'Man')
23     jiro = SuperPerson('jiro', 23, 'Man')
```

Personクラスを継承

継承

```
('_Person__sex', 'Man')
('__class__', <class '__main__.Person'>)
('__delattr__', <method-wrapper '__delattr__' of Person object at 0x1059afdd8>)
('__dict__', {'name': 'taro', '_Person__sex': 'Man', 'age': 25})
('__dir__', <built-in method __dir__ of Person object at 0x1059afdd8>)
('__doc__', 'Personクラス')
('__eq__', <method-wrapper '__eq__' of Person object at 0x1059afdd8>)
('__format__', <built-in method __format__ of
('__ge__', <method-wrapper '__ge__' of Person
('__getattribute__', <method-wrapper '__getat                                  fdd8>)
('__gt__', <method-wrapper '__gt__' of Person
('__hash__', <method-wrapper '__hash__' of Pe
('__init__', <bound method Person.__init__ of                                  8>>)
('__le__', <method-wrapper '__le__' of Person
('__lt__', <method-wrapper '__lt__' of Person
('__module__', '__main__')
('__ne__', <method-wrapper '__ne__' of Person object at 0x1059afdd8>)
('__new__', <built-in method __new__ of type object at 0x105679ea0>)
('__reduce__', <built-in method __reduce__ of Person object at 0x1059afdd8>)
('__reduce_ex__', <built-in method __reduce_ex__ of Person object at 0x1059afdd8>)
('__repr__', <method-wrapper '__repr__' of Person object at 0x1059afdd8>)
('__setattr__', <method-wrapper '__setattr__' of Person object at 0x1059afdd8>)
('__sizeof__', <built-in method __sizeof__ of Person object at 0x1059afdd8>)
('__str__', <method-wrapper '__str__' of Person object at 0x1059afdd8>)
('__subclasshook__', <built-in method __subclasshook__ of type object at 0x7f913145e4c8>)
('__weakref__', None)
('age', 25)
('greet', <bound method Person.greet of <__main__.Person object at 0x1059afdd8>>)
('name', 'taro')
```

taro

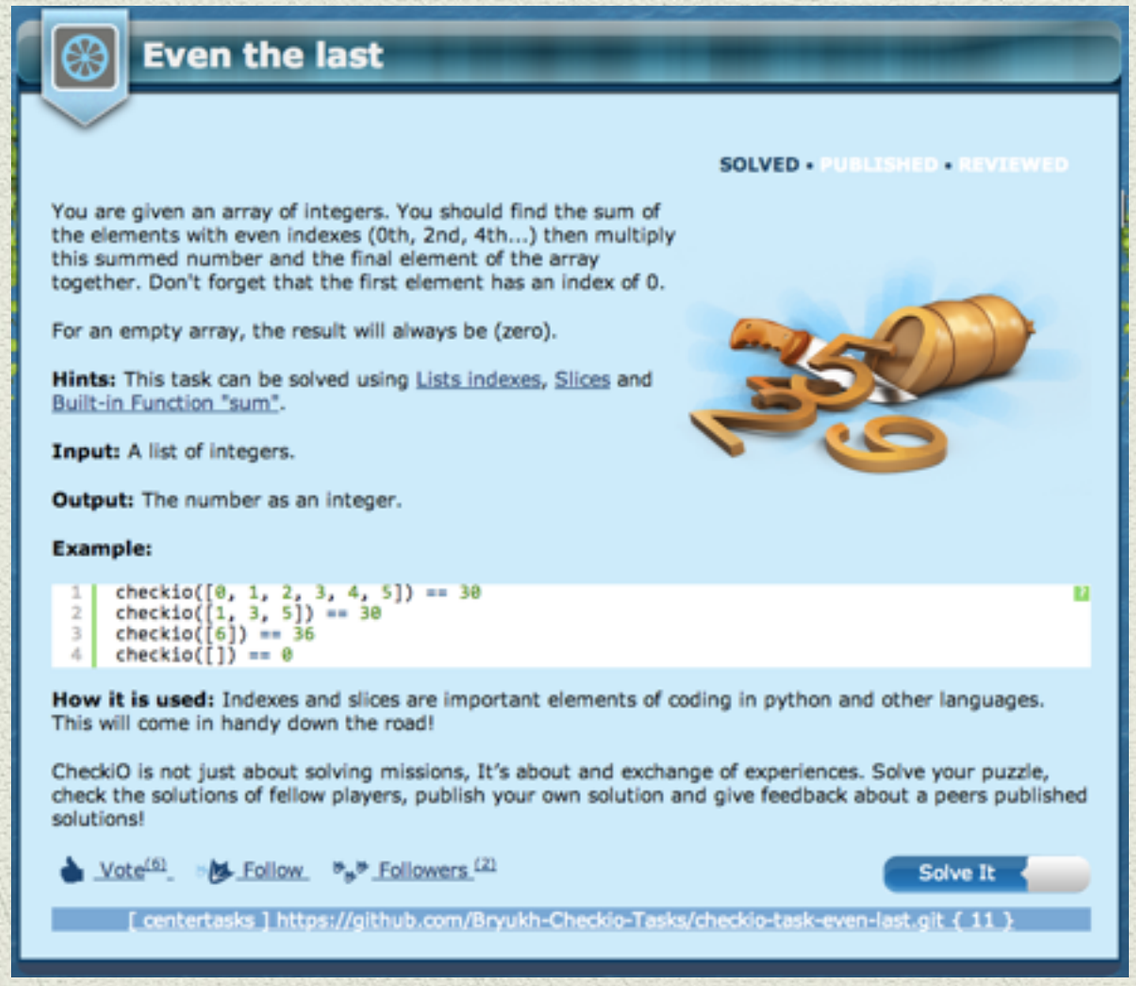
```
__.SuperPerson'>)
per '__delattr__' of SuperPerson object at 0x1059afe10>)
: 'Man', 'name': 'jiro', 'age': 23})
 __dir__ of SuperPerson object at 0x1059afe10>)
継承')
__eq__' of SuperPerson object at 0x1059afe10>)
hod __format__ of SuperPerson object at 0x1059afe10>)
__ge__' of SuperPerson object at 0x1059afe10>)
-wrapper '__getattri                                  59afe10>)
__gt__' of SuperPers
 '__hash__' of Super
uperPerson.__init__                                   1059afe10>>)
__le__' of SuperPers
__lt__' of SuperPers

__ne__' of SuperPers
 __new__ of type obj
('__reduce__', <built-in method __reduce__ of SuperPerson object at 0x1059afe10>)
('__reduce_ex__', <built-in method __reduce_ex__ of SuperPerson object at 0x1059afe10>)
('__repr__', <method-wrapper '__repr__' of SuperPerson object at 0x1059afe10>)
('__setattr__', <method-wrapper '__setattr__' of SuperPerson object at 0x1059afe10>)
('__sizeof__', <built-in method __sizeof__ of SuperPerson object at 0x1059afe10>)
('__str__', <method-wrapper '__str__' of SuperPerson object at 0x1059afe10>)
('__subclasshook__', <built-in method __subclasshook__ of type object at 0x7f913146f688>)
('__weakref__', None)
('age', 23)
('greet', <bound method SuperPerson.greet of <__main__.SuperPerson object at 0x1059afe10>>)
('name', 'jiro')
('shout', <bound method SuperPerson.shout of <__main__.SuperPerson object at 0x1059afe10>>)
```

jiro

演習

演習

- Even the last
  - http://www.checkio.org/mission/even-last/



Even the last

- ルール
  - 数字のリストが与えられる
  - 偶数番目の要素をすべて足した後，最後の要素を掛ける
  - 空リストの場合は 0 を返す
- ヒント
  - リスト：https://docs.python.org/3/tutorial/introduction.html#lists
  - スライス：https://docs.python.org/3/tutorial/introduction.html#strings
  - sum関数：https://docs.python.org/3/library/functions.html#sum