

@classmethod 多态总结

[toc]

Effective Python 24 条. 以 MapReduce 流程为例.

常规方法构建 MapReduce

```
1  import os
2  from threading import Thread
3
4
5  # InputData 基类
6  class InputData(object):
7      def read(self):
8          raise NotImplementedError
9
10
11 # InputData 具体子类
12
13 class PathInputData(InputData):
14     def __init__(self, path):
15         super().__init__()
16         self.path = path
17
18     def read(self):
19         return open(self.path).read()
20
21
22 # MapReduce 工作线程, 基类
23 class Worker(object):
24     def __init__(self, input_data):
25         # input_data 为 PathInputData的实例
26         self.input_data = input_data
27         self.result = None
28
29     def map(self):
30         raise NotImplementedError
31
32     def reduce(self):
33         raise NotImplementedError
34
35
36 # 子类, 换行符计数器
37 class LineCountWorker(Worker):
```

```

38     def map(self):
39         data = self.input_data.read()
40         self.result = data.count('\n')
41
42     def reduce(self, other):
43         self.result += other.result
44
45
46     """
47     需要手工写流程协调上面定义的各个组件
48     并实现 MapReduce
49     """
50
51
52     def generate_inputs(data_dir):
53         for name in os.listdir(data_dir):
54             yield PathInputData(os.path.join(data_dir, name))
55
56
57     def create_workers(input_list):
58         workers = []
59         for input_data in input_list:
60             workers.append(LineCountWorker(input_data))
61         return workers
62
63
64     def execute(workers):
65         threads = [Thread(target=w.map) for w in workers]
66         for thread in threads: thread.start()
67         for thread in threads: thread.join()
68
69         first, rest = workers[0], workers[1:]
70         for worker in rest:
71             first.reduce(worker)
72         return first.result
73
74
75     # 组合
76     def mapreduce(data_dir):
77         inputs = generate_inputs(data_dir)
78         workers = create_workers(inputs)
79         return execute(workers)
80
81
82     from tempfile import TemporaryDirectory, NamedTemporaryFile
83     import uuid
84
85
86     def write_test_file(tmpdir):

```

```

87     for i in range(100):
88         with open(os.path.join(tmpdir, "{}.txt".format(i)), 'w') as f:
89             f.write(str(uuid.uuid4()) + '\n')
90
91
92 def test():
93     with TemporaryDirectory() as tmpdir:
94         write_test_file(tmpdir)
95         result = mapreduce(tmpdir)
96         print('There are {} lines'.format(result))
97
98 if __name__ == '__main__':
99     test()

```

代码问题在于 MapReduce 不够通用, 每次都要手工写流程组合各个类模块. 因此引出 `@classmethod` 形式的多态.

@classmethod 形式的多态

`classmethod` 修饰符对应的函数不需要实例化, 不需要 `self` 参数, 但第一个参数需要是表示自身的 `cls` 参数, 可以用来调用类的属性, 类的方法, 实例化对象等。

```

1  import os
2  from threading import Thread
3
4
5  class GenericInputData(object):
6      def read(self):
7          raise NotImplementedError
8
9      # 第一个参数为 cls
10     @classmethod
11     def generate_inputs(cls, config):
12         raise NotImplementedError
13
14
15 class PathInputData(GenericInputData):
16     def __init__(self, path):
17         super().__init__()
18         self.path = path
19
20     def read(self):
21         return open(self.path).read()
22
23     @classmethod
24     def generate_inputs(cls, config):
25         data_dir = config['data_dir']
26         for name in os.listdir(data_dir):

```

```

27         # 类的多态, 以 cls 形式构造 PathInputData 对象
28         yield cls(os.path.join(data_dir, name))
29
30
31 # MapReduce 工作线程, 基类
32 class GenericWorker(object):
33     def __init__(self, input_data):
34         # input_data 为 PathInputData 的实例
35         self.input_data = input_data
36         self.result = None
37
38     def map(self):
39         raise NotImplementedError
40
41     def reduce(self):
42         raise NotImplementedError
43
44     @classmethod
45     def create_workers(cls, input_class, config):
46         workers = []
47         for input_data in input_class.generate_inputs(config):
48             workers.append(cls(input_data))
49         return workers
50
51
52 # 子类, 换行符计数器
53 class LineCountWorker(GenericWorker):
54     def map(self):
55         data = self.input_data.read()
56         self.result = data.count('\n')
57
58     def reduce(self, other):
59         self.result += other.result
60
61 # 组合
62 def execute(workers):
63     threads = [Thread(target=w.map) for w in workers]
64     for thread in threads: thread.start()
65     for thread in threads: thread.join()
66
67     first, rest = workers[0], workers[1:]
68     for worker in rest:
69         first.reduce(worker)
70     return first.result
71
72 def mapreduce(worker_class, input_class, config):
73     workers = worker_class.create_workers(input_class, config)
74     return execute(workers)
75

```

```
76
77 from tempfile import TemporaryDirectory, NamedTemporaryFile
78 import uuid
79
80
81 def write_test_file(tmpdir):
82     for i in range(100):
83         with open(os.path.join(tmpdir, "_{}.txt".format(i)), 'w') as f:
84             f.write(str(uuid.uuid4()) + '\n')
85
86
87 def test():
88     with TemporaryDirectory() as tmpdir:
89         write_test_file(tmpdir)
90         config = {'data_dir': tmpdir}
91         result = mapreduce(LineCountWorker, PathInputData, config)
92         print('There are {} lines'.format(result))
93
94
95 if __name__ == '__main__':
96     test()
```

此时, 需要给 `mapreduce` 函数传入更多的参数, 但不需要重新写辅助流程来组个各个模块了.