Python静态方法和类方法+反射

```
"""
python面向对象的静态方法和类方法

使用语法糖 @staticmethod 和  @classmethod

"""

class Dog(object):

    @staticmethod
    def run():
        print('狗在跑')

    # 这里第一个参数是cls,  表示调用当前的类名
    @classmethod
    def walk(cls):
        print(dir(cls))
        print('狗在走')


Dog.run()
Dog.walk()

```

反射----------

```
"""
python反射的用法
需要执行对象里的某个方法，或需要调用对象中的某个变量，
但是由于种种原因我们无法确定这个方法或变量是否存在，
这是我们需要用一个特殊的方法或机制要访问和操作这个未知的方法或变量
，这中机制就称之为反射


还有一个叫做魔法方法，参照下博客https://www.cnblogs.com/piperck/p/6354263.html
"""


class Person(object):

    def __init__(self,name):
        self.name = name

    def talk(self):
        print(self.name)

#

```

```
23
24 if __name__ == '__main__':
25     p = Person('wwj')
26     # 第一个方法 hasattr 判断对象中是否有这个方法或变量
27     print(hasattr(p,'name'))
28     print(hasattr(p,'talk'))
29     print(hasattr(p,'abc'))
30     # 第二个方法 getattr 获取对象中的方法或变量的内存地址
31     print(getattr(p,'name'))
32     t = getattr(p,'talk')
33     t()
34     print(getattr(p,'abc','没找到'))
35
36
37     """
38     余下2个方法为
39     setattr
40     为对象添加变量或方法
41
42 def abc(self):
43 print("%s正在交谈"%self.name)
44
45 class Person(object):
46     def __init__(self,name):
47         self.name = name
48
49 p = Person("laowang")
50 setattr(p,"talk",abc)   # 将abc函数添加到对象中p中，并命名为talk
51 p.talk(p)               # 调用talk方法，因为这是额外添加的方法，需手动传入对象
52
53
54 setattr(p,"age",30)     # 添加一个变量age,复制为30
55 print(p.age)
56
57     """
58
59     """
60     delattr
61     删除对象中的变量。注意：不能用于删除方法
62 class Person(object):
63 def __init__(self,name):
64         self.name = name
65 def talk(self):
66         print("%s正在交谈"%self.name)
67
68 p = Person("laowang")
69
70 delattr(p,"name")       # 删除name变量
71 print(p.name)           # 此时将报错
72
73     """
74
75
76
```