行为型模式-状态模式

概述：
    在软件开发过程中，应用程序可能会根据不同的情况作出不同的处理；最直接的解决方案是将这些所有可能发生的情况全都考虑到，然后使用if... ellse语句来做状态判断来进行不同情况的处理，但是对复杂状态的判断就显得"力不从心了". 随着增加新的状态或者修改一个状体(if else 语句的增多或者修改)可能会引起很大的修改，而程序的可读性 扩展性也会变得很弱. 维护也会很麻烦.

状态模式的缺点：
    1)状态模式的使用必然会增加系统类和对象的个数
    2)状态模式的结构与实现都较为复杂，如果使用不当将导致程序结构和代码的混乱.

示例代码：

In [1]:
```
#coding:utf8
from abc import ABCMeta, abstractmethod

class Base:
    __metaclass__ = ABCMeta
    @abstractmethod
    def run(self, value):
        pass

class Low(Base):
    def __init__(self):
        self.name = '较低占用率'
    def run(self, value):
        print('当前:%s 阈值:%s' %(self.name, value))
        print('没有具体行为!')

class Large(Base):
    def __init__(self):
        self.name = '较高占用率'
    def run(self, value):
        print("当前: %s 阈值: %s" %(self.name, value))
        print('发送邮件!')

class status:
    def __init__(self):
        self.value = 0.1
        self.low = Low()
        self.large = Large()
    def monitor(self):
        if self.value < 0.5:
            self.low.run(value=self.value)
        else:
            self.large.run(value=self.value)

if __name__ == '__main__':
    s = status()
    s.monitor()
    print('#' * 20)
    s.value = 0.7
    s.monitor()
```

```
当前:较低占用率 阈值:0.1
没有具体行为!
####################
当前: 较高占用率 阈值: 0.7
发送邮件!
```

In [ ]: