 初学Python的时候，对定义类的时候，为选择括号内是写object还是type而苦恼。在查看了Python的API后，总算明白了。现在总结如下:

    先来看object的说明:

    Python中关于object的说明很少，甚至只有一句话:

**Python代码**

1. **class** object
2. The most base type

    从介绍上看这也是Python对类型统一做出的努力。所以这里的object与Java的Object类有着异曲同工之妙，而且可以推测这个object很可能就是一个定义了一个类型的"空类"

    再来看type的说明:

**Python代码**

1. **class** type(object)
2. type(object) -> the object's type
3. type(name, bases, dict) -> a new type
5. Methods defined here:
7. \_\_call\_\_(...)
8. x.\_\_call\_\_(...) <==> x(...)
9. \_\_cmp\_\_(...)
10. x.\_\_cmp\_\_(y) <==> cmp(x,y)
11. \_\_delattr\_\_(...)
12. x.\_\_delattr\_\_('name') <==> **del** x.name
13. \_\_getattribute\_\_(...)
14. x.\_\_getattribute\_\_('name') <==> x.name
15. \_\_hash\_\_(...)
16. x.\_\_hash\_\_() <==> hash(x)
17. \_\_repr\_\_(...)
18. x.\_\_repr\_\_() <==> repr(x)
19. \_\_setattr\_\_(...)
20. x.\_\_setattr\_\_('name', value) <==> x.name = value
21. \_\_subclasses\_\_(...)
22. \_\_subclasses\_\_() -> list of immediate subclasses
23. mro(...)
24. mro() -> list
25. **return** a type's method resolution order

     从图上可以看出type类定义了很多方法，我们可以将它认为是一个帮助类吧。

     自从大家应该明白了type和object的区别吧。如果大家平时在定义类的时候，想尽可能的偷懒，可以直接从type继承，否则还是建议继承object吧

     另外贴上实际运行的区别吧:

   
 