
图解
Python
可能是迄今为止
最易懂的Python视频教程

```
"""类对象的特殊属性之__slots__"""

"""
    Python是动态语言，所以，在创建对象之后，可以对其动态地绑定属性和方法。
    如果想要对实例对象动态绑定的属性和方法的名称进行限制，可以在其对应的类对象中
定义特殊属性__slots__，并给__slots__赋值一个所有元素都为字符串的列表或元组，
这样，对实例对象动态绑定的属性和方法的名称就只能来自于__slots__中的元素。
"""
class MyClass(object):
    __slots__ = ("attr1", "do_sth1")

mc = MyClass()

mc.attr1 = 18

# mc.attr2 = 56   # AttributeError: 'MyClass' object has no attribute 'attr2'

def do_sth1(self):
    print("do_sth1被调用了")

from types import MethodType
mc.do_sth1 = MethodType(do_sth1, mc)
mc.do_sth1()

def do_sth2(self):
    print("do_sth2被调用了")

# AttributeError: 'MyClass' object has no attribute 'do_sth2'
# mc.do_sth2 = MethodType(do_sth2, mc)
```

```
"""
    默认情况下，访问实例对象的属性是通过访问该实例对象的特殊属性__dict__来实现的。例如：
访问obj.x其实访问的是obj.__dict__['x']。
    在类对象中定义了特殊属性__slots__后，其实例对象就不会在创建特殊属性__dict__了，
而是为每个属性创建一个描述器，访问属性时就会直接调用这个描述器。调用描述器比访问__dict__要快，
因此，在类对象中定义特殊属性__slots__可以提高属性的访问速度。
    此外，在类对象中定义了特殊属性__slots__后，由于其实例对象不再创建特殊属性__dict__，同时，
特殊属性__dict__是一个字典，字典的本质是哈希表，是一种用空间换取时间的数据结构，
因此，在类对象中定义特殊属性__slots__可以减少内存消耗。
"""
# AttributeError: 'MyClass' object has no attribute '__dict__'
# print(MyClass().__dict__)


"""
    特殊属性__slots__只对其所在类对象的实例对象起作用，对其所在类对象的子类的实例对象是不起作用的。
    如果子类也定义了特殊属性__slots__，那么子类的实例对象可以动态绑定的属性和方法的名称为
子类的__slots__加上父类的__slots__。
"""
class MyChildClass1(MyClass):
    pass

mcc1 = MyChildClass1()

mcc1.attr3 = 56
```

```python
class MyChildClass2(MyClass):
    __slots__ = ("attr2", "do_sth2")

mcc2 = MyChildClass2()

mcc2.attr1 = 18
mcc2.attr2 = 18
mcc2.do_sth1 = 18
mcc2.do_sth2 = 18

# AttributeError: 'MyChildClass2' object has no attribute 'attr3'
# mcc2.attr3 = 18
```