python复习6 面向对象2

1. __new__方法

__new__(cls, *args, **kwargs)

这是一个构造方法，创建并返回一个实例对象。

是一种特殊的负责创建类实例的静态方法（无需staticmethod修饰）。

该方法调用一次，就会有一个对象被创建。

该方法至少必须有一个参数cls，代表要实例化的类。

该方法必须要有返回值，返回其调用生成的实例。若__new__方法没有正确返回当前类cls的实例，那么__init__方法不会被调用。

__new__方法在__init__方法之前调用。

2. 单例模式

单例模式是一种设计模式，该模式确保类只有一个实例，并提供一个全局访问点以访问该实例。

单例模式可以使用__new__方法实现，也可以使用元类进行实现。

- 使用__new__方法实现

```
class Singleton:
    _instance = None

    def __new__(cls, *args, **kwargs):
        if not cls._instance:
            cls._instance = super(Singleton, cls).__new__(cls)
        return cls._instance
class MySingleton(Singleton):
    def __init__(self):
        print("Init...")
        pass

a = MySingleton()
b = MySingleton()
print(a is b) # True
```

- 使用元类实现

```python
def singleton(cls):
    instances = {}

    def _singleton(*args, **kw):
        if cls not in instances:
            instances[cls] = cls(*args, **kwargs)
        return instances[cls]
    return _singleton

@singleton
class TestSingleton(object):
    def __init__(self,num_sum):
        self.num_sum = num_sum

    def add(self):
        self.num_sum += 100

# 示例用法
instance1 = TestSingleton(3)
instance2 = TestSingleton(3)

print(instance1 is instance2)  # 应该打印: True
```

3. 协议编程

4. 描述器(Descriptor)协议

实现了__get__,__set__,__delete__中至少一种方法的对象，称为描述器。

描述器让对象可自定义属性查找、存储和删除操作。

描述器仅在作为一个类变量存储在另一个类中时才起作用，放入实例时则失效。

- `__get__(self, instance, owner)`

通过实例访问属性时调用，返回属性的值。

self是描述器实例，instance是访问属性的对象实例，owner是拥有属性的类

- `__set__(self, instance, value)`

通过实例设置属性的值时调用。

self是描述器实例，instance是设置属性的对象实例，value是要设置的新值

- `delete(self, instance)`

当通过实例删除属性时调用，用于删除属性。

self是描述器实例，instance是删除属性的对象实例

```python
class DescriptorExample:
    def __init__(self, initial_value=None):
        self._value = initial_value

    def __get__(self, instance, owner):
        print("Getting the value")
        return self._value

    def __set__(self, instance, value):
        print("Setting the value")
        self._value = value

    def __delete__(self, instance):
        print("Deleting the value")
        del self._value

class MyClass:
    my_attribute = DescriptorExample(42)

obj = MyClass()
print(obj.my_attribute)  # 调用 __get__ 方法，输出 "Getting the value" 和 42
obj.my_attribute = 100   # 调用 __set__ 方法，输出 "Setting the value"
del obj.my_attribute     # 调用 __delete__ 方法，输出 "Deleting the value"

```

5. 静态方法和类方法的Python实现

- StaticMethod

```

```

- ClassMethod

```

```