python中的new方法

1.创建类时先执行type的**init**方法

2.当一个类实例化时(创建一个对象)执行type的**call**方法，**call**方法的返回值就是实例化的对象

①**call**内部调用：

- 类.**new**方法，创建一个对象
- 类.**init**方法，初始化对象

②**new**() 方法的特性：

- **new**() 方法是在类准备将自身实例化时调用。
- **new**() 方法始终都是类的静态方法，即使没有被加上静态方法装饰器

3.实例化对象是谁取决于**new**方法,**new**返回什么就是什么【可以在一个类中重写父类object的**new**方法】

① 所有的类都继承自object（即所有类的父类都是object或者说object是所有新式类的基类）

② 如果（新式）类中没有重写**new**()方法，即在定义新式类时没有重新定义**new**()时，Python默认是调用该类的直接父类的**new**()方法来构造该类的实例，如果该类的父类也没有重写**new**()，那么将一直按此规矩追溯至object的**new**()方法，因为object是所有新式类的基类。

③如果要得到当前类的实例，应当在当前类中的 **new**() 方法语句中调用当前类的父类的 **new**() 方法。

例如，如果当前类是直接继承自 object，那当前类的 **new**() 方法返回的对象应该为：

```
def __new__(cls, *args, **kwargs):
    ...
    return object.__new__(cls) #传入参数是类对象，所以创建类的实例对象时（在不重写基类__new__方法的前提下），返回的就是是类的实例对象。
```

- **new**至少要有一个参数cls，代表要实例化的类，此参数在实例化时由Python解释器自动提供
- **new**必须要有返回值，返回实例化出来的实例，这点在自己实现**new**时要特别注意，可以return父类**new**出来的实例，或者直接是object的**new**出来的实例
- **init**有一个参数self，就是这个**new**返回的实例，**init**在**new**的基础上可以完成一些其它初始化的动作，**init**不需要返回值

4.例子：

```python
class Foo(object):
    pass
obj=Foo() #默认是调用该类的直接父类的__new__()方法来构造该类的实例
print(obj) #打印结果：<__main__.Foo object at 0x000002636FEAA208>

class F1(object):
    #重写__new__方法，返回这个重写的__new__方法
    def __new__(cls, *args, **kwargs):
        return 123
obj=F1() #实例化对象是谁取决于__new__方法，__new__返回什么就是什么
print(obj,type(obj)) #打印结果：123 <class 'int'>
```

```python
class F2(object):
    pass
class F3(object):
    def __new__(cls, *args, **kwargs):
        return F2()
obj=F3() #实例化对象是谁取决于__new__方法，__new__返回什么就是什么
print(obj) #<__main__.F2 object at 0x00000210119BA4A8>
```