Python new()与 init() 方法

一. __new__与__init__方法

__new__()是在新式类中新出现的方法，它作用在构造方法建造实例之前. 可以理解为，在Python中存在于类里面的构造方法__init__()负责将类实例化，而在 __init__()启动之前，__new__()决定是否要使用该__init__()方法，因为__new__()可以调用其他类的构造方法或者直接返回别的对象来作为本类的实例.

如果将类比喻为工厂，那么__init__()方法则是该工厂的生产工人，__init__()方法接受的初始化参数则是生产所需原料，__init__()方法会按照方法中的语句负责将原料加工成实例以供工厂出货. 而__new__()则是生产部经理，__new__()方法可以决定是否将原料提供给该生产部工人，同时它还决定着出货产品是否为该生产部的产品，因为这名经理可以借该工厂的名义向客户出售完全不是该工厂的产品.

1.2 __new__()方法的特性:

__new__() 方法是在类准备将自身实例化时调用，在实例化开始之后，在调用__init__()方法之前.

__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.

1.3 __new__()方法的第一个参数cls是当前正在实例化的类.

In [1]:
```
#示例代码:

class Foo(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __new__(cls, *args, **kwargs):
        return object.__new__(cls)
```

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. 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 return object.__new__(cls).

事实上如果(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.

而如果新式类中重写了__new__()方法，那么你可以自由选择任意一个的其他的新式类(因为只有新式类才有__new__()，因为所有新式类都是object的后代，而经典类是没有__new__()方法的，__new__()方法来制造实例，包括这个新式类的所有前代类和后代类，只要它们不会造成递归死循环.

注意:
1)在任何新式类的__new__()方法，不能调用自身的__new__()来制造实例，因为这会造成死循环.

In [4]:
```
#示例代码:（死循环代码）
class Foo(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __new__(cls, *args, **kwargs):
        return Foo.__new__(cls, *args, **kwargs)

    def __str__(self):
        return "a: %s, b: %s" %(self.a, self.b)

if __name__ == '__main__':
    x = Foo(1, 2)
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
<ipython-input-4-b34f3ad61825> in <module>
     12
     13 if __name__ == '__main__':
---> 14     x = Foo(1, 2)

<ipython-input-4-b34f3ad61825> in __new__(cls, *args, **kwargs)
      6
      7     def __new__(cls, *args, **kwargs):
----> 8         return Foo.__new__(cls, *args, **kwargs)
      9
     10     def __str__(self):

... last 1 frames repeated, from the frame below ...

<ipython-input-4-b34f3ad61825> in __new__(cls, *args, **kwargs)
      6
      7     def __new__(cls, *args, **kwargs):
----> 8         return Foo.__new__(cls, *args, **kwargs)
      9
     10     def __str__(self):

RecursionError: maximum recursion depth exceeded
```

2)使用object或者没有血缘关系的新式类的__new__()是安全的.

In [9]:
```
class Foo(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __new__(cls, *args, **kwargs):
        return object.__new__(cls)

    def __str__(self):
        return "a: %s, b: %s" %(self.a, self.b)


class Stranger(object):
    def __new__(cls, *args, **kwargs):
        return object.__new__(cls)

    def __init__(self, c):
        self.c = c

    def __str__(self):
        return "My args is %s" %self.c


if __name__ == '__main__':
    x = Foo(1, 2)
    print(x)
```

```
a: 1, b: 2
```

In [21]:
```
class Foo(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b
        print(self.a, self.b)

    def __new__(cls, *args, **kwargs):
        return object.__new__(Stranger) #Foo准备生产的是 Stranger __new__ 可以决定实例化的类是自身还是其他类(如: Stranger)

    def __str__(self):
        return "a: %s, b: %s" %(self.a, self.b)


class Stranger(object):
    def __new__(cls, *args, **kwargs):
        return object.__new__(cls)

    def __init__(self, c):
        self.c = c

    def __str__(self):
        return "My args is %s" %self.c


if __name__ == '__main__':
    x = Foo(1,2)
    print(x)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-21-36fb330d6be4> in <module>
     25 if __name__ == '__main__':
     26     x = Foo(1,2)
---> 27     print(x)

<ipython-input-21-36fb330d6be4> in __str__(self)
     20
     21     def __str__(self):
---> 22         return "My args is %s" %self.c
     23
     24

AttributeError: 'Stranger' object has no attribute 'c'
```

打印的结果显示foo其实是Stranger类的实例