Python 问题集

屈庆磊整理

quqinglei@icloud.com

2013 年 5 月

目录

1 单例模式

定义 简单的说就是一个类的对象在程序中只存在一份，比较难理解，必须从例子入手。

1.1 一个普通的类

```
# 定义一个类 X， 然后接下来看
>>> class X:
...     pass
...
>>> l = X() # l 是一个类的实例
>>> l.one = 'one' # 我们给 l 的 one 成员赋值
>>> l.one # 打印 l.one 的值
'one'
>>> m = X() # m 是一个新的实例
>>> m.one
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: X instance has no attribute 'one'
>>> print l, m
<__main__.X instance at 0xb742e1cc> <__main__.X instance at 0xb742e38c>
# 打印 x 和 y 的地址，从上面看是不同的，因为他们属于不同的实例
```

Listing 1: 一个普通的类和它的实例

1.2 单例模式示例一

```
>>> class X:
...     pass
...
>>> X = lambda singleton = X(): singleton
>>>
>>> x = X()
>>> x.one = 'one'
>>> x.one
'one'
>>> y = X()
>>> y.one
'one'
>>> print x, y
<__main__.X instance at 0xb742e26c> <__main__.X instance at 0xb742e26c>
# 打印 x 和 y 的地址，从上面看他们是相同的
```

Listing 2: 一个使用 lambda 的单实例的类和它的实例

我们从上面的第2页的代码框中可以看出类的所有实例都是那一个。而从第2页的代码框里可以看出，实例的内存地址是一样的，也就说明它们实际上是一个。

1.3 单例模式示例二

```
>>> def singleton(cls):
...     instance = {}
...     def get_instance():
...             if cls not in instance:
...                     instance[cls] = cls()
...             return instance[cls]
...     return get_instance
...
>>> @singleton
... class MyClass:
...     pass
...
>>> a = MyClass()
>>> b = MyClass()
>>> print a, b
<__main__.MyClass instance at 0xb6ac1aac> <__main__.MyClass instance at 0xb6ac1aac>
# 打印 x 和 y 的地址，从上面看他们是相同的
```

Listing 3: 单例模式示例二，采用 PEP318 的实现方式

1.4 单例模式示例三

```
class Singleton:
    """
    A non-thread-safe helper class to ease implementing singletons.
    This should be used as a decorator -- not a metaclass -- to the
    class that should be a singleton.

    The decorated class can define one `__init__` function that
    takes only the `self` argument. Other than that, there are
    no restrictions that apply to the decorated class.

    To get the singleton instance, use the `Instance` method. Trying
    to use `__call__` will result in a `TypeError` being raised.

    Limitations: The decorated class cannot be inherited from.

    """

    def __init__(self, decorated):
        self._decorated = decorated

    def Instance(self):
        """
        Returns the singleton instance. Upon its first call, it creates a
        new instance of the decorated class and calls its `__init__` method.
        On all subsequent calls, the already created instance is returned.

        """
        try:
            return self._instance
        except AttributeError:
            self._instance = self._decorated()
            return self._instance

    def __call__(self):
        raise TypeError('Singletons must be accessed through `Instance()`.')

    def __instancecheck__(self, inst):
        return isinstance(inst, self._decorated)


@Singleton
   class Foo:
       def __init__(self):
           print 'Foo created'

   f = Foo() # Error, this isn't how you get the instance of a singleton

   f = Foo.Instance() # Good. Being explicit is in line with the Python Zen
   g = Foo.Instance() # Returns already created instance

   print f is g # True
```

Listing 4: 单例模式示例三

2 类型介绍

2.1 大体简单的介绍

类型	描述	语法示例
str	A character string: an immutable sequence of Unicode codepoints.	`'Wikipedia', "Wikipedia", """Spanning multiple lines"""`
bytearray	A mutable sequence of bytes.	`bytearray(b'Some ASCII') bytearray(b"Some ASCII") bytearray([119, 105, 107, 105])`
bytes	An immutable sequence of bytes.	codeb'Some ASCII' b"Some ASCII" bytes([119, 105, 107, 105])
list	Mutable list, can contain mixed types.	[4.0, 'string', True]
tuple	Immutable, can contain mixed types.	(4.0, 'string', True)
set, frozenset	Unordered set, contains no duplicates. A frozenset is immutable.	4.0, 'string', True frozenset([4.0, 'string', True])
set, frozenset	Unordered set, contains no duplicates. A frozenset is immutable.	4.0, 'string', True frozenset([4.0, 'string', True])
dict	mutable associative array of key and value pairs.	`{'key1': 1.0, 3: False}`
int	An immutable integer of unlimited magnitude.[43]	`42`
float	An immutable floating point number (system-defined precision).	3.1415927
complex	immutable complex number with real and imaginary parts.	3+2.7j
bool	immutable truth value.	`True False`

表 1: python 类型，来自维基百科

3 排序

3.1 Python 字典排序

```
>>> lst = {'a': 100, 'b': 50, 'c': 1000}
>>> sorted(lst.items(), key = lambda x: x[1])
[('b', 50), ('a', 100), ('c', 1000)]
```