函数的文档

你在调用函数的时候，你像是函数这个产品的用户。

而你写一个函数，像是做一个产品，这个产品将来可能会被很多用户使用 —— 包括你自己。

产品，就应该有产品说明书，别人用得着，你自己也用得着 —— 很久之后的你，很可能把当初的各种来龙去脉忘得一干二净，所以也同样需要产品说明书，别看那产品曾经是你自己设计的。

Python 在这方面很用功，把函数的 “产品说明书” 当作语言内部的功能，这也是为什么 Python 有 Sphinx 这种工具，而绝大多数其他语言没有的原因之一罢。

Docstring

在函数定义内部，我们可以加上 **Docstring**；将来函数的 “用户” 就可以通过 `help()` 这个内建函数，或者 `.__doc__` 这个 Method 去查看这个 Docstring，即，该函数的 “产品说明书”。

先看一个 Docstring 以及如何查看某个函数的 Docstring 的例子：

```
def is_prime(n):
    """
    Return a boolean value based upon
    whether the argument n is a prime number.
    """
    if n < 2:
        return False
    if n == 2:
        return True
    for m in range(2, int(n**0.5)+1):
        if (n % m) == 0:
            return False
    else:
        return True


help(is_prime)
print(is_prime.__doc__)
is_prime.__doc__
```

```
Help on function is_prime in module __main__:

is_prime(n)
    Return a boolean value based upon
    whether the argument n is a prime number.


    Return a boolean value based upon
```

```
    whether the argument n is a prime number.

'\n    Return a boolean value based upon\n    whether the argument n is a prime
number.\n    '
```

Docstring 可以是多行字符串，也可以是单行字符串：

```
def is_prime(n):
    """Return a boolean value based upon whether the argument n is a prime
number."""

    if n < 2:
        return False
    if n == 2:
        return True
    for m in range(2, int(n**0.5)+1):
        if (n % m) == 0:
            return False
    else:
        return True


help(is_prime)
print(is_prime.__doc__)
is_prime.__doc__
```

```
Help on function is_prime in module __main__:

is_prime(n)
    Return a boolean value based upon whether the argument n is a prime number.

Return a boolean value based upon whether the argument n is a prime number.
'Return a boolean value based upon whether the argument n is a prime number.'
```

Docstring 如若存在，必须在函数定义的内部语句块的开头，也必须与其它语句一样保持相应的缩进（Indention）。Docstring 放在其它地方不起作用：

```
def is_prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    for m in range(2, int(n**0.5)+1):
        if (n % m) == 0:
            return False
    else:
```

```
        return True
    """
    Return a boolean value based upon
    whether the argument n is a prime number.
    """

help(is_prime)
print(is_prime.__doc__)
is_prime.__doc__
```

```
Help on function is_prime in module __main__:

is_prime(n)

None
```

书写 Docstring 的规范

规范，虽然是人们最好遵守的，但其实通常是很多人并不遵守的东西。

既然学，就要**像样** —— 这真的很重要。所以，非常有必要认真阅读 Python PEP 257 关于 Docstring 的规范。

简要总结一下 PEP 257 中必须掌握的规范：

> 1. 无论是单行还是多行的 Docstring，一概使用三个双引号扩起；
> 2. 在 Docstring 内部，文字开始之前，以及文字结束之后，都不要有空行；
> 3. 多行 Docstring，第一行是概要，随后空一行，再写其它部分；
> 4. 完善的 Docstring，应该概括清楚以下内容：参数、返回值、可能触发的错误类型、可能的副作用，以及函数的使用限制等等；
> 5. 每个参数的说明都使用单独的一行.....

由于我们还没有开始研究 Class，所以，关于 Class 的 Docstring 应该遵守什么样的规范就暂时略过了。然而，这种规范你总是要反复去阅读参照的。关于 Docstring，有两个规范文件：

> - PEP 257: Docstring Convensions
> - PEP 258: Docutils Design Specification

需要**格外注意**的是：

> Docstring 是**写给人看的**，所以，在复杂代码的 Docstring 中，写 **Why** 要远比写 *What* 更重要 —— 你先记住这点，以后的体会自然会不断加深。

Sphinx 版本的 Docstring 规范

Sphinx 可以从 `.py` 文件里提取所有 Docstring，而后生成完整的 Documentation。将来若是你写大型的项目，需要生成完善的文档的时候，你就会发现 Sphinx 是个 “救命” 的家伙，省时、省力、省心、省命.....

在这里，没办法一下子讲清楚 Sphinx 的使用，尤其是它还用它自己的一种标记语言，reStructureText，文件尾缀使用 `.rst`.....

但是，可以看一个例子：

```
class Vehicle(object):
    '''
    The Vehicle object contains lots of vehicles
    :param arg: The arg is used for ...
    :type arg: str
    :param `*args`: The variable arguments are used for ...
    :param `**kwargs`: The keyword arguments are used for ...
    :ivar arg: This is where we store arg
    :vartype arg: str
    '''
    def __init__(self, arg, *args, **kwargs):
        self.arg = arg
    def cars(self, distance, destination):
        '''We can't travel a certain distance in vehicles without fuels, so here's
the fuels
        :param distance: The amount of distance traveled
        :type amount: int
        :param bool destinationReached: Should the fuels be refilled to cover
required distance?
        :raises: :class:`RuntimeError`: Out of fuel
        :returns: A Car mileage
        :rtype: Cars
        '''
        pass

help(Vehicle)
```

```
Help on class Vehicle in module __main__:

class Vehicle(builtins.object)
 |  Vehicle(arg, *args, **kwargs)
 |
 |  The Vehicle object contains lots of vehicles
 |  :param arg: The arg is used for ...
 |  :type arg: str
 |  :param `*args`: The variable arguments are used for ...
 |  :param `**kwargs`: The keyword arguments are used for ...
 |  :ivar arg: This is where we store arg
 |  :vartype arg: str
 |
 |  Methods defined here:
 |
 |  __init__(self, arg, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  cars(self, distance, destination)
 |      We can't travel a certain distance in vehicles without fuels, so here's
```

```
the fuels
 |
 |      :param distance: The amount of distance traveled
 |      :type amount: int
 |      :param bool destinationReached: Should the fuels be refilled to cover
required distance?
 |      :raises: :class:`RuntimeError`: Out of fuel
 |
 |      :returns: A Car mileage
 |      :rtype: Cars
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

通过插件，Sphinx 也能支持 Google Style Docstring 和 Numpy Style Docstring。

以下两个链接，放在这里，以便你将来查询：

> - sphinx.ext.napoleon – Support for NumPy and Google style docstrings
> - sphinx.ext.autodoc – Include documentation from docstrings