7.1 可接受任意数量参数的函数¶

问题¶

你想构造一个可接受任意数量参数的函数。

解决方案¶

为了能让一个函数接受任意数量的位置参数，可以使用一个*参数。例如：

```
def avg(first, *rest):
    return (first + sum(rest)) / (1 + len(rest))

# Sample use
avg(1, 2) # 1.5
avg(1, 2, 3, 4) # 2.5
```

在这个例子中，rest是由所有其他位置参数组成的元组。然后我们在代码中把它当成了一个序列来进行后续的计算。

为了接受任意数量的关键字参数，使用一个以**开头的参数。比如：

```
import html

def make_element(name, value, **attrs):
    keyvals = [' %s="%s"' % item for item in attrs.items()]
    attr_str = ''.join(keyvals)
    element = '<{name}{attrs}>{value}</{name}>'.format(
                name=name,
                attrs=attr_str,
                value=html.escape(value))
    return element

# Example
# Creates '<item size="large" quantity="6">Albatross</item>'
make_element('item', 'Albatross', size='large', quantity=6)

# Creates '<p>&lt;spam&gt;</p>'
make_element('p', '<spam>')
```

在这里，attrs是一个包含所有被传入进来的关键字参数的字典。

如果你还希望某个函数能同时接受任意数量的位置参数和关键字参数，可以同时使用*和**。比如：

```
def anyargs(*args, **kwargs):
    print(args) # A tuple
    print(kwargs) # A dict
```

使用这个函数时，所有位置参数会被放到args元组中，所有关键字参数会被放到字典kwargs中。

讨论¶

一个*参数只能出现在函数定义中最后一个位置参数后面，而 **参数只能出现在最后一个参数。 有一点要注意的是，在*参数后面仍然可以定义其他参数

参数后面仍然可以是关其他参数。

```
def a(x, *args, y):
    pass

def b(x, *args, y, **kwargs):
    pass
```

这种参数就是我们所说的强制关键字参数，在后面7.2小节还会详细讲解到。