函数

IlleniumDillon

2024 年 6 月 3 日

定义函数使用关键字 def，后跟函数名与括号内的形参列表。函数语句从下一行开始，并且必须缩进。函数内的第一条语句是字符串时，该字符串就是文档字符串，也称为 docstring。

函数在执行时使用函数局部变量符号表，所有函数变量赋值都存在局部符号表中；引用变量时，首先，在局部符号表里查找变量，然后，是外层函数局部符号表，再是全局符号表，最后是内置名称符号表。因此，尽管可以引用全局变量和外层函数的变量，但最好不要在函数内直接赋值（除非是 global 语句定义的全局变量，或 nonlocal 语句定义的外层函数变量）。

1 默认值参数

函数定义时，可以给形参指定默认值，调用函数时，可以不传递默认值参数，这时，函数使用默认值参数。默认值参数必须放在非默认值参数后面。

注意：默认值参数在函数定义时，只计算一次，即在函数定义时，计算默认值参数的值，而不是在函数调用时计算默认值参数的值。

默认值只计算一次。默认值为列表、字典或类实例等可变对象时，会产生与该规则不同的结果。例如，下面的函数会累积后续调用时传递的参数：

不想在后续调用之间共享默认值时，应以如下方式编写函数：

```
def f(a, L=[]):
    L.append(a)
    return L
print(f(1))
print(f(2))
print(f(3))
# 输出: [1], [1, 2], [1, 2, 3]
```

```
def f(a, L=None):
    if L is None:
        L = []
    L.append(a)
    return L
```

2 关键字参数

函数调用时，可以使用关键字参数，即在调用函数时，指定形参的名称。关键字参数可以在任意顺序中使用，不需要按照函数定义的顺序。

函数调用时，关键字参数必须跟在位置参数后面。所有传递的关键字参数都必须匹配一个函数接受的参数。最后一个形参为 **name 形式时，接收一个字典，该字典包含与函数中已定义形参对应之外的所有关键字参数。**name 形参可以与 *name 形参组合使用（*name 必须在 **name 前面）， *name 形参接收一个元组，该元组包含形参列表之外的位置参数。例如，可以定义下面这样的函数：

关键字参数在输出结果中的顺序与调用函数时的顺序一致。

3 特殊参数

默认情况下，参数可以按位置或显式关键字传递给 Python 函数。为了让代码易读、高效，最好限制参数的传递方式，这样，开发者只需查看函数定义，即可确定参数项是仅按位置、按位置或关键字，还是仅按关键

```python
def f(a, b, *args, **kwargs):
    print(a, b)
    print(args)
    print(kwargs)
f(1, 2, 3, 4, 5, x=6, y=7)
# 输出: 1 2
# 输出: (3, 4, 5)
# 输出: {'x': 6, 'y': 7}
```

字传递。

```python
def f(pos1, pos2, /, pos_or_kwd, *, kwd1, kwd2):
#      -----------    ----------     ----------
#        |             |                  |
#        |        Positional or keyword   |
#        |                                - Keyword only
#         -- Positional only
```

函数定义中未使用 / 和 * 时，参数可以按位置或关键字传递给函数。

特定形参可以标记为仅限位置。仅限位置时，形参的顺序很重要，且这些形参不能用关键字传递。仅限位置形参应放在 / 前。/ 用于在逻辑上分割仅限位置形参与其它形参。如果函数定义中没有 /，则表示没有仅限位置形参。/ 后可以是位置或关键字或仅限关键字形参。

把形参标记为仅限关键字，表明必须以关键字参数形式传递该形参，应在参数列表中第一个仅限关键字形参前添加 *。说明：

- 使用仅限位置形参，可以让用户无法使用形参名。形参名没有实际意义时，强制调用函数的实参顺序时，或同时接收位置形参和关键字时，这种方式很有用。
- 当形参名有实际意义，且显式名称可以让函数定义更易理解时，阻止用户依赖传递实参的位置时，才使用关键字。

- 对于 API，使用仅限位置形参，可以防止未来修改形参名时造成破坏性的 API 变动。

4 任意实参列表

有时，函数需要接受任意数量的实参，这时，可以使用 *name 形参。*name 形参接收一个元组，在可变数量的实参之前，可能有若干个普通参数，这些普通参数是位置参数，不能用关键字传递。*name 形参可以与 **name 形参组合使用，但必须在 **name 形参之前。*name形参后面的所有形参都是关键字参数。

5 解包实参列表

函数调用要求独立的位置参数，但实参在列表或元组里时，要执行相反的操作。例如，内置的 range() 函数要求独立的 start 和 stop 实参。如果这些参数不是独立的，则要在调用函数时，用 * 操作符把实参从列表或元组解包出来：

```
args = [3, 6]
print(list(range(*args)))
# 输出: [3, 4, 5]
```

同样，字典可以用 ** 操作符传递关键字参数。

6 匿名函数

lambda 关键字用于创建小巧的匿名函数。Lambda 函数可用于任何需要函数对象的地方。在语法上，匿名函数只能是单个表达式。在语义上，它只是常规函数定义的语法糖。与嵌套函数定义一样，lambda 函数可以引用包含作用域中的变量。可以把匿名函数用作传递的实参。

7 文档字符串

以下是文档字符串内容和格式的约定。

第一行应为对象用途的简短摘要。为保持简洁，不要在这里显式说明对象名或类型，因为可通过其他方式获取这些信息（除非该名称碰巧是描述函数操作的动词）。这一行应以大写字母开头，以句点结尾。

文档字符串为多行时，第二行应为空白行，在视觉上将摘要与其余描述分开。后面的行可包含若干段落，描述对象的调用约定、副作用等。

8 函数注解

函数注解是关于用户定义函数的元数据信息。注解可以是任何表达式。注解存储在函数的 __annotations__ 属性中，是一个字典，键是参数名，值是注解。

形参标注的定义方式是在形参名后加冒号，后面跟一个会被求值为标注的值的表达式。返回值标注的定义方式是加组合符号 -> ，后面跟一个表达式，这样的校注位于形参列表和表示 def 语句结束的冒号。