函数基础

1

函数基础

——创建函数

def 语句

- 函数用def语句创建，语法如下：

```
def function_name(arguments):
    "function_documentation_string"
    function_body_suite
```

- 标题行由def关键字，函数的名字，以及参数的集合（如果有的话）组成
- def子句的剩余部分包括了一个虽然可选但是强烈推荐的文档字串，和必需的函数体

前向引用

- 函数不允许在函数未声明之前对其进行引用或者调用

```
def foo():
    print('in foo')
    bar()

foo()              #报错，因为bar没有定义
-----------------------------------------------------------
def foo():
    print('in foo')
    bar()

def bar():
    print('in bar’)
foo()              #正常执行，虽然bar的定义在foo定义后面
```

内部函数

- 在函数体内创建另外一个函数是完全合法的，这种函数叫做内部/内嵌函数

```
>>> def foo():
...         def bar():
...             print('bar() is called')
...         print('foo() is called’)
...         bar()
>>> foo()
foo() is called
bar() is called
>>> bar()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'bar' is not defined
```

2

函数基础

—— 调用函数

函数操作符

- 使用一对圆括号()调用函数，如果没有圆括号，只是对函数的引用
- 任何输入的参数都必须放置在括号中

```
>>> def foo():
...     print('Hello world!')
...
>>> foo()
Hello world!
>>> foo
<function foo at 0x7f18ce311b18>
```

关键字参数

- 关键字参数的概念仅仅针对函数的调用
- 这种理念是让调用者通过函数调用中的参数名字来区分参数
- 这样规范允许参数缺失或者不按顺序

```
>>> def get_info(name, age):
...     print("%s's age is %s" % (name, age))
...
>>> get_info(23, 'bob')
23's age is bob
>>> get_info(age = 23, name = 'bob')
bob's age is 23
```

参数组

- python允许程序员执行一个没有显式定义参数的函数
- 相应的方法是通过一个把元组（非关键字参数）或字典（关键字参数）作为参数组传递给函数

```
func(*tuple_grp_nonkw_args, **dict_grp_kw_args)
```

应用案例

- 编写一个简单的加减法数学游戏
  1. 随机生成两个100以内的数字
  2. 随机选择加法或是减法
  3. 总是使用大的数字减去小的数字
  4. 如果用户答错三次，程序给出正确答案

更多精彩...

http://bj.linux.tedu.cn/
企业QQ： 86198501