函数

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。函数能提高应用的模块性，和代码的重复利用率。Python提供了许多内建函数，比如print()。可以自己创建函数，这被叫做用户自定义函数。

定义一个函数

函数定义有以下准则

1. 函数代码块以 def 关键词开头，后接函数标识符名称和圆括号 ()。
2. 任何传入参数和自变量必须放在圆括号中间，圆括号之间可以用于定义参数。
3. 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
4. 函数内容以冒号起始，并且缩进。
5. return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回 None。

语法：def关键字

```
def 函数名（参数列表）:
    函数体
```

实例1:利用函数打印hello world

```
def hello():
    print("Hello World!")

hello()
```

输出结果：

```
Hello World!
```

实例2:函数计算面积

```
def area(width, height):
    return width * height

def print_welcome(name):
    print("Welcome", name)

print_welcome("Runoob")
w = 4
h = 5
print("width =", w, " height =", h, " area =", area(w, h))
```

输出结果：

```
Welcome Runoob
width = 4  height = 5  area = 20
```

函数调用

什么是函数的调用

定义一个函数：给了函数一个名称，指定了函数里包含的参数，和代码块结构。

这个函数的基本结构完成以后，你可以通过另一个函数调用执行，也可以直接从 Python 命令提示符执行。

如下实例调用了 printme() 函数：

实例1:简单的函数调用

```
# 定义函数
def printme( str ):
   # 打印任何传入的字符串
   print (str)
   return

# 调用函数
printme("我要调用用户自定义函数!")
printme("再次调用同一函数")
```

输出结果：

```
我要调用用户自定义函数!
再次调用同一函数
```

参数

以下是调用函数时可使用的正式参数类型：

- 必需参数
- 关键字参数
- 默认参数
- 不定长参数

必须参数

必需参数须以正确的顺序传入函数。调用时的数量必须和声明时的一样。调用 printme() 函数，必须传入一个参数，不然会出现语法错误：

实例1:必须参数的使用

```
#可写函数说明
def printme( str ):
   "打印任何传入的字符串"
   print (str)
   return

# 调用 printme 函数，不加参数会报错
printme()
```

输出结果：

```
Traceback (most recent call last):
  File "test.py", line 10, in <module>
    printme()
TypeError: printme() missing 1 required positional argument: 'str'
```

关键字参数

关键字参数和函数调用关系紧密，函数调用使用关键字参数来确定传入的参数值。使用关键字参数允许函数调用时参数的顺序与声明时不一致，因为 Python 解释器能够用参数名匹配参数值。

以下实例在函数 printme() 调用时使用参数名：

实例1:关键字参数的简单使用

```
def printme( str ):
   "打印任何传入的字符串"
   print (str)
   return

#调用printme函数
printme( str = "菜鸟教程")
```

输出结果：

```
菜鸟教程
```

实例2:参数传递不需要制定顺序

```
def printinfo( name, age ):
   "打印任何传入的字符串"
   print ("名字: ", name)
   print ("年龄: ", age)
   return

#调用printinfo函数
printinfo( age=50, name="runoob" )
```

输出结果：

```
名字:  runoob
年龄:  50
```

return语句

return [表达式] 语句用于退出函数，选择性地向调用方返回一个表达式。不带参数值的return语句返回None。之前的例子都没有示范如何返回数值，以下实例演示了 return 语句的用法：

实例1:return语句的简单使用

```
def sum( arg1, arg2 ):
   # 返回2个参数的和."
   total = arg1 + arg2
   print ("函数内 : ", total)
   return total

# 调用sum函数
total = sum( 10, 20 )
print ("函数外 : ", total)
```

输出结果：

```
函数内 :  30
函数外 :  30
```

默认参数

调用函数时，如果没有传递参数，则会使用默认参数。以下实例中如果没有传入 age 参数，则使用默认值：

实例1:默认参数的简单使用

```
def printinfo( name, age = 35 ):
   "打印任何传入的字符串"
   print ("名字: ", name)
   print ("年龄: ", age)
   return

#调用printinfo函数
printinfo( age=50, name="runoob" )
print ("------------------------")
printinfo( name="runoob" )
```

输出结果：

```
名字:  runoob
年龄:  50
------------------------
名字:  runoob
年龄:  35
```

输入输出

在输出中我们常用到的format()和%格式化输出，在输入中input()函数常被使用到。

输出格式美化：format函数

实例1:format函数的简单用法

```
print('{}网址: "{}!"'.format('菜鸟教程', 'www.runoob.com'))
```

输出结果：

```
菜鸟教程网址: "www.runoob.com!"
```

括号及其里面的字符 (称作格式化字段) 将会被 format() 中的参数替换。在括号中的数字用于指向传入对象在 format() 中的位置，如下所示：

实例2：格式化字段的使用

```
print('{0} 和 {1}'.format('Google', 'Runoob'))
print('{1} 和 {0}'.format('Google', 'Runoob'))
```

输出结果：

```
Google 和 Runoob
Runoob 和 Google
```

如果在 format() 中使用了关键字参数, 那么它们的值会指向使用该名字的参数。

实例3:使用关键字参数格式化输出

```
print('{name}网址: {site}'.format(name='菜鸟教程', site='www.runoob.com'))
```

输出结果：

```
菜鸟教程网址: www.runoob.com
```

旧的格式化输入：%操作符

% 操作符也可以实现字符串格式化。 它将左边的参数作为类似 sprintf() 式的格式化字符串, 而将右边的代入, 然后返回格式化后的字符串.

实例1：利用%进行格式化输出

```
pi = 3.1415926
# 5表示输出占5位，3表示保留3位小数
print('常量 PI 的值近似为：%5.3f。' % pi)
```

输出结果：

```
常量 PI 的值近似为：3.142。
```

注意：因为 format() 是比较新的函数， 大多数的 Python 代码仍然使用 % 操作符。但是因为这种旧式的格式化最终会从该语言中移除, 应该更多的使用 format()。

读取键盘输入

Python提供了 input() 内置函数从标准输入读入一行文本，默认的标准输入是键盘。input() 可以接收一个Python表达式作为输入，并将运算结果返回。

实例1:input()函数的简单使用

```
str = input("请输入：");
print ("你输入的内容是：", str)
```

输出结果：

```
请输入：菜鸟教程
你输入的内容是：  菜鸟教程
```