

PYTHON教程

1. 简介

2. Python历史

3. 安装Python

3.1. Python解释器

4. 第一个Python程序

4.1. 使用文本编辑器

4.2. 输入和输出

5. Python基础

5.1. 数据类型和变量

5.2. 字符串和编码

5.3. 使用list和tuple

5.4. 条件判断

5.5. 模式匹配

5.6. 循环

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

8. 函数式编程

9. 模块

10. 面向对象编程

11. 面向对象高级编程

12. 错误、调试和测试

13. IO编程

14. 进程和线程

15. 正则表达式

16. 常用内建模块

17. 常用第三方模块

18. 图形界面

19. 网络编程

20. 电子邮件

21. 访问数据库

22. Web开发

23. 异步IO

24. FAQ

25. 期末总结

[下载PDF](#)

定义函数



廖雪峰 GitHub 知乎 Twitter

资深软件开发工程师，业余马拉松选手。

在Python中，定义一个函数要使用 `def` 语句，依次写出函数名、括号、括号中的参数和冒号 `:`，然后，在缩进块中编写函数体，函数的返回值用 `return` 语句返回。

我们以自定义一个求绝对值的 `my_abs` 函数为例：

```
def my_abs(x):
    if x >= 0:
        return x
    else:
        return -x

print(my_abs(-99))
```

请自行测试并调用 `my_abs` 看看返回结果是否正确。

请注意，函数体内部的语句在执行时，一旦执行到 `return` 时，函数就执行完毕，并将结果返回。因此，函数内部通过条件判断和循环可以实现非常复杂的逻辑。

如果没有 `return` 语句，函数执行完毕后也会返回结果，只是结果为 `None`。`return None` 可以简写为 `return`。

在Python交互环境中定义函数时，注意Python会出现 `...` 的提示。函数定义结束后需要按两次回车重新回到 `>>>` 提示符下：

```
Windows PowerShell - □ x
>>> def my_abs(x):
...     if x >= 0:
...         return x
...     else:
...         return -x
...
>>> my_abs(-9)
9
>>>
```

如果你已经把 `my_abs()` 的函数定义保存为 `abstest.py` 文件了，那么，可以在该文件的当前目录下启动Python解释器，用 `from abstest import my_abs` 来导入 `my_abs()` 函数，注意 `abstest` 是文件名（不含 `.py` 扩展名）：

```
Windows PowerShell - □ x
>>> from abstest import my_abs
>>> my_abs(-9)
9
>>>
```

`import` 的用法在后续模块一节中会详细介绍。

空函数

如果想定义一个什么事也不做的空函数，可以用 `pass` 语句：

```
def nop():
    pass
```

`pass` 语句什么都不做，那有什么用？实际上 `pass` 可以用来作为占位符，比如现在还没想好怎么写函数的代码，就可以先放一个 `pass`，让代码能运行起来。

`pass` 还可以用在其他语句里，比如：

```
if age >= 18:
    pass
```

缺少了 `pass`，代码运行就会有语法错误。

PYTHON教程

- 1. 简介
- 2. Python历史
- 3. 安装Python
 - 3.1. Python解释器
- 4. 第一个Python程序
 - 4.1. 使用文本编辑器
 - 4.2. 输入和输出
- 5. Python基础
 - 5.1. 数据类型和变量
 - 5.2. 字符串和编码
 - 5.3. 使用list和tuple
 - 5.4. 条件判断
 - 5.5. 模式匹配
 - 5.6. 循环
 - 5.7. 使用dict和set
- 6. 函数
 - 6.1. 调用函数
 - 6.2. 定义函数
 - 6.3. 函数的参数
 - 6.4. 递归函数
- 7. 高级特性
- 8. 函数式编程
- 9. 模块
- 10. 面向对象编程
- 11. 面向对象高级编程
- 12. 错误、调试和测试
- 13. IO编程
- 14. 进程和线程
- 15. 正则表达式
- 16. 常用内建模块
- 17. 常用第三方模块
- 18. 图形界面
- 19. 网络编程
- 20. 电子邮件
- 21. 访问数据库
- 22. Web开发
- 23. 异步IO
- 24. FAQ
- 25. 期末总结

[下载PDF](#)

参数检查

调用函数时，如果参数个数不对，Python解释器会自动检查出来，并抛出 `TypeError`：

```
>>> my_abs(1, 2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: my_abs() takes 1 positional argument but 2 were given
```

但是如果参数类型不对，Python解释器就无法帮我们检查。试试 `my_abs` 和内置函数 `abs` 的差别：

```
>>> my_abs('A')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "<stdin>", line 2, in my_abs
TypeError: unorderable types: str() >= int()
>>> abs('A')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: bad operand type for abs(): 'str'
```

当传入了不恰当的参数时，内置函数 `abs` 会检查出参数错误，而我们定义的 `my_abs` 没有参数检查，会导致 `if` 语句出错，出错信息和 `abs` 不一样。所以，这个函数定义不够完善。

让我们修改一下 `my_abs` 的定义，对参数类型做检查，只允许整数和浮点数类型的参数。数据类型检查可以用内置函数 `isinstance()` 实现：

```
def my_abs(x):
    if not isinstance(x, (int, float)):
        raise TypeError('bad operand type')
    if x >= 0:
        return x
    else:
        return -x
```

添加了参数检查后，如果传入错误的参数类型，函数就可以抛出一个错误：

```
>>> my_abs('A')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "<stdin>", line 3, in my_abs
TypeError: bad operand type
```

错误和异常处理将在后续讲到。

返回多个值

函数可以返回多个值吗？答案是肯定的。

比如在游戏中经常需要从一个点移动到另一个点，给出坐标、位移和角度，就可以计算出新的坐标：

```
import math

def move(x, y, step, angle=0):
    nx = x + step * math.cos(angle)
    ny = y - step * math.sin(angle)
    return nx, ny
```

`import math` 语句表示导入 `math` 包，并允许后续代码引用 `math` 包里的 `sin`、`cos` 等函数。

然后，我们就可以同时获得返回值：

```
>>> x, y = move(100, 100, 60, math.pi / 6)
>>> print(x, y)
151.96152422706632 70.0
```

但其实这只是一种假象，Python函数返回的仍然是单一值：

PYTHON教程

- 1. 简介
- 2. Python历史
- 3. 安装Python
- 3.1. Python解释器
- 4. 第一个Python程序
- 4.1. 使用文本编辑器
- 4.2. 输入和输出
- 5. Python基础
- 5.1. 数据类型和变量
- 5.2. 字符串和编码
- 5.3. 使用list和tuple
- 5.4. 条件判断
- 5.5. 模式匹配
- 5.6. 循环
- 5.7. 使用dict和set
- 6. 函数
 - 6.1. 调用函数
 - 6.2. 定义函数
 - 6.3. 函数的参数
 - 6.4. 递归函数
- 7. 高级特性
- 8. 函数式编程
- 9. 模块
- 10. 面向对象编程
- 11. 面向对象高级编程
- 12. 错误、调试和测试
- 13. IO编程
- 14. 进程和线程
- 15. 正则表达式
- 16. 常用内建模块
- 17. 常用第三方模块
- 18. 图形界面
- 19. 网络编程
- 20. 电子邮件
- 21. 访问数据库
- 22. Web开发
- 23. 异步IO
- 24. FAQ
- 25. 期末总结

[下载PDF](#)

```
>>> r = move(100, 100, 60, math.pi / 6)
>>> print(r)
(151.96152422706632, 70.0)
```

原来返回值是一个tuple! 但是，在语法上，返回一个tuple可以省略括号，而多个变量可以同时接收一个tuple，按位置赋给对应的值，所以，Python的函数返回多值其实就是返回一个tuple，但写起来更方便。

练习

请定义一个函数 `quadratic(a, b, c)`，接收3个参数，返回一元二次方程 $ax^2 + bx + c = 0$ 的两个解。

提示：

一元二次方程的求根公式为：

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

计算平方根可以调用 `math.sqrt()` 函数：

```
>>> import math
>>> math.sqrt(2)
1.4142135623730951
```

```
import math

def quadratic(a, b, c):
    pass

# 测试:
print('quadratic(2, 3, 1) =', quadratic(2, 3, 1))
print('quadratic(1, 3, -4) =', quadratic(1, 3, -4))

if quadratic(2, 3, 1) != (-0.5, -1.0):
    print('测试失败')
elif quadratic(1, 3, -4) != (1.0, -4.0):
    print('测试失败')
else:
    print('测试成功')
```

参考源码

`def_func.py`

小结

定义函数时，需要确定函数名和参数个数；

如果有必要，可以先对参数的数据类型做检查；

函数体内部可以用 `return` 随时返回函数结果；

函数执行完毕也没有 `return` 语句时，自动 `return None`。

函数可以同时返回多个值，但其实就是一个tuple。

[« 调用函数](#)

[函数的参数 »](#)

 gitee | 企业版

一站式 DevOps 研发效能平台

灵活选择部署方式 | 支持 SaaS 在线使用 | 私有化部署

[进入 Gitee 官网](#)

Comments

Loading comments...

PYTHON教程

- 1. 简介
- 2. Python历史
- 3. 安装Python
 - 3.1. Python解释器
- 4. 第一个Python程序
 - 4.1. 使用文本编辑器
 - 4.2. 输入和输出
- 5. Python基础
 - 5.1. 数据类型和变量
 - 5.2. 字符串和编码
 - 5.3. 使用list和tuple
 - 5.4. 条件判断
 - 5.5. 模式匹配
 - 5.6. 循环
 - 5.7. 使用dict和set
- 6. 函数
 - 6.1. 调用函数
 - 6.2. 定义函数
 - 6.3. 函数的参数
 - 6.4. 递归函数
- 7. 高级特性
- 8. 函数式编程
- 9. 模块
- 10. 面向对象编程
- 11. 面向对象高级编程
- 12. 错误、调试和测试
- 13. IO编程
- 14. 进程和线程
- 15. 正则表达式
- 16. 常用内建模块
- 17. 常用第三方模块
- 18. 图形界面
- 19. 网络编程
- 20. 电子邮件
- 21. 访问数据库
- 22. Web开发
- 23. 异步IO
- 24. FAQ
- 25. 期末总结

 下载PDF