

[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](#)

我们知道圆的面积计算公式为：

$$S = \pi r^2$$

当我们知道半径 `r` 的值时，就可以根据公式计算出面积。假设我们需要计算3个不同大小的圆的面积：

```
r1 = 12.34
r2 = 9.08
r3 = 73.1
s1 = 3.14 * r1 * r1
s2 = 3.14 * r2 * r2
s3 = 3.14 * r3 * r3
```

当代码出现有规律的重复的时候，你就需要当心了，每次写 `3.14 * x * x` 不仅很麻烦，而且，如果要把 `3.14` 改成 `3.14159265359` 的时候，得全部替换。

有了函数，我们就不再每次写 `s = 3.14 * x * x`，而是写成更有意义的函数调用 `s = area_of_circle(x)`，而函数 `area_of_circle` 本身只需要写一次，就可以多次调用。

基本上所有的高级语言都支持函数，Python也不例外。Python不但能非常灵活地定义函数，而且本身内置了很多有用的函数，可以直接调用。

抽象

抽象是数学中非常常见的概念。举个例子：

计算数列的和，比如：`1 + 2 + 3 + ... + 100`，写起来十分不方便，于是数学家发明了求和符号 Σ ，可以把 `1 + 2 + 3 + ... + 100` 记作：

$$\sum_{n=1}^{100} n$$

这种抽象记法非常强大，因为我们看到 Σ 就可以理解成求和，而不是还原成低级的加法运算。

而且，这种抽象记法是可扩展的，比如：

$$\sum_{n=1}^{100} (n^2 + 1)$$

还原成加法运算就变成了：

$$(1 \times 1 + 1) + (2 \times 2 + 1) + (3 \times 3 + 1) + \dots + (100 \times 100 + 1)$$

可见，借助抽象，我们才能不关心底层的具体计算过程，而直接在更高的层次上思考问题。

写计算机程序也是一样，函数就是最基本的一种代码抽象的方式。

[« 使用dict和set](#)[调用函数 »](#)

Comments

Comments loaded. To post a comment, please [Sign In](#)

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



/🍡芋泥小饼 @ 2025/12/10 23:21:46

老师捉虫，最后那段，求和符号优先级比+高，所以 n^2+1 要小括号



廖雪峰 @ 2025/12/11 21:42:43



吃西瓜不吐籽 @ 2025/10/27 10:22:50

```
PI=3.1415926 def area_of_circle(r): if r<0: raise ValueError("输入不能为负数") return PI*r**2
r=float(input("输入半径:")) area=area_of_circle(r) print(f"半径为{r:.2f}\n圆的面积为{area:.2f}")
```



學不懂Fourier @ 2025/10/2 04:33:33

借助抽象，我们才能不关心底层的具体计算过程，而直接在更高的层次上思考问题。



狼图腾-崛起 @ 2025/9/28 09:54:55

内置函数就是常用的约定俗成的一些工具，调用函数就是类似于调用工具，工具的用法可以类比成函数的用法。



Magnolia @ 2025/9/10 03:54:51

```
def area_of_circle(x):
    x=float(x)
    s=3.14*x*x
    return s

x=input('Please enter the radius: ')
s=area_of_circle(x)
print(f'The area of circle is {s:.2f}.')
```



失眠的树 @ 2025/9/1 08:18:28

```
PI = 3.1415926535897931
r =float(input('请输入圆的半径: '))

def area_of_circle(r):
    if r < 0:
        raise ValueError('半径不能为负数')
    else:
        return PI * r * r
print('半径为%.2f的圆的面积为%.2f' % (r, area_of_circle(r)))
```



钟馗 @ 2025/8/13 19:18:41

廖老师大才，讲的通透。



小π @ 2025/8/13 02:26:31

打卡



@ @ 2025/8/7 05:11:28

打卡



123 @ 2025/7/28 20:13:45

打卡

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



... @ 2025/7/26 05:01:18

```
PI = 3.14159 r = 10 def area_of_circle(r): if r > 0: return PI * r ** 2 else: return None
print(area_of_circle(r))
```



未至 @ 2025/7/25 06:31:39

大卡



CD @ 2025/7/25 01:51:08

打卡



Stepbystep @ 2025/7/16 03:13:24

打卡打卡



群青 @ 2025/7/11 04:17:12

这个抽象的举例好通俗易懂



☁ @ 2025/7/9 02:31:19

Day 4



钟 @ 2025/7/8 11:33:25

打卡



市井小民 @ 2025/7/5 22:38:39

打卡



南安 @ 2025/7/4 07:59:18

di di



😬 @ 2025/5/22 02:15:42

day4



晴雅 @ 2025/7/1 02:49:56

哈哈，你很有毅力

©liaoxuefeng.com - 微博 - GitHub - License