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
7.1. 切片
7.2. 迭代
7.3. 列表生成式
7.4. 生成器
7.5. 迭代器
8. 函数式编程
8.1. 高阶函数
8.1.1. map/reduce
8.1.2. filter
8.1.3. sorted
8.2. 返回函数
8.3. 匿名函数
8.4. 装饰器
8.5. 偏函数
9. 模块
9.1. 使用模块
9.2. 安装第三方模块
10. 面向对象编程
10.1. 类和实例
10.2. 访问限制
10.3. 继承和多态
10.4. 获取对象信息
10.5. 实例属性和类属性
11. 面向对象高级编程
11.1. 使用__slots__
11.2. 使用@property
11.3. 多重继承
11.4. 定制类
11.5. 使用枚举类
11.6. 使用元类
12. 错误、调试和测试
12.1. 错误处理
12.2. 调试
12.3. 单元测试
12.4. 文档测试
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

函数式编程

廖雪峰
资深软件开发工程师，业余马拉松选手。

函数是Python内建支持的一种封装，我们通过把大段代码拆成函数，通过一层一层的函数调用，就可以把复杂任务分解成简单的任务，这种分解可以称之为面向过程的程序设计。函数就是面向过程的程序设计的基本单元。

而函数式编程（请注意多了一个“式”字）——Functional Programming，虽然也可以归结到面向过程的程序设计，但其思想更接近数学计算。

我们首先要搞明白计算机（Computer）和计算（Compute）的概念。

在计算机的层次上，CPU执行的是加减乘除的指令代码，以及各种条件判断和跳转指令，所以，汇编语言是最贴近计算机的语言。

而计算则指数学意义上的计算，越是抽象的计算，离计算机硬件越远。

对应到编程语言，就是越低级的语言，越贴近计算机，抽象程度低，执行效率高，比如C语言；越高级的语言，越贴近计算，抽象程度高，执行效率低，比如Lisp语言。

函数式编程就是一种抽象程度很高的编程范式，纯粹的函数式编程语言编写的函数没有变量，因此，任意一个函数，只要输入是确定的，输出就是确定的，这种纯函数我们称之为没有副作用。而允许使用变量的程序设计语言，由于函数内部的变量状态不确定，同样的输入，可能得到不同的输出，因此，这种函数是有副作用的。

函数式编程的一个特点就是，允许把函数本身作为参数传入另一个函数，还允许返回一个函数！

Python对函数式编程提供部分支持。由于Python允许使用变量，因此，Python不是纯函数式编程语言。

« 迭代器 高阶函数 »

Comments

Comments loaded. To post a comment, please Sign In

sumo @ 2025/11/5 02:59:50

简单说，函数式编程的核心思想是： 把代码当成 “数学计算”，用纯函数组合逻辑，尽量避免依赖外部状态，让代码更简洁、可预测。Python 虽然不是纯函数式语言，但支持这种思想，你可以根据需要选择用函数式风格写代码。

初八吃什么 @ 2025/10/15 05:32:50

挺住

森林 @ 2025/5/26 01:59:02

打卡

- 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
  - 7.1. 切片
  - 7.2. 迭代
  - 7.3. 列表生成式
  - 7.4. 生成器
  - 7.5. 迭代器
- 8. **函数式编程**
  - 8.1. 高阶函数
    - 8.1.1. map/reduce
    - 8.1.2. filter
    - 8.1.3. sorted
  - 8.2. 返回函数
  - 8.3. 匿名函数
  - 8.4. 装饰器
  - 8.5. 偏函数
- 9. 模块
  - 9.1. 使用模块
  - 9.2. 安装第三方模块
- 10. 面向对象编程
  - 10.1. 类和实例
  - 10.2. 访问限制
  - 10.3. 继承和多态
  - 10.4. 获取对象信息
  - 10.5. 实例属性和类属性
- 11. 面向对象高级编程
  - 11.1. 使用__slots__
  - 11.2. 使用@property
  - 11.3. 多重继承
  - 11.4. 定制类
  - 11.5. 使用枚举类
  - 11.6. 使用元类
- 12. 错误、调试和测试
  - 12.1. 错误处理
  - 12.2. 调试
  - 12.3. 单元测试
  - 12.4. 文档测试
- 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

痕迹 @ 2025/7/30 09:13:58

打卡

😜 @ 2025/6/10 02:08:17

day9

路 @ 2025/5/14 21:17:13

8. 函数式编程(Functional Programming) done

日向夏橘 @ 2025/5/5 00:28:14

打卡

寥寥星辰 @ 2025/4/16 07:38:03

day8

阿巴阿巴阿巴 @ 2025/4/11 03:10:24

打卡 day 4

0.0 @ 2025/3/30 10:11:26

day2

luseike @ 2025/3/9 21:05:32

ioser 转行自学python day3

方雨 @ 2025/2/11 22:27:01

这块好难啊，会影响我这代码民工吗。

开罗的猫 @ 2024/8/1 03:23:14

函数式编程的一个特点就是，允许把函数本身作为参数传入另一个函数，还允许返回一个函数！

大侦探福尔摩一 @ 2024/4/21 04:04:26

函数式编程是一种编程范式，目前理解来看主要目的是没有副作用，有点懵，继续学习

在下胡汉三 @ 2022/4/12 06:02:04

高阶函数与函数式编程

LA LA LAND @ 2023/10/31 03:42:30

牛的，讲解细致，演示到位

Beyond the stars @ 2024/3/16 08:33:28

Chat GPT：好的，我来尽量用更通俗的语言来解释函数式编程和面向过程编程的区别。

面向过程编程

想象你在做一顿饭，面向过程编程就像是你按照食谱的指示，一步一步地去做。每一步都是一个过程，比如：

- 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
  - 7.1. 切片
  - 7.2. 迭代
  - 7.3. 列表生成式
  - 7.4. 生成器
  - 7.5. 迭代器
- 8. 函数式编程
  - 8.1. 高阶函数
    - 8.1.1. map/reduce
    - 8.1.2. filter
    - 8.1.3. sorted
  - 8.2. 返回函数
  - 8.3. 匿名函数
  - 8.4. 装饰器
  - 8.5. 偏函数
- 9. 模块
  - 9.1. 使用模块
  - 9.2. 安装第三方模块
- 10. 面向对象编程
  - 10.1. 类和实例
  - 10.2. 访问限制
  - 10.3. 继承和多态
  - 10.4. 获取对象信息
  - 10.5. 实例属性和类属性
- 11. 面向对象高级编程
  - 11.1. 使用__slots__
  - 11.2. 使用@property
  - 11.3. 多重继承
  - 11.4. 定制类
  - 11.5. 使用枚举类
  - 11.6. 使用元类
- 12. 错误、调试和测试
  - 12.1. 错误处理
  - 12.2. 调试
  - 12.3. 单元测试
  - 12.4. 文档测试
- 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

1. 切菜：你按照食谱的指示去切各种蔬菜。
2. 煮水：然后把水煮沸。
3. 炒菜：最后按照一定的顺序把蔬菜放进锅里炒。

在这种编程方式中，你关心的是如何做：先做什么，然后做什么，每一步都具体指明做的动作。

函数式编程

Read More ▼

梦幻蔚蓝 @ 2023/10/31 23:06:13

根本停不下来

∑ @ 2023/9/16 16:11:22

如果我们有两个函数f，g满足

f:a -> b

g:b -> c

则有

g(f(x)) 即(g°f)(x)

那么我们就称

h=g°f是函数f和g的组合。

只要一门语言可以完成这件事，那么他就是支持函数式的，如果他能舒服的完成这件事，那么他就是一门优秀的函数式语言了。

°Destiny @ 2022/6/18 21:57:43

多个函数的嵌套调用，开闭原则

Bundle @ 2022/5/5 09:08:03

函数式编程中每个函数只实现一种功能，且纯净无其他变量（避免不必要的I/O操作）。返回值只受参数影响。通过这些函数的组合调用来实现计算任务。

举个例子：求（1+2）* 3 - 4

由数学计算规则知，应当先算加法，再算乘法，最后算减法。一共需要三种运算。

根据函数式编程，这三种运算应分别由三个函数来实现。最后组合调用。

三种运算的函数定义如下：

def add(x,y):

return x+y

def mul(x,y):

Read More ▼

空天阔海_1990 @ 2019/9/30 07:00:44

我的理解是下面这样的：

函数编程：相当于一种对数据的直接加工。假如数据是面粉，函数可以是做和面的机器。调用它就可以做完和面这个过程，输出--->和好的面。

函数式编程：相当于把这个过程更抽象了一层。编写函数式编程的过程，相当于调用各种低级机器（函数），组装制造出一条更复杂的设备，相当于一个制造工厂。这个制造工厂，可以输入函数1（和面的机器）、函数2（发酵的机器）、函数3（产生蒸汽的机器），组装好后就输出-->函数4（制造馒头的机器）？

- 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
  - 7.1. 切片
  - 7.2. 迭代
  - 7.3. 列表生成式
  - 7.4. 生成器
  - 7.5. 迭代器
- 8. **函数式编程**
  - 8.1. 高阶函数
    - 8.1.1. map/reduce
    - 8.1.2. filter
    - 8.1.3. sorted
  - 8.2. 返回函数
  - 8.3. 匿名函数
  - 8.4. 装饰器
  - 8.5. 偏函数
- 9. 模块
  - 9.1. 使用模块
  - 9.2. 安装第三方模块
- 10. 面向对象编程
  - 10.1. 类和实例
  - 10.2. 访问限制
  - 10.3. 继承和多态
  - 10.4. 获取对象信息
  - 10.5. 实例属性和类属性
- 11. 面向对象高级编程
  - 11.1. 使用__slots__
  - 11.2. 使用@property
  - 11.3. 多重继承
  - 11.4. 定制类
  - 11.5. 使用枚举类
  - 11.6. 使用元类
- 12. 错误、调试和测试
  - 12.1. 错误处理
  - 12.2. 调试
  - 12.3. 单元测试
  - 12.4. 文档测试
- 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

使用的时候就可以输入 面粉+函数4，直接输出馒头?

longtometosee @ 2022/2/6 22:20:47

**多个函数的嵌套调用，开闭原则 **

thirty22 @ 2021/12/6 08:04:15

函数式编程就是一种抽象程度很高的编程范式，纯粹的函数式编程语言编写的函数没有变量，因此，任意一个函数，只要输入是确定的，输出就是确定的，这种纯函数我们称之为没有副作用。而允许使用变量的程序设计语言，由于函数内部的变量状态不确定，同样的输入，可能得到不同的输出，因此，这种函数是有副作用的。

函数式编程的一个特点就是，允许把函数本身作为参数传入另一个函数，还允许返回一个函数！

Python对函数式编程提供部分支持。由于Python允许使用变量，因此，Python不是纯函数式编程语言。