Scheme和Python中迭代操作的实现差异

温永潜

2024/11/21

摘要

迭代操作（程序中的循环与递归操作）是解决问题的重要手段之一。因此，绝大部分编程语言都支持迭代操作。但是函数式编程语言和过程式编程语言在迭代操作和实现上存在区别。由于，Scheme是函数式编程语言的代表，Python是过程式编程的代表。因此本文将以Scheme，Python为例，分析函数式编程与过程式编程在循环与递归在实现层面的差异，具体包括编程思想，实现方式，适用场景。

t

1 引言

迭代操作，既包括程序中的循环操作，如python中的for循环，又包括程序中的递归操作，如scheme中常见的阶乘运算的实现。绝大部分的不同编程范式的编程语言都支持迭代操作，但也在实现层面存在差异。因此，理解不同编程范式中的迭代操作的实现差异对于深入理解编程思想和优化解决问题手段具有重要意义。本文将以典型的函数式编程语言Scheme和过程式编程语言Python为例，介绍函数式编程和过程式编程中迭代操作的实现差异，包括编程思想，实现方式，适用场景。编程思想包括编程哲学与状态管理，编程哲学是确定操作为何存在和一种对操作的高维度的概括，状态管理是对于迭代操作中非常重要的一个概念——状态的处理。实现方式则是具体代码的书写形式，本文将以斐波那契数列为例去讲解在具体语言下，循环操作与递归操作具体是如何实现的。适用场景则是对每个操作应该在哪里使用才能发挥最大作用的总结。

2 编程思想

编程哲学 Scheme是一种函数式编程语言。函数式编程语言的迭代操作主要通过递归。递归操作通过自身调用自身的，通过一种从上到下的抽象手段实现程序，强调一件事“是什么”，使代码更直观。 Python是一种过程式编程语言。过程式编程语言的迭代操作主要通过递归和循环。过程式编程语言的递归操作在编程哲学层面与函数式编程语言的递归操作区别并不大，因此我们主要在这里讨论过程式编程的循环操作。循环操作通过显式控制状态（控制循环中变量的更新和条件的限制）来实现程序，强调“如何做”一件事，使代码可读性高，逻辑性强，易于跟踪状态。

状态管理 在过程式编程中，状态是循环操作的核心。状态在每次循环操作中都会被改变，每次循环操作也只依赖于目前的状态。因此循环操作会产生副作用（如改变变量值），也使程序的实现有迹可循。

函数式编程的递归操作不改变状态，而是通过创建一个新的函数调用来模拟新的状态的产生，而非修改原有状态。因此，程序的状态由函数的输入值的返回值来控制，同时，递归操作不产生副作用。当然对状态的控制与管理也使得副作用的产生，这会使得函数称为非纯函数，也会使得细节隐藏在状态管理的代码里头，而状态管理的代码影响外部环境的作用难以被侦察。

而放弃状态管理可以使得函数不存在难以侦察的副作用。

3 实现方式

下面将以实现阶乘运算的简单求值为例，介绍Scheme与Python中的迭代操作的差异

Scheme

```
# 示例代码
(define (fib n)
 (cond ((= n 1) 1)
   (else (* n (fib (- n 1))))))
```

Python

```
def fib(n):
 result = 1
 for i in range(1, n+1, 1):
   result *= i
 return result
```

4 适用场景

Scheme Scheme的递归操作更适合在对代码性能要求不高，对代码准确性要求高，要求验证的情形下使用。Scheme的递归操作推荐在代码性能要求不高的情形下使用，是由于函数递归调用会造成一些使用过程式编程的循环操作便可以避免的空间开销；递归操作推荐在对代码准确性要求高的情形下使用，是由于函数式编程天然具有很明显的输入-¿输出的标准路径，代码书写者很容易能发现自己的问题；推荐在要求验证的情形下使用是因为函数式编程的求值路径更相似于数学公式中的函数求值，更倾向于数学的本质，因此对函数式编程的递归操作进行数学的验证相对于迭代操作会方便一些。

Python Python的循环操作更适合在代码构造难度较大，要求浅显易懂的情形下使用。 Python的循环操作推荐在代码构造难度较大时使用，是因为循环操作相对函数式编程来说更符合人们的直观思考，人们更倾向于想要知道一个“怎么做”的具体算法，人们也可以通过明晰第一步是什么，第二步是什么的方式来构造迭代操作，而函数式编程的递归更倾向于从一个“是什么”的角度解决问题，因此，用递归解决问题需要具有较强的宏观观察能力。因此，当问题难度较大时，Python的循环操作更符合人类直观思考；循环操作推荐在要求代码浅显易懂的情形下使用是由于接触过程式编程的循环操作的人占绝大多数，且相比于理解递归，理解函数式编程，理解循环的难度是更低的，理解循环的普及度也是最广的。因此，如果想让代码能让绝大多数人都能看懂，更推荐Python的循环操作。

需要额外强调的是，相比Python会经常影响状态的循环操作，Scheme的递归操作更容易把模块化的编程哲学体现出来，这会使得代码更加可维护，

可迭代，易于管理与理解。

5 结论

总之，Scheme与Python在迭代操作的实现在编程思想，实现方式，适用场景三个方面存在差异：在编程思想上，Scheme的迭代操作更强调问题是什么，而Python的迭代操作（尤指循环操作）更强调问题具体应该怎么做；在实现方式上，Scheme与Python本身语法不同，且编程思想不同，实现方式必然存在着区别；在适用场景上，Scheme的迭代操作更适合在对代码性能要求不高，对代码准确性要求高，要求验证的情形下使用。而Python的循环操作更适合在代码构造难度较大，要求浅显易懂的情形下使用。