第六周问答作业

朱士杭 231300027

2024 年 4 月 3 日

1 python 执行环境如何保证递归执行的正确性

1.1 调用栈管理

python 递归的时候每一次遇到执行自身函数/方法的时候，都会创建一个新的栈帧 StackFrame（具体的栈帧分离放到下一点去讲）来表示当前的函数调用，并将新的数据存储都压入到一个叫作调用栈（Call Stack）数据结构当中去，跟 C++ 中普通的栈一样都是后进先出的。

通过对调用栈的管理使得每一层递归的层次结构就变的很清晰，而且函数调用关系、变量的值保存追踪也会很容易。每一次创建的新的栈帧都会压入到调用栈当中去，每一次这一层递归结束之后旧栈帧会从调用栈当中弹出并且销毁掉

1.2 栈帧隔离

每个新栈帧和就栈帧之间是相互隔离的，对一个栈帧的操作不会影响其他的栈帧，因为调用栈本身就是一个只能处理最上面一层栈帧的数据结构，会有一个指针指向当前的栈帧并且固定每一个栈帧内存大小，只要指针指向最上面的那一层栈帧就不会对其他的栈帧产生影响。

1.3 递归深度限制

每一次递归都会创建新的栈帧都需要申请新的内存空间，而内存空间有限，栈帧是有限制的，python 解释器会对栈帧的数量和大小进行限制，因此递归的深度也是需要有限制的，以防止栈溢出和资源耗尽。可以使用 sys 库里面的 sys.setrecursionlimit(最大递归层数) 来限制递归深度防止栈溢出。

2 解释器如何执行一个调用函数

当在主函数（这里其实引用 C++ 中的一个概念其实在 python 中也是相通的）中调用其他函数的时候，首先必须需要记录下现在已经执行到哪里了，这个时候调用栈作用就很重要了，会将要调用的函数压入调用栈中，并在调用栈里面记录下有关函数调用的信息（比如函数当前栈帧、函数返回地址、函数当前局部变量、函数代码信息等）
（另外再说一句，解释器由于是逐行解释，因此函数定义必须放到调用它之前去，否则会报错）

而对于多层函数逐层调用的过程，每一次调用都会向栈里压如一次有关函数调用的信息，这其中最重要的就是 f_lineno 函数返回地址和 f_locals 函数当前局部变量这两个信息，一个记录当前函数调用发生在第几行，一个记录当前环境下的变量信息，前者是为了在函数执行结束后找到回来的位置，后者是为了回来以后还原上下文环境。

3 基于 frame 计算模型描述 python 函数调用过程

函数调用的时候，在当前函数会有一个 frame 记录当前函数的信息，当在当前函数中调用其他函数时，会创建一个新的 frame 并将其压入到调用栈中，而在这个新的 frame 当中会存有新函数信息，比如 local-variable 局部变量

当调用的函数执行完了之后，会将该函数的 frame 弹出栈销毁掉，而调用函数的同时会在栈中记录下上一层函数执行到了哪一行，这样的话解释器就可以通过这个信息重新回到上一层函数执行，直到调用栈的最底层的帧也被弹出销毁掉结束

4 Pure-Function 与 Non-Pure-Function

4.1 Pure-Function

4.1.1 什么是 Pure-Function

Pure-Function 是指那些不会对外界产生干扰的函数，说白了就是不会产生副作用，在 pure-function 中处理的变量全都是 local-variable 局部变量，对其他函数与主函数都不会产生影响，输入相同，输出也相同，与外界完全隔离与其余状态无关

4.1.2 Pure-Function 的应用场景

没有副作用，利于重构，不会影响函数外部元素的状态，可以放心的对纯函数进行重构，而不必担心会对其他模块产生任何影响。

不受外部状态影响，利于并行处理, 纯函数完全独立于外部状态，不受所有与共享可变状态有关问题的影响，在跨多个 CPU 以及整个分布式计算集群进行并行处理时是极佳选择。

更容易理解和测试，纯函数行为是确定的，不会受到外部环境的影响。

4.2 Non-Pure-Function

4.2.1 Non-Pure-Function

Non-Pure-Function 是指会产生副作用的函数，由于函数对变量处理的时候优先级为 local-frame，parent-frame，global-frame 依次排列去寻找该变量在哪里，当自身未定义相关局部变量的时候会对全局变量进行处理修改，会产生一定的副作用.

有时候我们确实需要这样的副作用，用于修改全局变量、修改文件、打印输出等，但是目前来说为了让程序更加稳定可控仍然还是推崇纯函数占比更多。

4.2.2 Pure-Function 的应用场景

与外界交互、调用 I/O 输入输出的接口、从函数范围之外检索值、磁盘检索、抛出异常等情况