

Caculator

解析 : 转换语言 (converting a language)

编译器的解析步骤:

step I: take program (基本上都是长字符串)

step II: convert them into a sequence of internal forms

例如在 Scheme 中 ($+ 1 2$) 的内部表示形式为 Pair('+', Pair(1, Pair(2, nil)))

step I 将按行读取的字符串存入 lines 中

step II 将 lines 中的字符串 tokenize (将其分解为 atoms 的列表)

step III 将令牌化的 lines 做成 buffer (一次分发一个令牌)

step IV 从 buffer 读取 tokens, 转化为 Scheme 表达式.



计算 : 计算表达式的值

在 Scheme 中就是对 Scheme 列表求值.

假定计算表达式的函数为 eval, 那么它应是一个递归形式.

对所有子表达式调用 eval 得到操作数, 再调用 apply 计算结果.

def eval(exp):

判断 exp 是否为 atoms, 若是则直接返回, 代码略.

if isinstance(exp, Pair):

args = exp.rest.map(eval)

return apply(exp.first, args)

而 apply 则接受 procedure 和 *args 两个参数, 返回对 *args 调用 procedure 的结果.

Read-Eval-Print Loop (REP Loop)

从用户的程序到呈现的结果实际上就是 读入 - 解析 - 计算 - 输出 的循环

在其中若有任何一步出现错误都将引发错误 (raise Error)