The Interpreter + Calling Functions

STk> (sqrt 16)
4
STk> (+ 3 (sqrt 16))
7
STk> (+ 3 4 5 6)
18

Why not 3 + 4
Here we were calling the function +

Scheme was designed by people

STk> (+)
0
STk> (*)
1
STk> (- 9 5 2)
2

Someone designed these to make sense. EVERYTHING should make sense!
You might disagree, but you should still understand the rationale

Parentheses Matter

STk> 9
9
STk> (+ #[closure arglist=arg 196d20])

We asked scheme what this is, it said: 9
We asked scheme what this is, it said: a procedure

BIG IDEA: Procedures are a “thing”

STk> * #[closure arglist=arg 1970d0]

You can not include extra parentheses!

Clickers

• Clickers are required (register online)
• Helps me see what concepts are challenging for the class
• Helps you see what concepts are challenging for you
• Provides
  – Explanations from peers
  – Experience explaining tough concepts
TRY IT

\[ \text{STk>} (+ 7 (* 3 4) (* (/ 10 5) (- 3 10))) \]

How many parentheses do we need at the end?

a) 0  
b) 1  
c) 2  
d) 3  
e) 4  

If you finish early calculate the answer!

Quoting stuff

\[ \text{STk>} ' + \]
\[ \text{STk>} 'hello hello} \]
\[ \text{STk>} ' (+ 2 3) \]
\[ (+ 2 3) \]
\[ \text{STk>} ' (good morning)) \]
\[ \text{(good morning)} \]

These are words

These are sentences

Notice the lack of quote on the result

Functions for words/sentences

\[ \text{STk>} (\text{first } 274) \]
\[ 2 \]
\[ \text{STk>} (\text{butfirst } 274) \]
\[ 74 \]
\[ \text{STk>} (\text{first } '\text{hello}) \]
\[ \text{h} \]
\[ \text{STk>} (\text{first } '(\text{hello})) \]
\[ \text{hello} \]

This is a word

This is a sentence

Sort of weird that they work on:

numbers  
words  
sentences

REVIEW: Two Types of ’s so far

\[ (\text{first } ')(\text{hello})) \]

Call a function

Indicate a sentence

REVIEW: Two Types of ”’s so far

‘hi ’(hello)

This is a word

This is a sentence

http://csillustrated.berkeley.edu
Functions for words/sentences

STk> (first (butfirst 'hello))
  e

STk> (se (first 23) (last 45))
  24

Work from the inside out. 10% knowledge 90% care

TRY IT

STk> (first '(hi))
  hi
STk> (butfirst '(hi))
  a) i
  b) 'i
  c) <= this is blank
  d) ()
  e) '()
Undefined Variables

STk> 'pi
pi
*** Error:
  undefined variable: pi
Current eval stack:
---------------------
  0 pi

The word pi
Without a quote scheme
things this is a variable
This shows
what calls
preceded the
error

Defining Variables

STk> (define pi 3.14)
pi
STk> pi
3.14
STk>'pi
pi
STk>(+ pi 7)
0.14

Make a variable
Now no error!
The word
pi is still
different
Can be
used in
expressions

REVIEW: Way to define variables

(define variable value)
Keyword
& special
form
An
expression
Shouldn't
be an
expression

Bad Function

STk> (define pi 3.14)
pi
STk> (pi 5)
*** Error:
  eval bad function in: (pi 5)
Current eval stack:
---------------------
  0 (pi 5)

Make a variable
This assumes
that pi was a
function

Function Definition

STk> (define pi 3.14)
pi
STk> (define (square x)
  (* x x))
square
STk> (square 5)
25

Make a symbol (aka
variable)
Define a
function

Function Name
Formal parameters
Keyword
Body
Actual argument
value
Defining plural

(define (plural wd) (word wd 's))

Implicitly returns last thing

Predicates

• Predicates are procedures that return #t or #f
  – by convention, their names end with a "?"

odd? (odd? 3) ➞ #t
even? (even? 3) ➞ #f
vowel? (vowel? 'a) ➞ #t
  (vowel? (first 'fred)) ➞ #f
sentence? (sentence? 'fred) ➞ #f

Defining Plural (try 2)

(define (plural wd)
  (if (equal? (last wd) 'y)
      (word (bl wd) 'ies)
      (word wd 's))
)

IF & COND Statements

(if <predicate>
  <true case>
  <false case>)
(cond
  (predicate1 return_expression1)
  (predicate2 return_expression2)
  (else return_expression3))

Try It

Write a better plural function using cond

  • Works for “fox/foxes” and “wolf/wolves”

    (cond
     (predicate1 return_expression1)
     (predicate2 return_expression2)
     (else return_expression3))

Factorial (Recursion Review)

• 10! = 10*9! (recursive case) x! = x*(x-1)!
• 1! = 1 (base case)
• 0! = 1 (base case)

(define (factorial x)
  (if (< x 2)
      1
      (* x (factorial (- x 1))))
)
What is $(\text{factorial } 4)$?

$(\text{define (factorial x)})$

$(\text{if (< x 2)})$

$\ 1$

$(\text{(* x (factorial (- x 1)))))$

All Recursive Procedures Need

1. Base Case (s)
   - Where the problem is simple enough to be solved directly
2. Recursive Cases (s)
   - Divide the Problem (Make the problem Smaller!)
     - into one or more smaller problems
   - Invoke the function
     - Have it call itself recursively on each smaller part
   - Combine the solutions
     - Combine each subpart into a solution for the whole

Try It!

- Write $\text{count}$ that takes in a sentence and counts the words in the sentence.

Count the number of words in a sentence

$(\text{define (count sent)})$

$(\text{if (empty? sent)})$ ;no more?

$0$ ;base case: return 0

$(+ 1 \text{(count (bf sent))})$ ;recurse on the rest of sent

$(\text{count '(a b c)})$

Try It!

- Write $\text{copies}$ that takes in a word and a variable $n$ and repeats the word $n$ times in a sentence.

Copies

$(\text{define (copies n wd)})$

$(\text{if (< n 1)})$

$'()$

$(\text{sentence wd (copies (- n 1) wd))))$