Lispereter Documentation

Release v0.7.0

Muhammad Arslan

Jan 12, 2018

Contents

A personal project to create a Lisp interpreter with Python.

Any suggestions/reviews are welcome.

Contents:

CHAPTER 1

Overview

An attempt to improve my python skills.

Suggested at http://pythonpracticeprojects.com/.

CHAPTER 2

Parser

The parser functions and their brief description.

lispereter.parser.**atom**(*token*)

Numbers become numbers; every other token is a symbol.

Parameters **token** : str

The token to be turned to a LISP atom.

Returns **lisp_atom** : Atom

A LISP atom.

Examples

```
>>> atom("11")
11
```

```
>>> atom(11)
11
```

```
>>> atom("11.0")
11.0
```

```
>>> atom("begin")
'begin'
```

lispereter.parser.**parse**(*program*)

Read a Scheme expression from a string.

Parameters **program** : str

The expression to parse.

Returns **expression** : Exp

A Scheme expression (list/atom).

Examples

```
>>> parse("(begin (define r 10) (* pi (* r r)))")
['begin', ['define', 'r', 10], ['*', 'pi', ['*', 'r', 'r']]]
```

lispereter.parser.**read_from_tokens**(*tokens*)

Read an expression from a sequence of tokens.

Parameters **tokens** : list

A list of tokens.

Returns **parsable** : Exp

An expression to be parsed.

Examples

```
>>> read_from_tokens(['(', 'begin', '(', 'define', 'r', '10', ')', '(', '*', 'pi',
↪ '(', '*', 'r', 'r', ')', ')', ')'])
['begin', ['define', 'r', 10], ['*', 'pi', ['*', 'r', 'r']]]
```

lispereter.parser.**tokenize**(*chars*)

Convert a string of characters into a list of tokens.

Parameters **chars** : str

The input string to tokenize.

Returns **token_list** : List

A list of tokens.

Examples

```
>>> tokenize("(begin (define r 10) (* pi (* r r)))")
['(', 'begin', '(', 'define', 'r', '10', ')', '(', '*', 'pi', '(', '*', 'r', 'r',
↪')', ')', ')']
```

CHAPTER 3

Eval

The *evaluate* function, and *Env* and *Procedure* class.

class lispereter.environ.**Env**(*params=()*, *args=()*, *outer=None*)

An environment: a dict of {'var': val} paris, with an outer Env.

Parameters **params** : set, optional

Set of variables.

args : set, optional

The values associated with variables in *params*.

outer : Env, optional

Outer environment

find(*var*)

Find the innermost environment where the variable occurs.

Parameters **var** : str

The variable to find.

class lispereter.environ.**Procedure**(*params*, *body*, *env*)

A user-defined Scheme procedure.

Parameters **params** : set

A set of parameters for the procedure.

body : str

The actual expression of the procedure to execute.

env : Env

The current environment of the procedure.

lispereter.environ.**evaluate**(*x*, *env=global_env*)

Evaluate a given expression *x* using *env*.

Parameters **x** : Exp

The expression to evaluate.

env : dict, optional

The environment holding the variables etc.

Returns **val** : (Exp, Atom)

Some answer.

Examples

```
>>> evaluate(parse("(begin (define r 10) (* pi (* r r)))"))
314.1592653589793
```

```
>>> evaluate(parse('(if (< 2 3) (print 2) (print 3))'))
2
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index