EECS 368
Programming Language Paradigms

Dr. Andy Gill

Department of Electrical Engineering & Computer Science
University of Kansas

October 6, 2015

(Scheme)

We can define new atoms on the command line.

```
> (define atom "Smallish Thing")
> (define turkey "Fair Game")
> atom
"Smallish Thing"
> turkey
"Fair Game"
```

Better is putting the definitions in a file, and loading them each time.

We store the following definitions in a file.

```
(define atom "Smallish Thing")
(define turkey "Fair Game")
(define *abc$ "Silly String")
```

```
> *abc$
"Silly String"
> (atom)
procedure application: expected procedure,
            given: "Smallish Thing" (no arguments)
```

The problem is that the command line **expects** a function call, not data.

Quoting

In Java, there is a difference between `foo` and ``` ``foo'' ```.

```
String foo = "foo";
```

One is a variable, the other data.

In Scheme, we quote our data by prefixing a single quote.

- `turkey` is the name of an expression.
- `'turkey` is an expression.

S-Expressions

```
SExp ::= atom
       | string
       | ' SExp
       | ( SExp* )
```

- Atoms are names that do not begin with `'('`, `')'`, `'"'`, `'‘'`
- Strings are like Java Strings, starting with `'"'`, and ending with `'"'`

Scheme

We know know most of Scheme.

- The syntax is *really* simple
- The same syntax is used for data as well as programs

```
(define atom "Smallish Thing")
(define turkey "Fair Game")
```

```
> (atom)
procedure application: expected procedure,
            given: "Smallish Thing" (no arguments)
> '(atom)
(list 'atom)
```

The problem was that the command line **expects** a function call, not data, so we used ' to denote the list as data.

.. no defintions ..

```
> 'atom
atom
> '(atom)
(list 'atom)
> '(atom turkey or)
(list 'atom 'turkey 'or)
> '(atom turkey) or
 or: bad syntax in: or
> '((atom turkey) or)
(list (list 'atom 'turkey) 'or)
```

```
(((how) are) ((you) (doing so)) far)
```

How many S-expressions are in this list?

- Is ( ) a list?

- Is ( ) a list? Yes
- It is an empty list!

- Is ( ) a list? Yes
- It is an empty list!

- Is ( ) an atom?

- Is () a list? Yes
- It is an empty list!

- Is () an atom? No
- It is a list, not an atom

- Is (() () () ()) a list?

- Is `(() () () ())` a list? Yes
- It is a list of empty lists