Function representation

- Functions in lisp are represented as lists with specific formats
- `(function` *`fname`*`)` returns the list representing the function
- Three different formats, based on how the function was created
- Functions can be created with defun, lambda, or labels
- Each format starts with a special symbol identifying the format
- Two formats contain a trio of environment lists, which provide information such as what variables/values were visible at the point of the function creation, the point of call, etc

The defun format

- Functions created with defun start with the symbol ‘lambda-block, followed by the function name, parameter list, and body, e.g.

```
(defun g (x y) (* x y))
```

- For g above, `(function g)` would return the following list

```
(lambda-block g (x y) (* x y))
```

- We could thus write code to look at the innards of g:

```
(format t “param list of g is: (nth 2 (function g)))
```

The lambda format

- Functions created with lambda start with the symbol ‘lambda-closure, followed by the parameter list, followed by the body

```
(lambda (x y) (* x y))
```

- If run at the global scope, this returns the list

```
(lambda-closure () () () (x y) (* x y))
```

- The three empty lists at the front are the environment lists
- If we ran lambda from inside another block then they would contain lists of what was defined/visible in that block

Caution about environment lists

- It is entirely possible the environment lists will include cyclic references, so if you try to display or print them you get infinite output
- Remember our trick for safe printing of cyclic lists:

```
(let ((*print-circle* t))
      (format t “~A~%” myCyclicList))
```

Labels format

- Local functions can also be defined in a labels block (much like local variables in a let block), e.g.

```
(labels ((h (x y) (* x y)))
    ; can call h in the block, we'll just return form
     (function h))
; this would return the following list:
(lambda-block-closure (**) (**) (**) h (x y) (* x y))
```

- The three (**) are the environment lists, what they contain will depend on the enclosing blocks

Parsing the form of a function

- If we've got the list representing a function, the symbol at the front tells us which format it is, and thus if we should expect to find the three environment lists and/or a function name before the parameter list
- (everything after the parameter list is the function body)