

Principles of Programming Languages Handout

See EoPL Section 2.3.

One way to think of environments is as finite functions that map names to their values.

```
(define create-empty-ff
  (lambda ()
    (lambda (sym)
      (error "Symbol not found"))))
```

```
(define extend-ff
  (lambda (sym val ff)
    (lambda (sym2)
      (if (eq? sym2 sym)
          val
          (ff sym2)))))
```

```
(define apply-ff
  (lambda (ff sym)
    (ff sym)))
```

Examples:

```
(define e
  (extend-ff 'x 2
            (extend-ff 'y 3
                      (create-empty-ff))))
```

```
(apply-ff e 'x)
2
```

```
(apply-ff e 'y)
3
```

```
(apply-ff e 'z)
*** ERROR -- Symbol not found
```

```
(apply-ff (extend-ff 'x 5 e) 'x)
5
```

Exercise Does this representation of finite functions/environments work with dynamic binding (that is, if the `lambda` in the above code behaves as it does in Emacs Lisp)? Why?

It is more realistic to represent environments as data structures.

```
(define-datatype environment environment?  
  (empty-env-record)  
  (extended-env-record  
    (syms (list-of symbol?))  
    (vals (list-of scheme-value?))  
    (env environment?)))
```

```
(define scheme-value? (lambda (v) #t))
```

We define a procedure `empty-env` that returns the empty environment, which contains no bindings:

```
(define empty-env  
  (lambda ()  
    (empty-env-record)))
```

The procedure `extend-env` extends a given environment with some more bindings:

```
(define extend-env  
  (lambda (syms vals env)  
    (extended-env-record syms vals env)))
```

```
(define apply-env  
  (lambda (env sym)  
    (cases environment env  
      (empty-env-record ()  
        (eopl:error 'apply-env "No binding for ~s" sym))  
      (extended-env-record (syms vals env)  
        (let ((pos (list-find-position sym syms)))  
          (if (number? pos)  
              (list-ref vals pos)  
              (apply-env env sym))))))))
```

```
(define list-find-position  
  (lambda (sym los)  
    (list-index (lambda (sym1) (eqv? sym1 sym)) los)))
```

Exercise Does this representation of environments work with dynamic binding? Why?