Closures

Rushikesh Joshi
IIT Bombay

What is a closure

A lambda expression that remembers the current state of its environment for later application

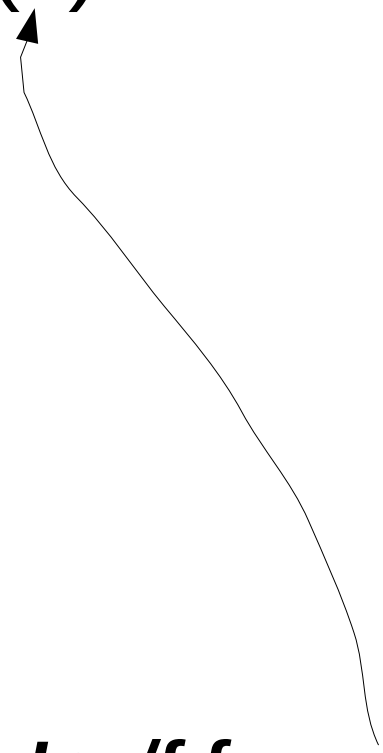
Such an expression (a closure) can be returned from within another lambda expression which contains the environment to be remembered inside the closure

```
def data(x):
  return (lambda f:
                f(x)
          )

def val(x):
   return (x)

c = data(3)

v = c (val)
print v
```



In /f.f x, x is a free variable

This free variable x is bound (closed). It is bound to the value in the enclosing environment

A use of closure

It hides state inside a lambda abstraction
---> **Data Abstraction**

Later, this value hidden inside the lambda function body can be extracted.

This is demonstrated in the example
on the previous slide

Pairs revisited – pairs as closures

```
def pair (x,y):
   return ( lambda f:
                f (x,y) )

def first (x):
    return x (true)

def second (x):
    return x (false)
```

```
def cons(h,t):
    return (pair(false,pair(h, t)))

def head (l):
    return first (second(l))

def tail(l):
    return second (second(l))

def isnil (l):
    return first (l)

def nillist():
    return pair (true,true)

l1 = cons (1,cons(2,cons(3,nillist())))
l2 = cons (1,cons(2,cons(3,cons(4,cons(5,nillist())))))

print (head(l1))
print (head(tail(tail(l2))))
```

Revisiting Lists.. Lists use pairs

Revisiting lists: Is a List a nillist?

```
print isnil(l1)('yes','no')
print isnil(l2)('yes','no')

print isnil(tail(l1))('yes','no')
print isnil(tail(tail(l1)))('yes','no')
```

```
def data(x,y):
  return (lambda f:
            f(x,y)
          )

def swap (x,y):
   return (lambda g:
               g(y,x) )

def first(x,y):
        return (x)

def second(x,y):
        return (y)
```

Swapping Values enclosed in a closure – return a new closure!

```
def data(x,y):
  return (lambda f:
          f(x,y)
        )

def swap (x,y):
   return (lambda g:
             g(y,x) )

def first(x,y):
        return (x)

def second(x,y):
        return (y)
```

```
c = data(3,4)

d = c (swap)

v = c (first)
print v
v = c (second)
print v

v = d (first)
print v
v = d (second)
print v
```

Swapping Values