61A Lecture 12

Monday, September 24

Dictionaries

```
{'Dem': 0}
```

Restrictions on Dictionaries

Dictionaries are **unordered** collections of key-value pairs.

Dictionary keys do have two restrictions:

- A key of a dictionary **cannot be** an object of **a mutable built-in** type.
- Two **keys cannot be equal.** There can be at most one value for a given key.

This first restriction is tied to Python's underlying implementation of dictionaries.

The second restriction is an intentional consequence of the dictionary abstraction.

Sharing and Identity

```
demo = []
```

What Would Python Print?

The print function returns None. It also displays its arguments (separated by spaces) when it is called.

If you're not sure what will happen, draw environment diagrams

```
from operator import add, mul
def square(x):
    return mul(x, x)

def delay(arg):
    print('delayed')
    def g():
        return arg
    return g
```

A function that takes any argument and returns a function that returns that arg

Names in nested def statements can refer to their enclosing scope

$\underbrace{\texttt{print(add(3, 4)}}_{7}\texttt{, }\underbrace{\texttt{print(5)}}_{\texttt{None}}\texttt{)}$

```
delay(delay)()(6)()
```

```
print(delay(print)())
```

What Would Python Print?

The print function returns None. It also displays its arguments (separated by spaces) when it is called.

If you're not sure what will happen, draw environment diagrams

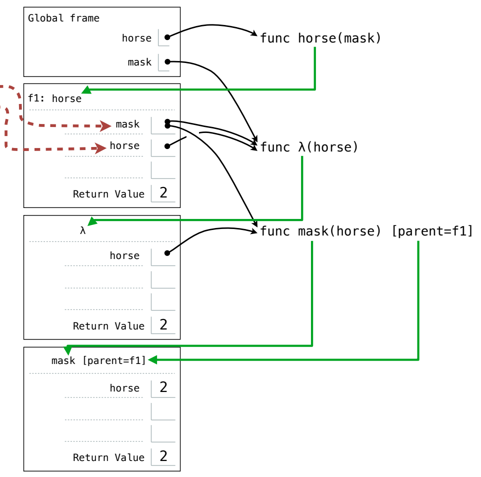
```
from operator import add, mul
def square(x):
    return mul(x, x)

def pirate(arggg):
    print('matey')
    def plunder(arggg):
        return arggg
    return plunder
```

A function that always returns the identity function

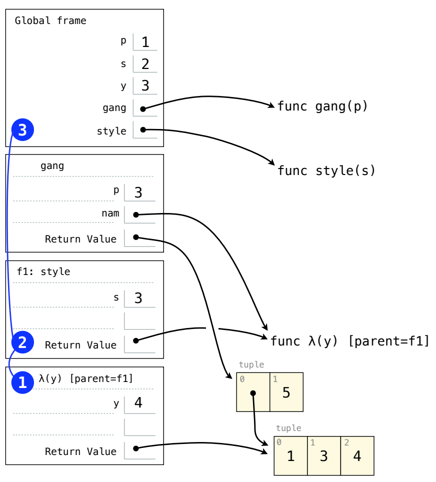
$\texttt{add(}\underbrace{\underbrace{\texttt{pirate(3)(square)}}_{\textit{func square(x)}}\texttt{(4)}}_{\textit{16}}\texttt{, 1)}$

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

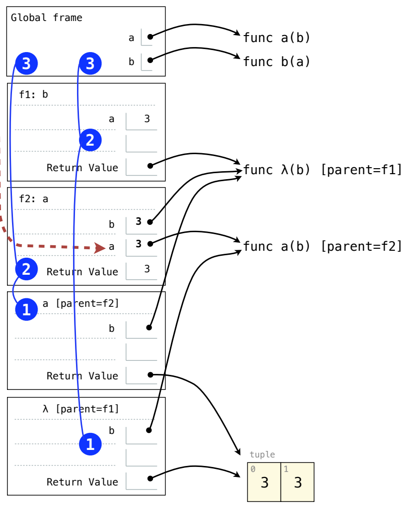
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



```
p, s, y = 1, 2, 3

def gang(p):
    nam = style(p)
    return (nam(4), 5)

def style(s):
    return lambda y: (p, s, y)

gang(3)
```



```
def a(b):
    def a(b):
        return b(a)
    a, b = a(b)
    return a

def b(a):
    return lambda b: (a, a)

a(b(3))
```



Inverse Functions

If g is the inverse of invertible f, then x = f(g(x))

Key equation: g(x) is the value y, such that **f(y) = x**

Rearrange to use Newton's method: **f(y) − x = 0**

For variable y
and constant x,
f(y) − x

```
def invert(f):
    def g(x):
        return find_root(lambda y: f(y) - x)
    return g
```