Function Examples

Announcements

Hog Contest Rules

- Up to two people submit one entry;
  Max of one entry per person
- Your score is the number of entries against which you win more than 50.00001% of the time
- Strategies are time-limited
- All strategies must be deterministic, pure functions of the players' scores
- Winning entries will receive a paltry amount of extra credit
- The real prize: honor and glory
- See website for detailed rules

Fall 2011 Winners

Keegan Mann
Yan Duan & Ziming Li
Brian Prike & Zhenghao Qian
Parker Schuh & Robert Chatham

Fall 2012 Winners

Chenyang Yuan
Joseph Hui

Fall 2013 Winners

Paul Bramsen
Sam Kumar & Kangsik Lee
Kevin Chen

Fall 2014 Winners

Alan Tong & Elaine Zhao
Zhenyang Zhang
Adam Robert Villaflor & Joany Gao
Zhen Qin & Dian Chen
Zizheng Tai & Yihe Li

cs61a.org/proj/hog_contest

Hog Contest Winners

Spring 2015 Winners

Sinho Chewi & Alexander Nguyen Tran
Zhaoxi Li
Stella Tao and Yao Ge

Fall 2015 Winners

Micah Carroll & Vasilis Oikonomou
Matthew Wu
Anthony Yeung and Alexander Dai

Spring 2016 Winners

Michael McDonald and Tianrui Chen
Andrei Kassiantchouk
Benjamin Krieges

Fall 2016 Winners

Cindy Jin and Sunjoon Lee
Anny Patino and Christian Vasquez
Asana Choudhury and Jenna Wen
Michelle Lee and Nicholas Chew

Your name could be here FOREVER!

Fall 2017 Winners

Alex Yu and Tanmay Khattar
James Li
Justin Yokota

Spring 2018 Winners

Eric James Michaud
Ziyu Dong
Xuhui Zhou

Fall 2018 Winners

Rahul Arya
Jonathan Bodine
Sumer Kohli and Neelesh Ramachandran

Fall 2019 Winners

Currying

Function Currying

```
def make_adder(n):
    return lambda k: n + k
```

```
>>> make_adder(2)(3)
5
>>> add(2, 3)
5
```

There's a general relationship between these functions

(Demo)

Curry: Transform a multi-argument function into a single-argument, higher-order function

Decorators

Function Decorators

(Demo)

Function decorator

```
@trace1
def triple(x):
    return 3 * x
```

Decorated function

is identical to

Why not just use this?

```
def triple(x):
    return 3 * x
triple = trace1(triple)
```

Review

What Would Python Display?

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

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

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

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

Names in nested def statements can refer to their enclosing scope

This expression	Evaluates to	Interactive Output
5	5	5
print(5)	None	5
print($\underbrace{\texttt{print(5)}}_{\texttt{None}}$)	None	5 None
delay(delay)()(6)()	6	delayed delayed 6
print(delay(print)()(4))	None	delayed 4 None

What Would Python Print?

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

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

A function that always returns the identity function

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

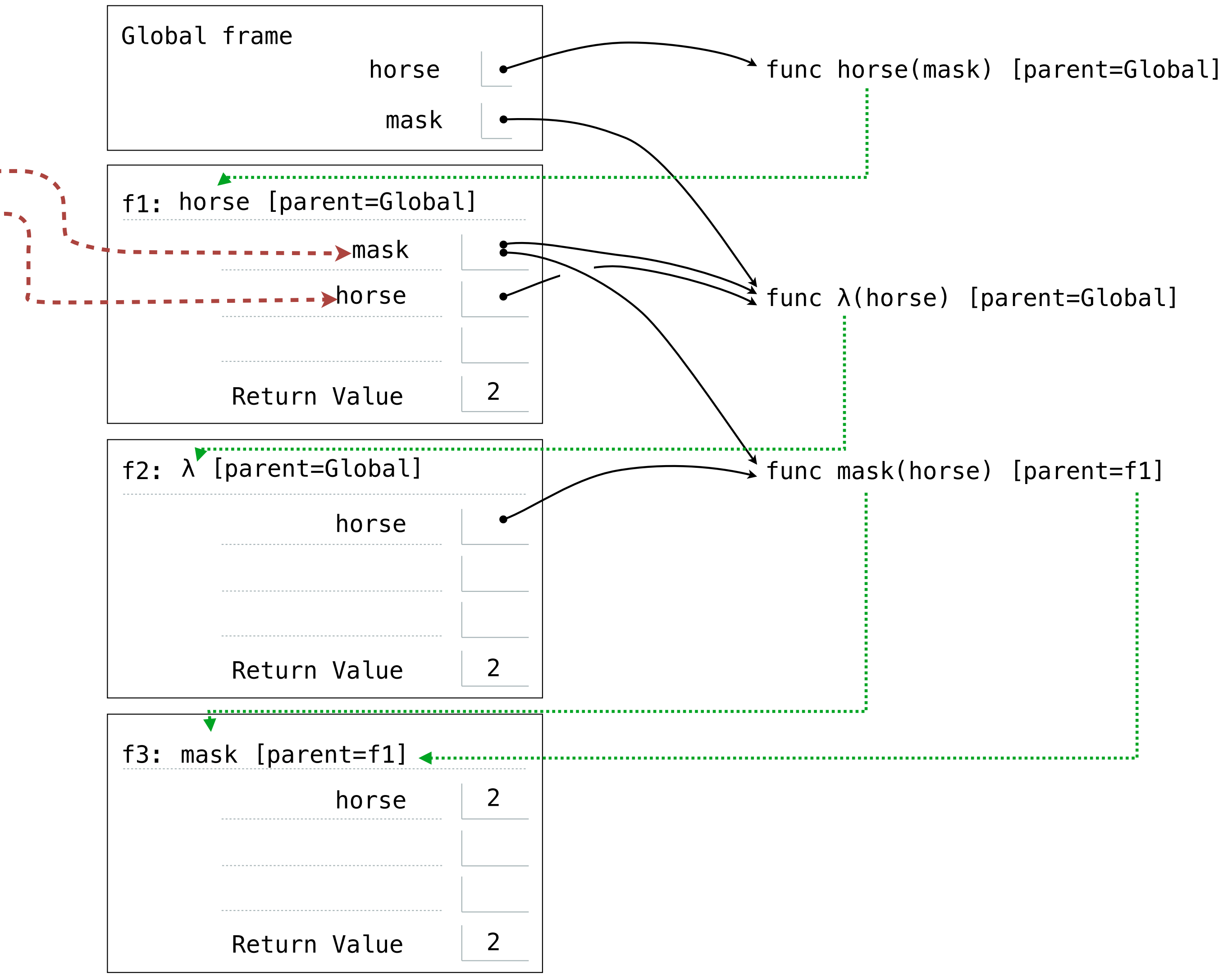
This expression	Evaluates to	Interactive Output
add(pirate(3)(square)(4), 1) *func square(x)* *16*	17	Matey 17
pirate(pirate(pirate))(5)(7) *Identity function* *5*	Error	Matey Matey Error

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)
```



Implementing Functions

Implementing a Function

```
def remove(n, digit):
    """Retu  n al   digits of non-negative N
            re            IT, for some
      non-nega            IT less than 10.

    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept, digits = 0, 0

    while ________n > 0________:

        n, last = n // 10, n % 10

        if ______last != digit______:

            kept = ~~10*~~kept + last*10**digits

            digits = ___digits + 1___

    return ______kept______
```

231

4

231

```
    1
 + 30
+ 200
-----
  231
```

Read the description

Verify the examples & pick a simple one

Read the template

Implement without the template, then change your implementation to match the template.
OR
If the template is helpful, use it.

Annotate names with values from your chosen example

Write code to compute the result

Did you really return the right thing?

Check your solution with the other examples

Implementing a Function

```
def remove(n, digit):
    """Retu n all  igits of non-negative N
      231  re   3   IT, for some
        n  ga        IT less than 10.

    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept, digits = 0, 0

    while ___________n > 0___________:

        n, last = n // 10, n % 10

        if _____last != digit________:

            kept = ____kept/10 +    last____

  21        digits = ____digits + 1______

    return __round(kept * 10 ** (digits-1))__
```

Read the description

Verify the examples & pick a simple one

Read the template

Implement without the template, then change your implementation to match the template.
OR
If the template is helpful, use it.

Annotate names with values from your chosen example

Write code to compute the result

Did you really return the right thing?

Check your solution with the other examples