# Function Examples

# Announcements

# Hog Contest Rules

cs61a.org/proj/hog_contest

## Hog Contest Rules

- Up to two people submit one entry;
  Max of one entry per person

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

cs61a.org/proj/hog_contest

## 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

## 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

# 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

# Function Currying

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

# Function Currying

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

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

# 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

# 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)

# 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 Decorators

(Demo)

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

# Function Decorators

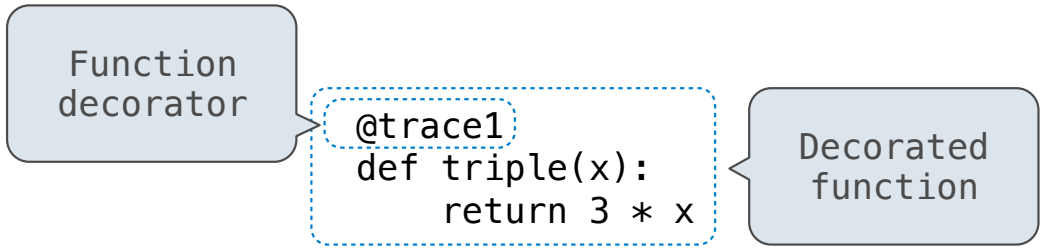
(Demo)

Function decorator

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

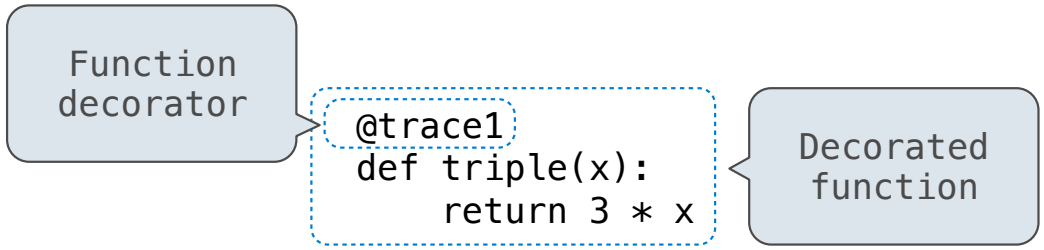
# Function Decorators

(Demo)



# Function Decorators

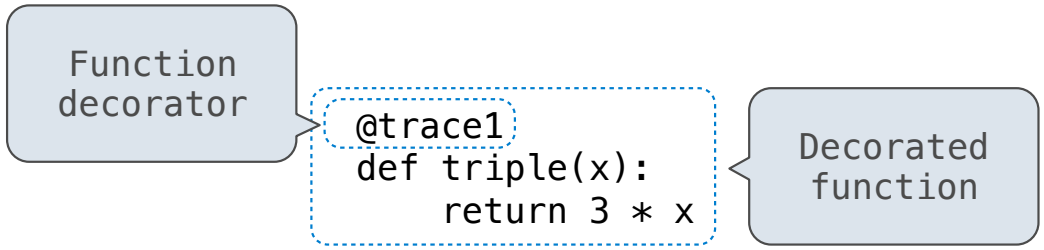
(Demo)




*is identical to*

# Function Decorators

(Demo)




*is identical to*

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

# 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?

## What Would Python Display?

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

## 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)
```

## 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)
```

| This expression | Evaluates to | Interactive Output |
| --- | --- | --- |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| `5` | 5 | 5 |
| `print(5)` | | |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | 5 |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | 5 |
| print(print(5)) | | |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | 5 |
| print(print(5)) | | |

None

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | 5 |
| print(print(5)) <br> None | | 5 <br> None |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | 5 |
| print(print(5)) <br> None | None | 5 <br> None |

# 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)
```

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

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | 5 |
| print(print(5)) <br> None | None | 5<br>None |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| 5 | 5 | 5 |
| print(5) | None | 5 |
| print(print(5)) <br> None | None | 5 <br> None |

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

Names in nested def statements can refer to their enclosing scope

# 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(print(5)) <br> None | None | 5 <br> None |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | | |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | | |

# 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(print(5))<br>None | None | 5<br>None |
| delay(delay)()(6)() | | |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | | |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | | |

# 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(print(5)) | None | 5<br>None |
| None | | |
| delay(delay)()(6)() | | delayed |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | | delayed <br> delayed |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | | delayed <br> delayed <br> 6 |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | 6 | delayed <br> delayed <br> 6 |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | 6 | delayed <br> delayed <br> 6 |
| print(delay(print)()(4)) | | |

# 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(print(5)) <br> None | None | 5 <br> None |
| delay(delay)()(6)() | 6 | delayed <br> delayed <br> 6 |
| print(delay(print)()(4)) | | delayed |

# 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(print(5)) <br> None | None | 5<br>None |
| delay(delay)()(6)() | 6 | delayed<br>delayed<br>6 |
| print(delay(print)()(4)) | | delayed<br>4 |

# 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(print(5)) <br> None | None | 5<br>None |
| delay(delay)()(6)() | 6 | delayed<br>delayed<br>6 |
| print(delay(print)()(4)) | | delayed<br>4<br>None |

# 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(print(5)) <br> None | None | 5<br>None |
| delay(delay)()(6)() | 6 | delayed<br>delayed<br>6 |
| print(delay(print)()(4)) | None | delayed<br>4<br>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)
```

| This expression | Evaluates to | Interactive Output |
| --- | --- | --- |

# 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)
```

| This expression | Evaluates to | Interactive Output |
| --- | --- | --- |

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

# 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)
```

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

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| `add(pirate(3)(square)(4), 1)` | | |

# 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)
```

| This expression | Evaluates to | Interactive Output |
|---|---|---|
| `add(pirate(3)(square)(4), 1)` | | |

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

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

# 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)` | | |

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

# 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)` | | |

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

# 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)` | | Matey |

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

# 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) | | Matey |

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

# 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) | | Matey |

*func square(x)*

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

# 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)* | Matey |

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

# 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) | | Matey |
| *func square(x)* | | |
| *16* | | |

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

# 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)` | | Matey<br>17 |
| *func square(x)* | | |
| *16* | | |

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

# 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) | 17 | Matey<br>17 |

*func square(x)*

*16*

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

# 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)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7) | | |

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

# 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)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7) | | |

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

# 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) | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7) | | |

*func square(x)*

*16*

*Identity function*

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

# 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)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7)<br>*Identity function* | | Matey |

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

# 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)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7)<br>*Identity function* | | Matey<br>Matey |

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

# 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)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7)<br>*Identity function* | | Matey<br>Matey |

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

# 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)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7)<br>*Identity function*<br>*5* | | Matey<br>Matey |

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

# 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)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7)<br>*Identity function*<br>*5* | | Matey<br>Matey<br>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.

# 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)
```

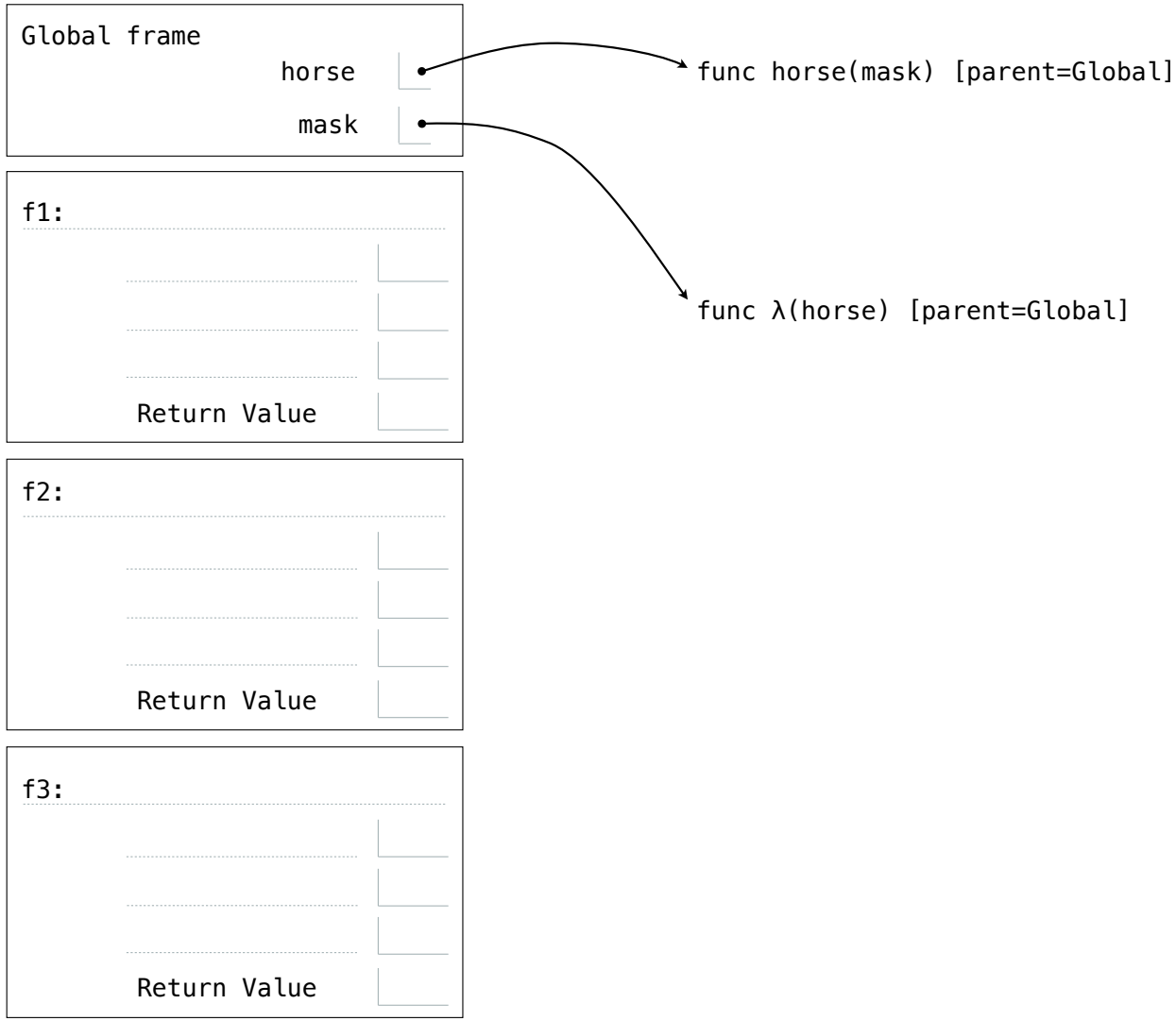
A function that always returns the identity function

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

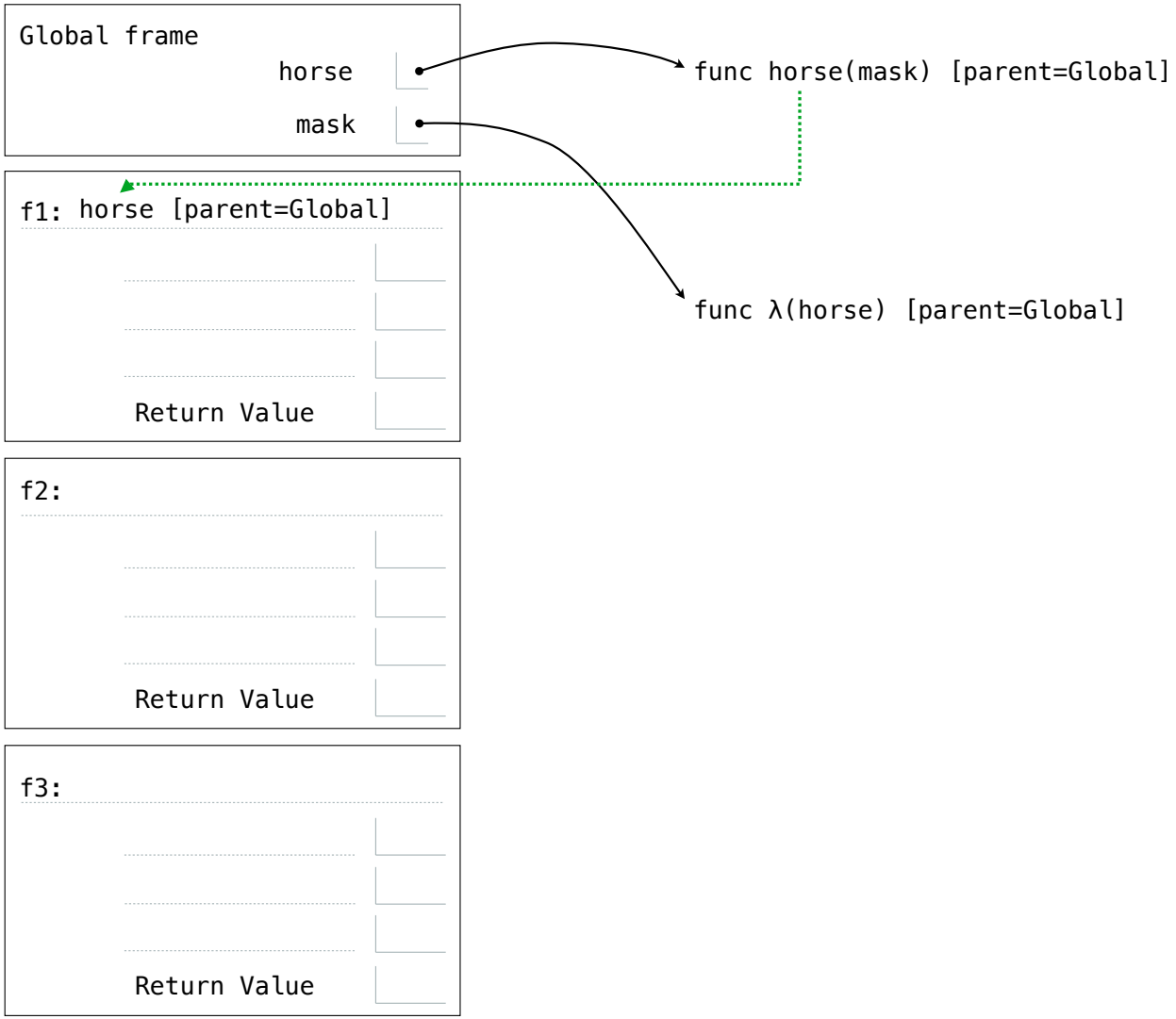
| This expression | Evaluates to | Interactive Output |
|---|---|---|
| add(pirate(3)(square)(4), 1)<br>*func square(x)*<br>*16* | 17 | Matey<br>17 |
| pirate(pirate(pirate))(5)(7)<br>*Identity function*<br>*5* | Error | Matey<br>Matey<br>Error |

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

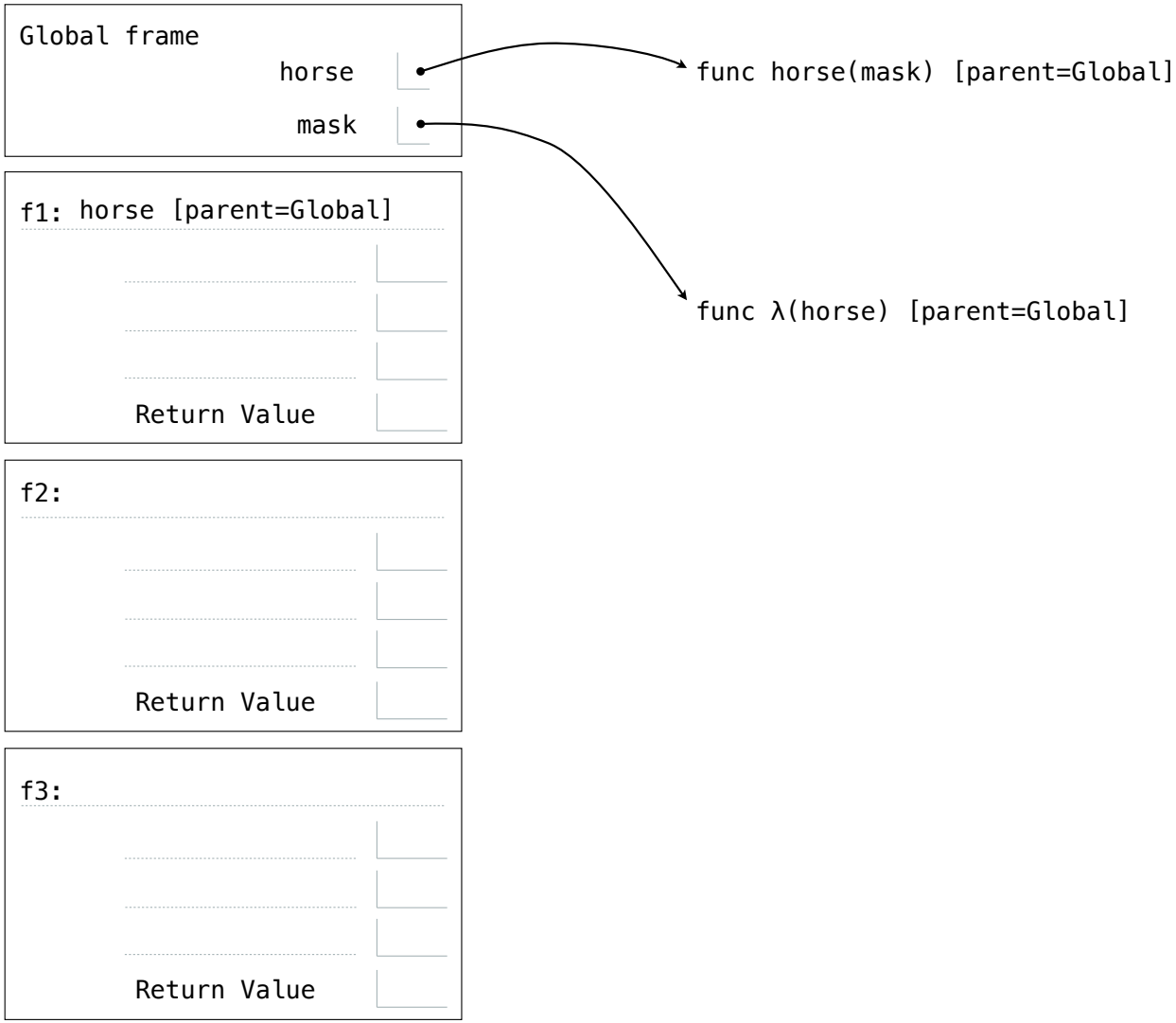
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



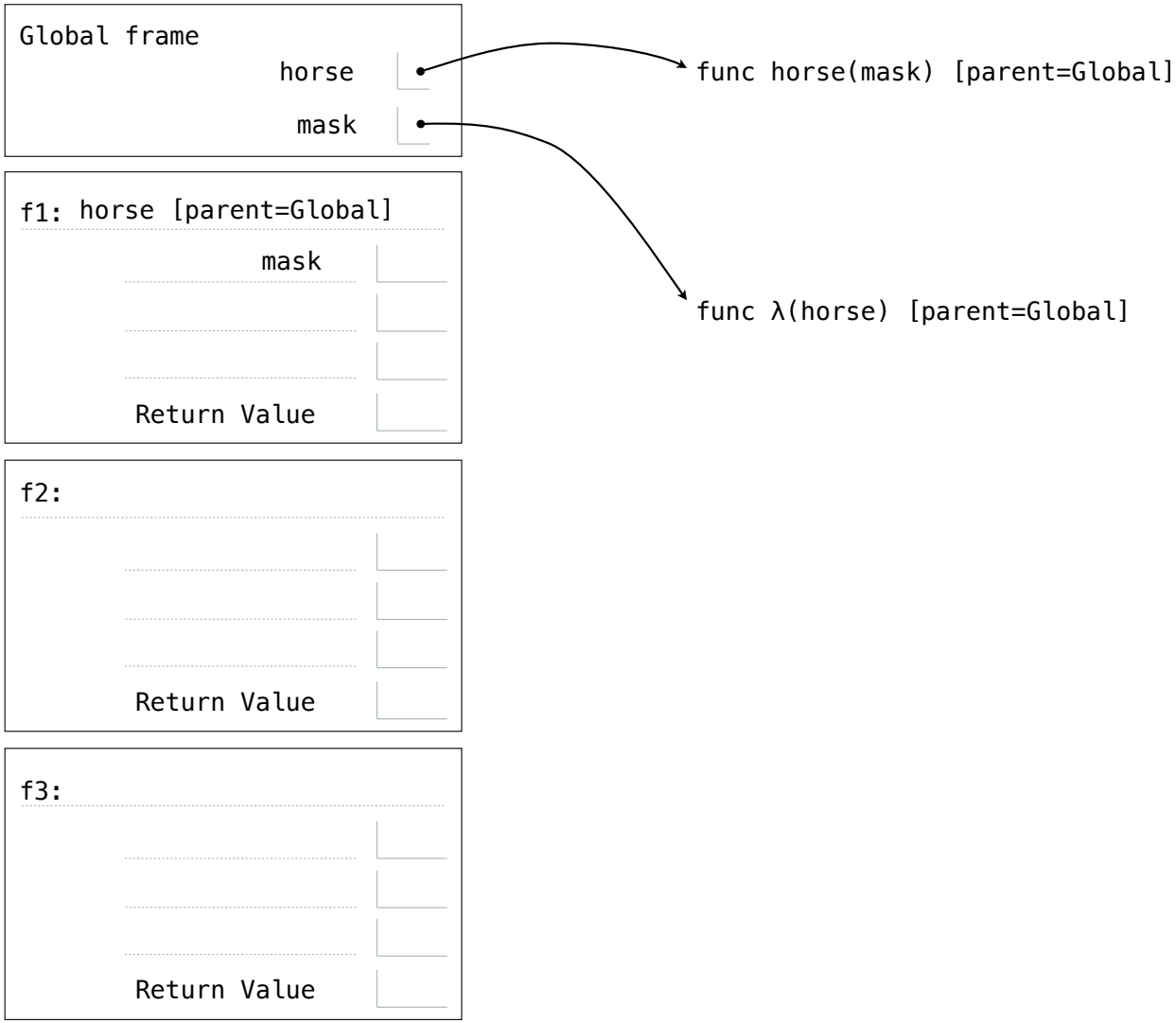
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



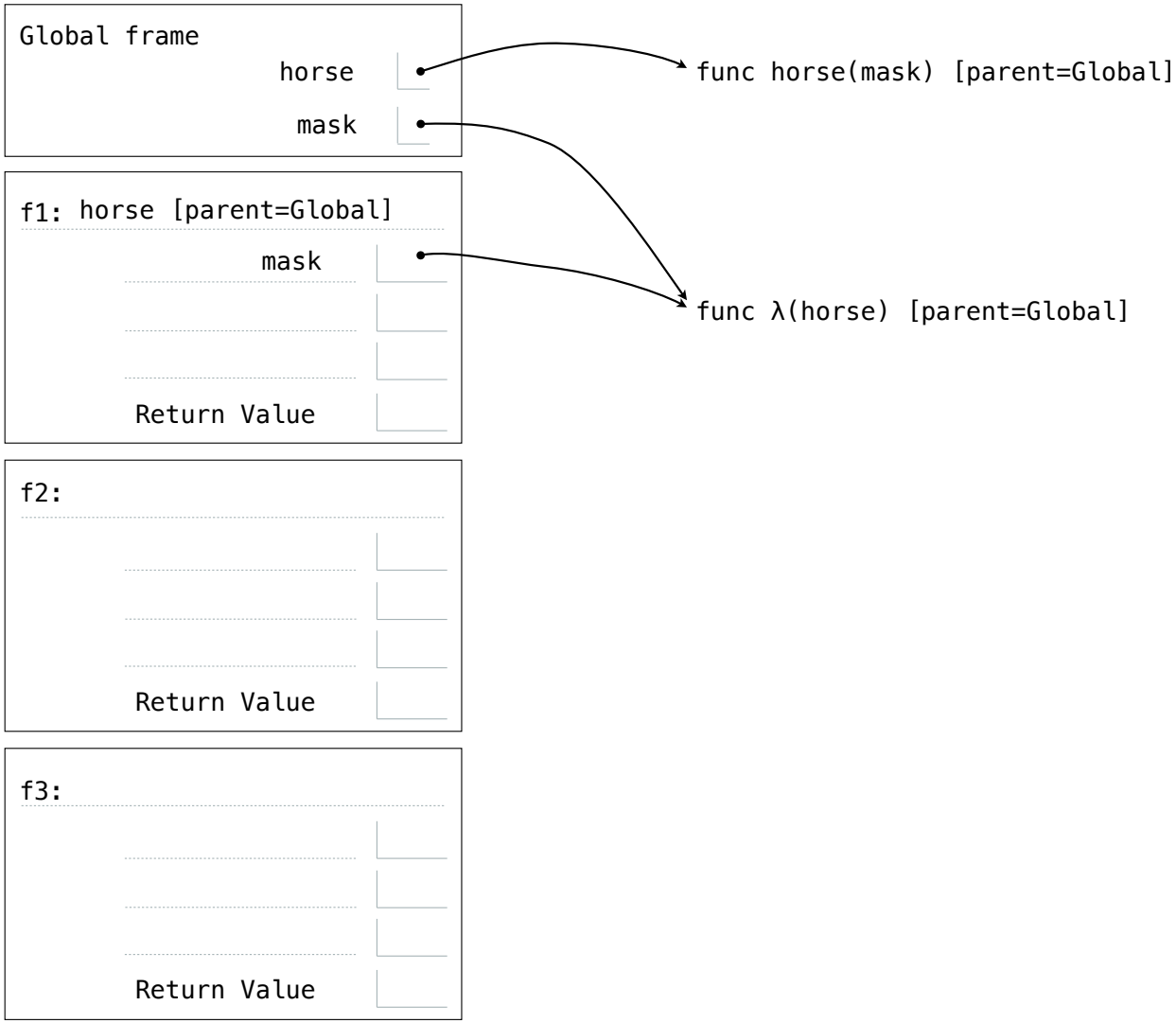
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



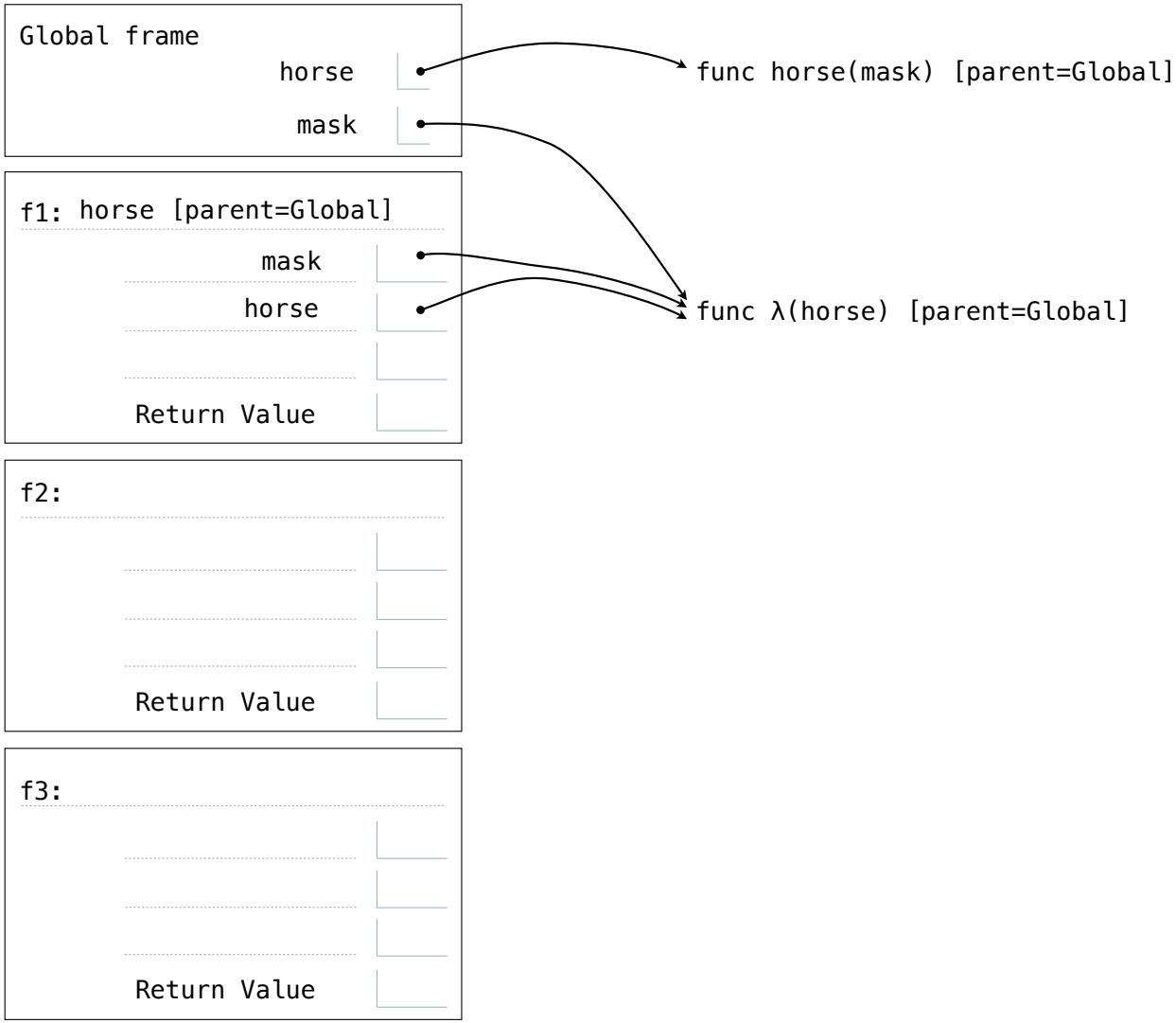
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



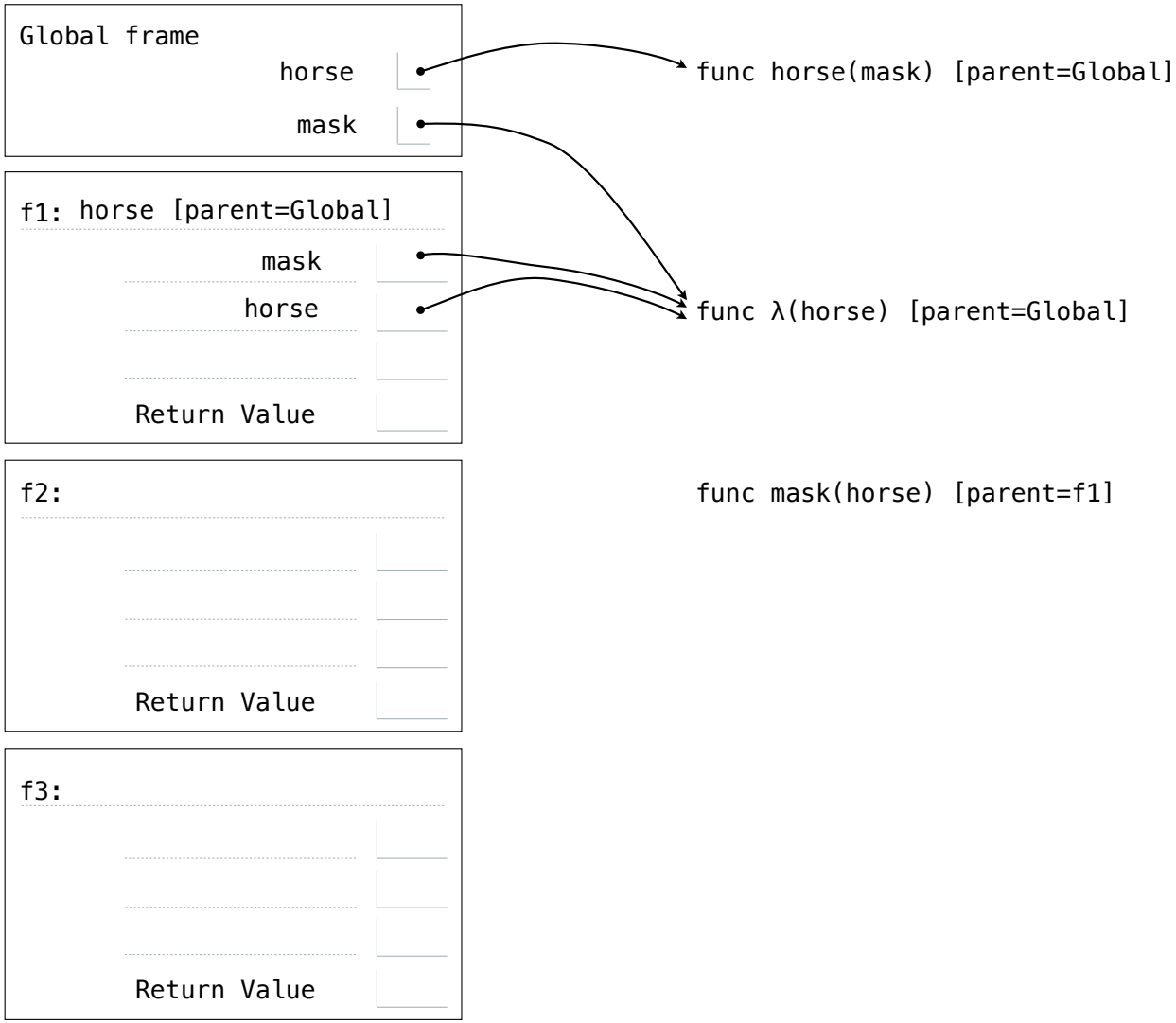
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



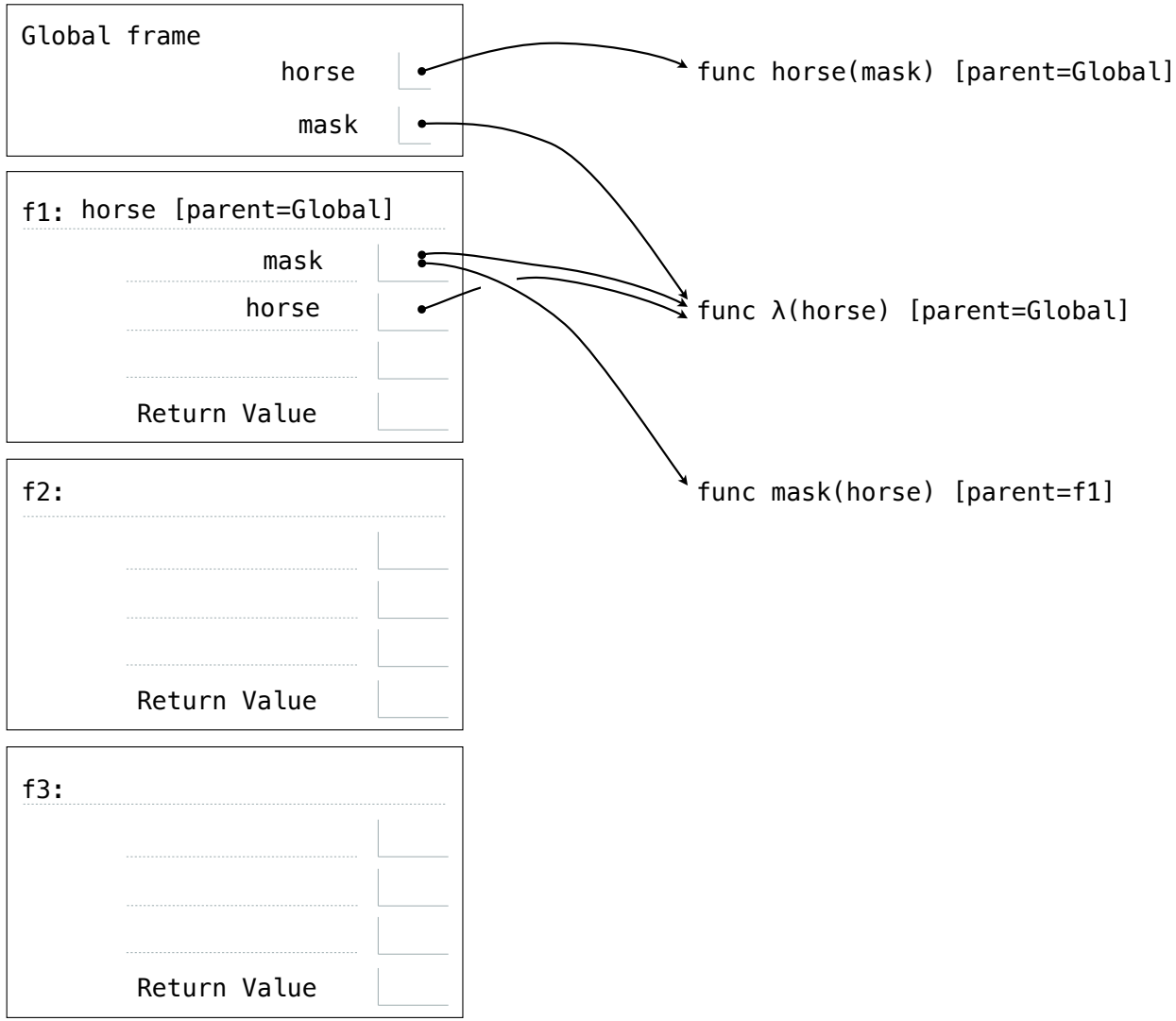
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



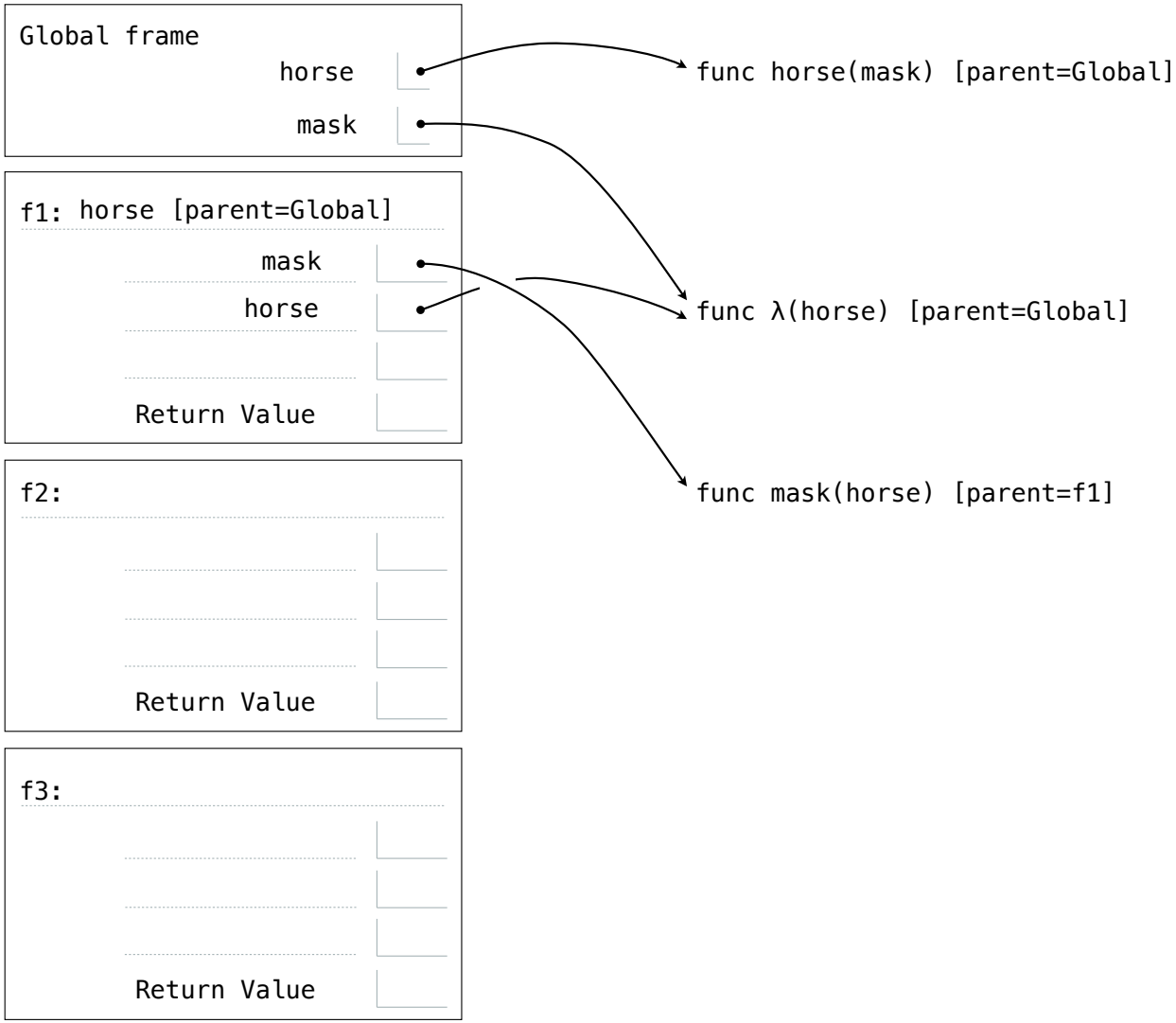
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



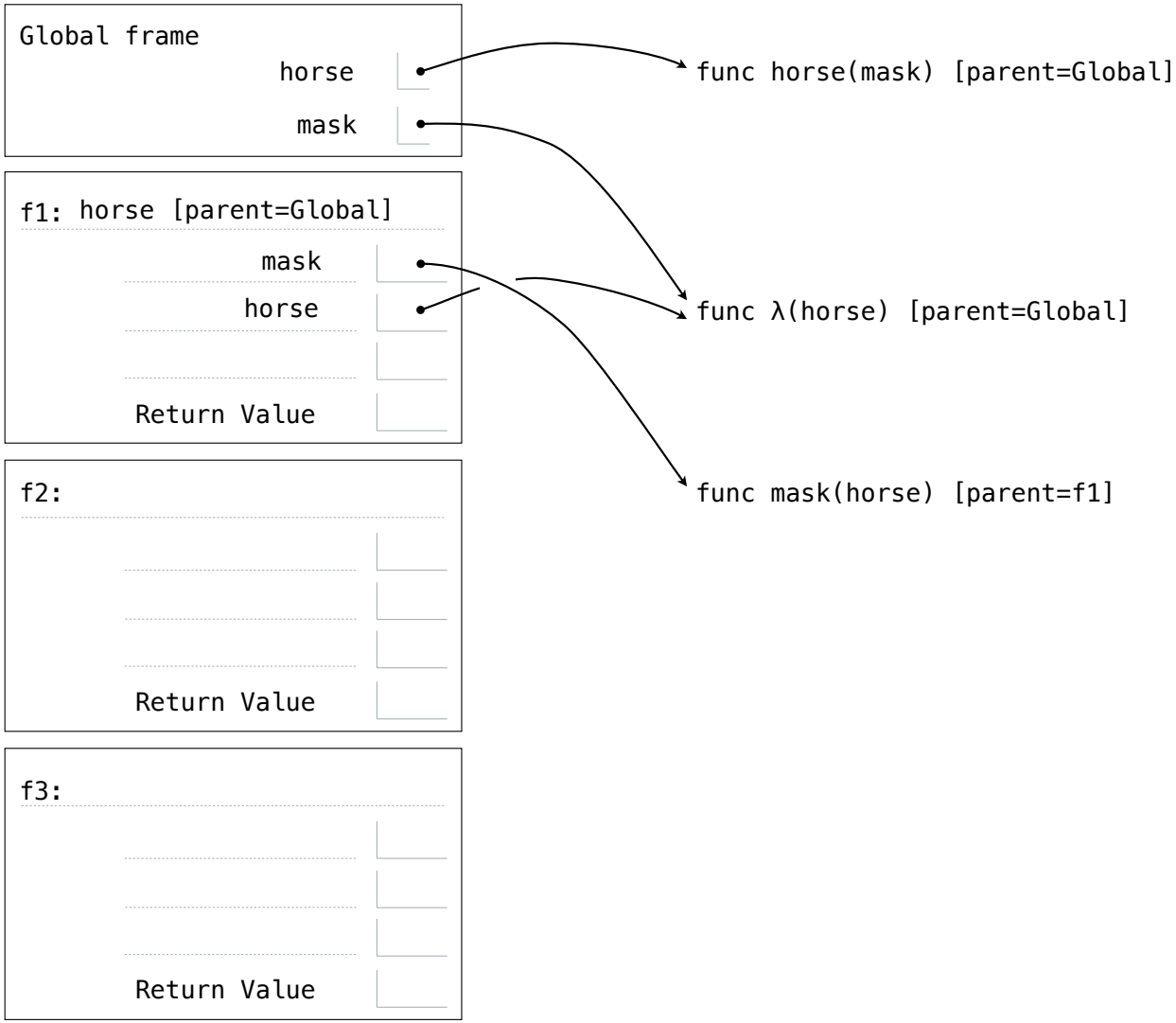
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



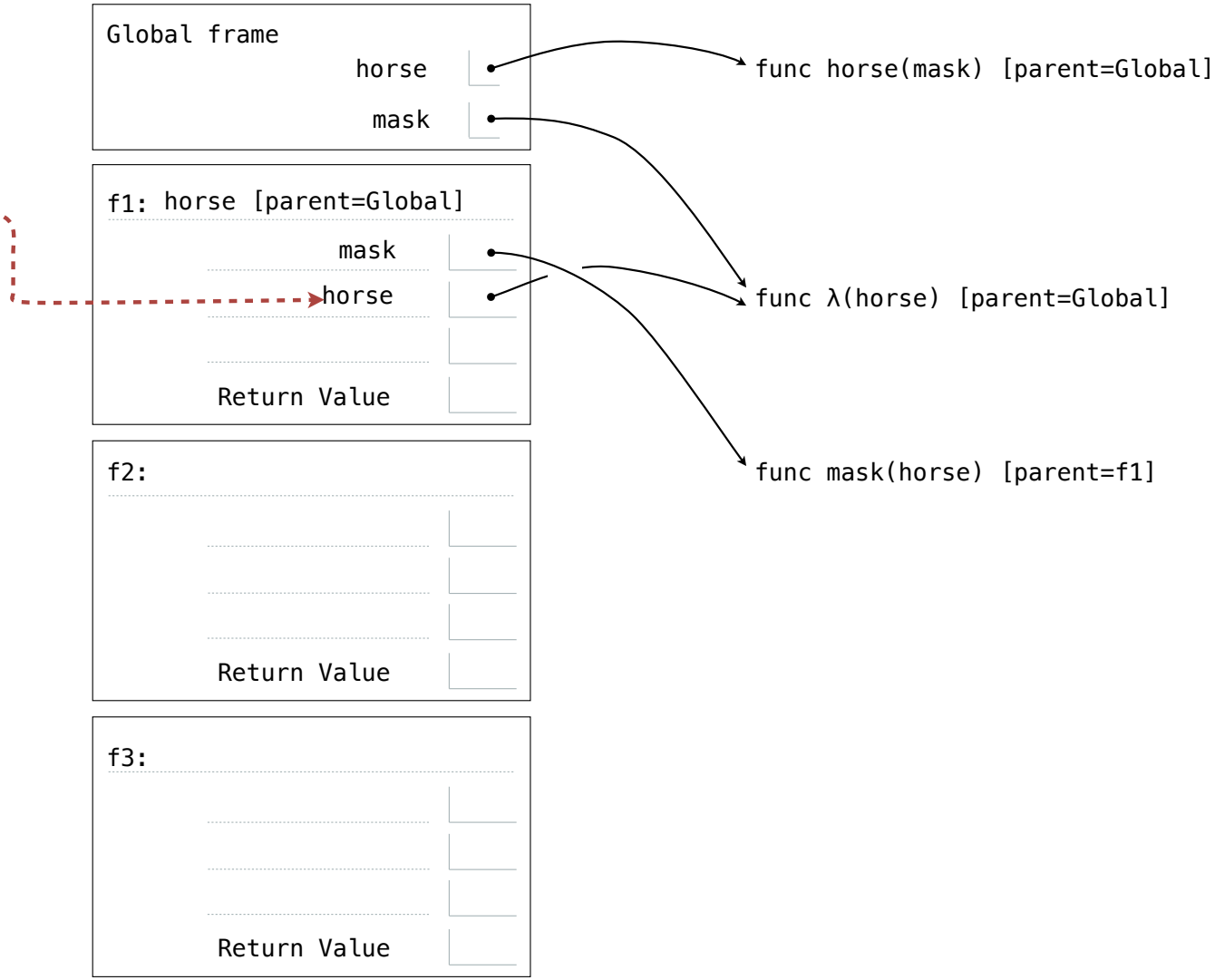
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



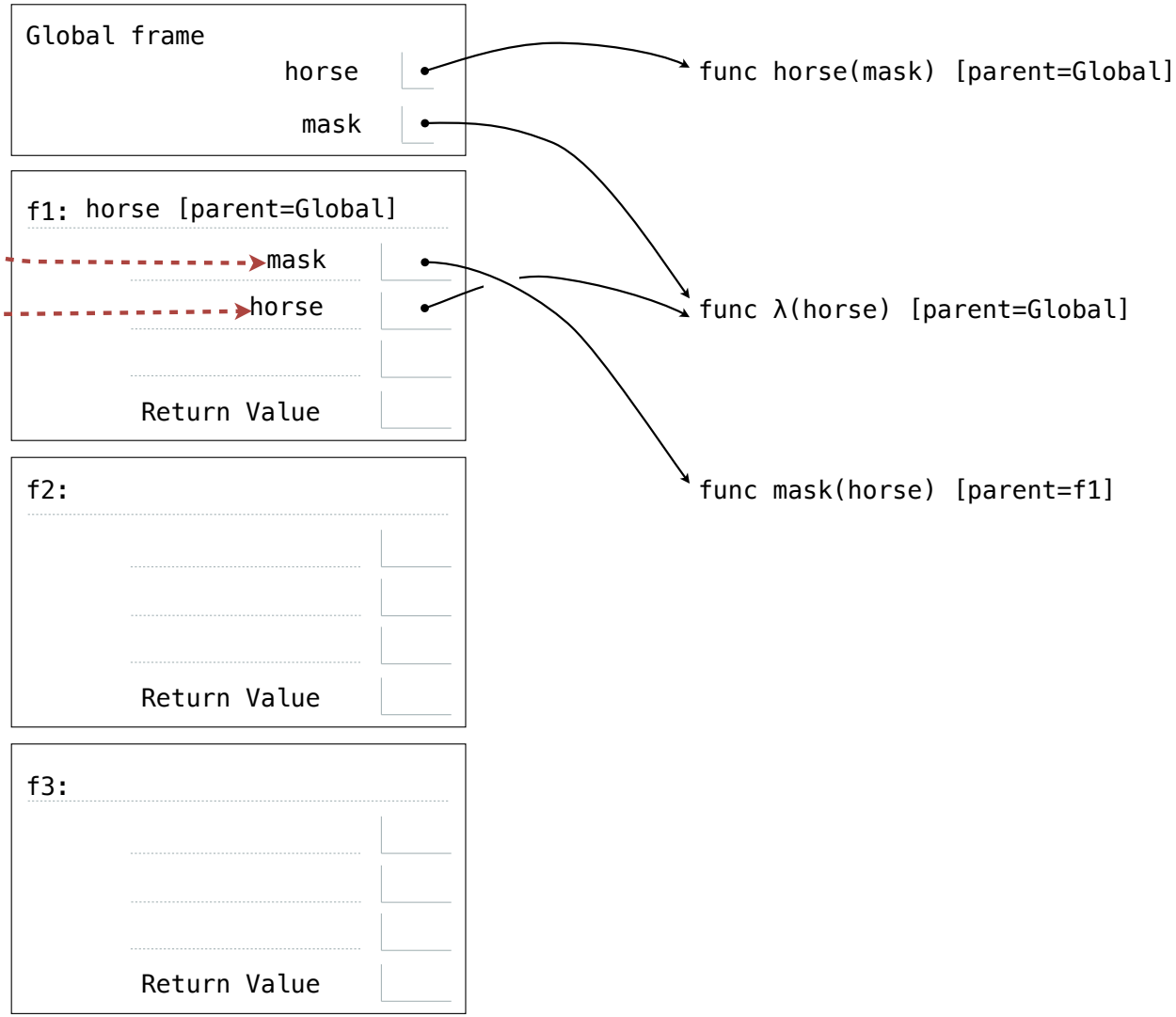
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



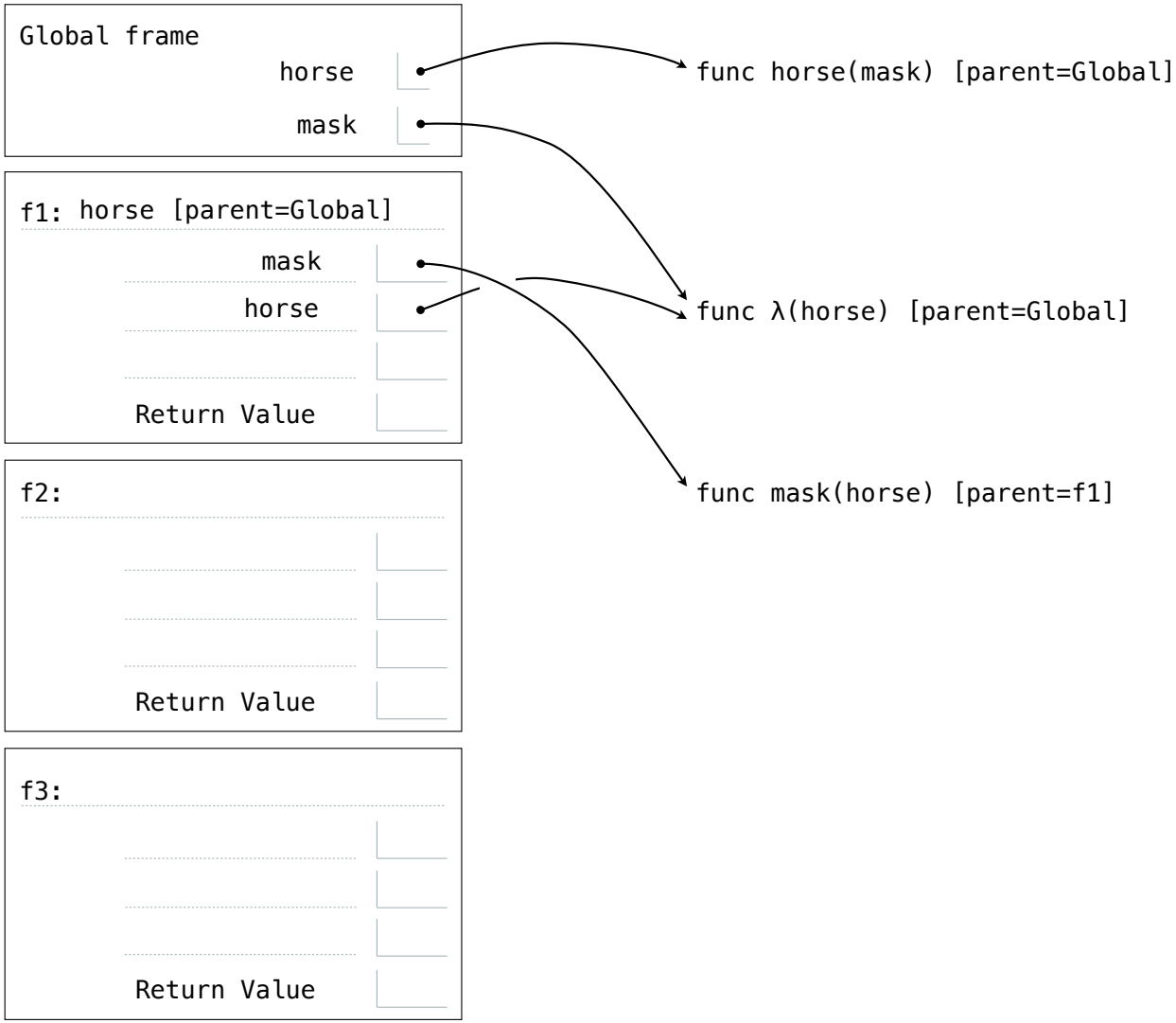
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



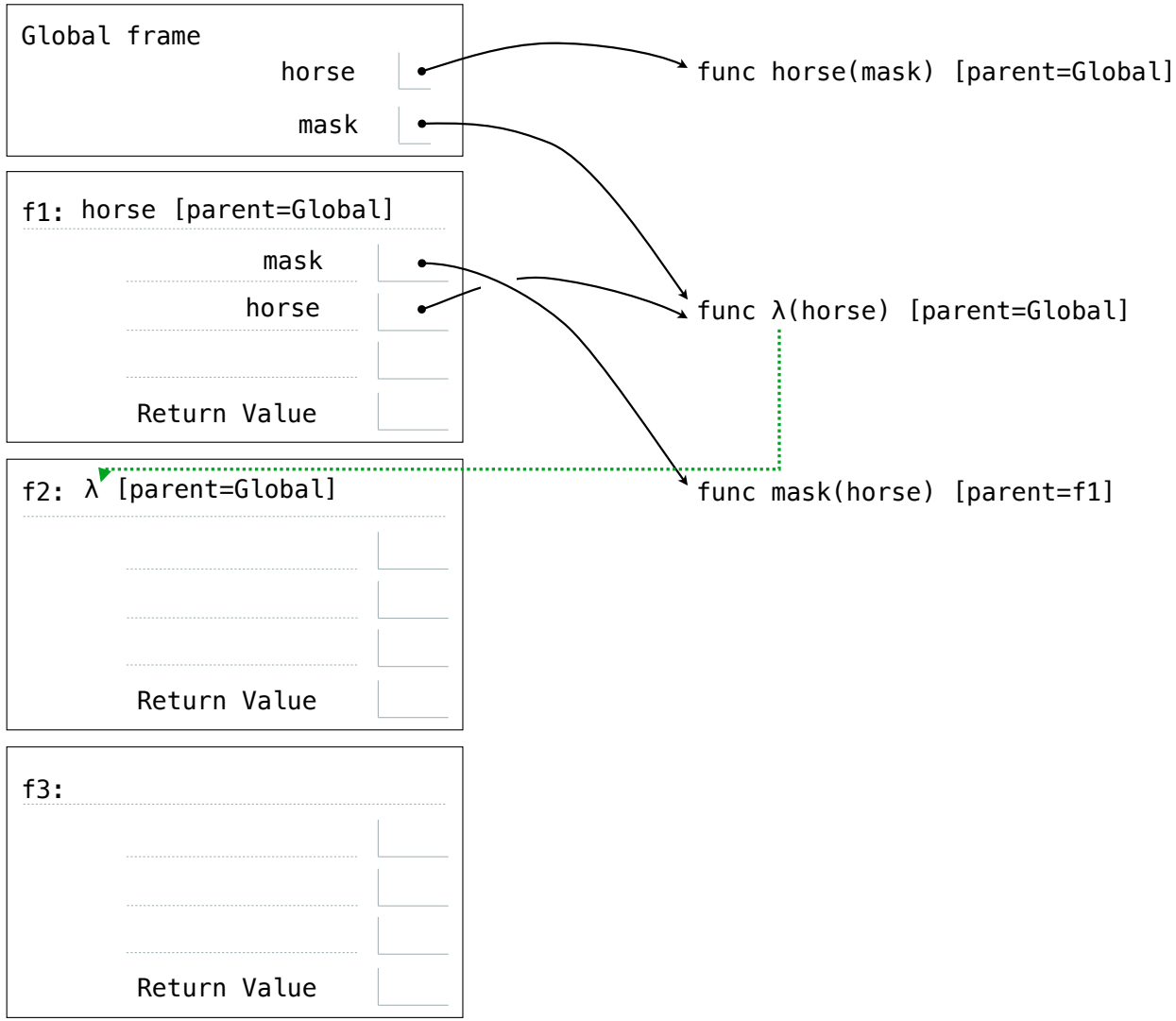
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



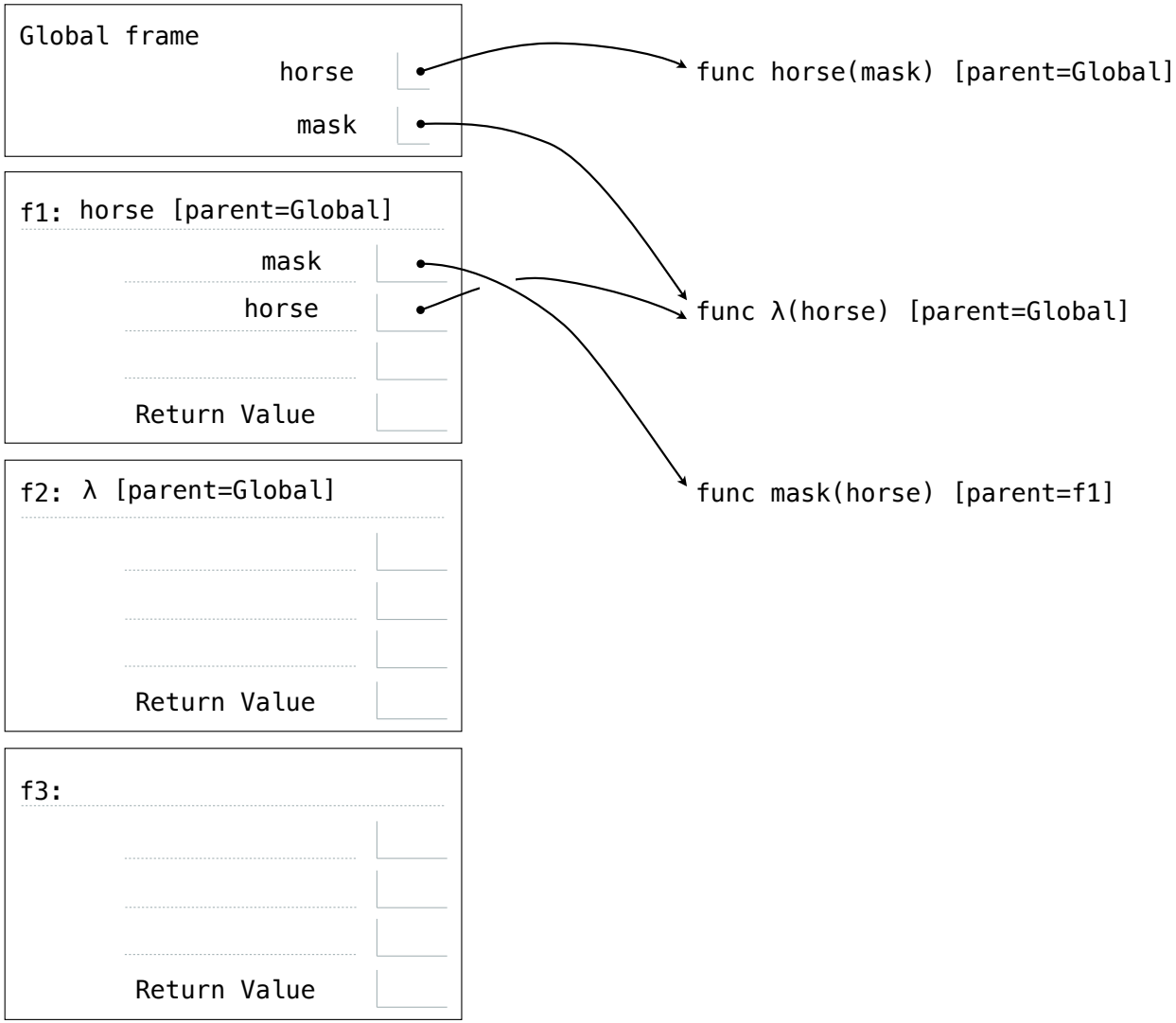
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



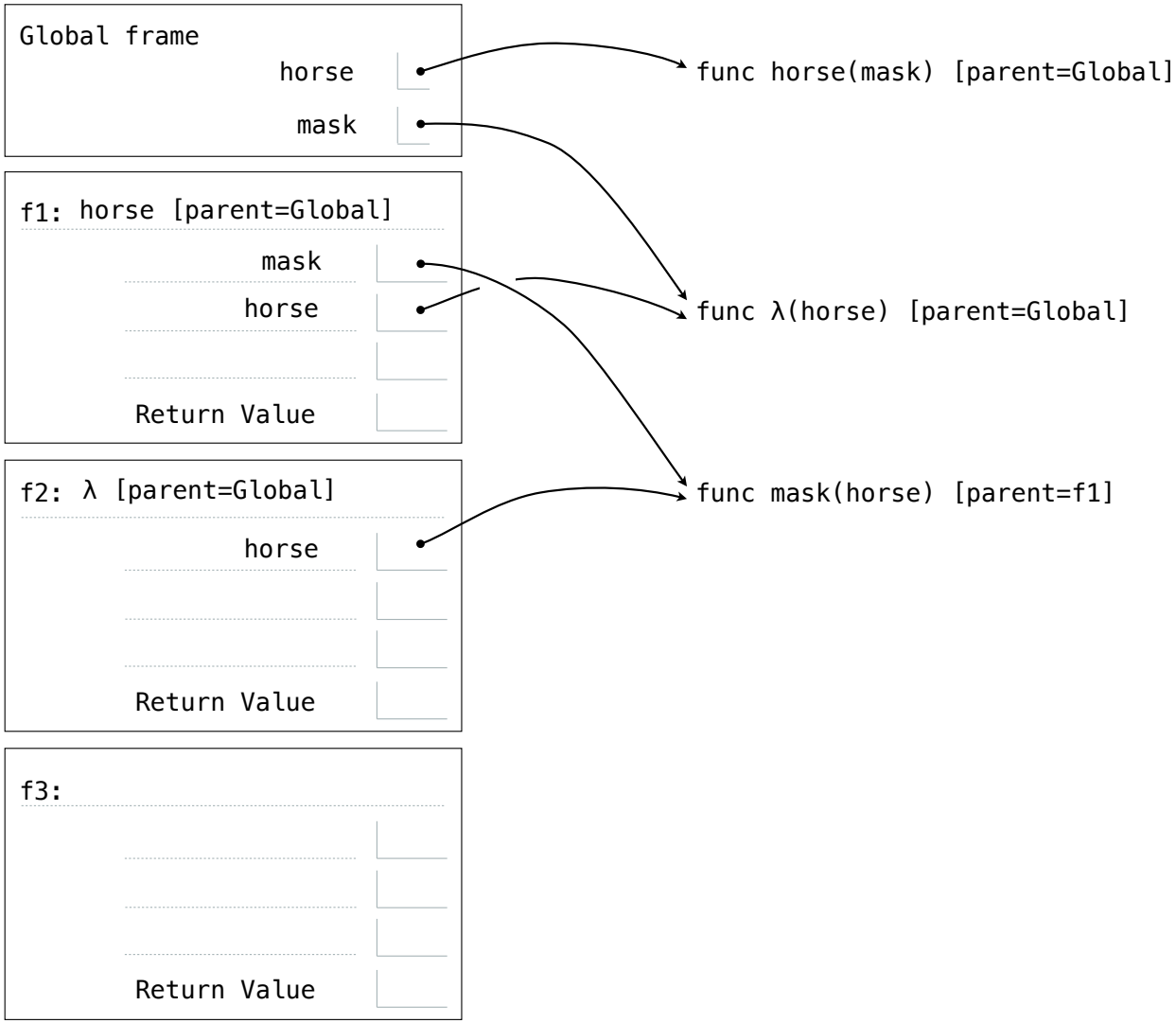
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



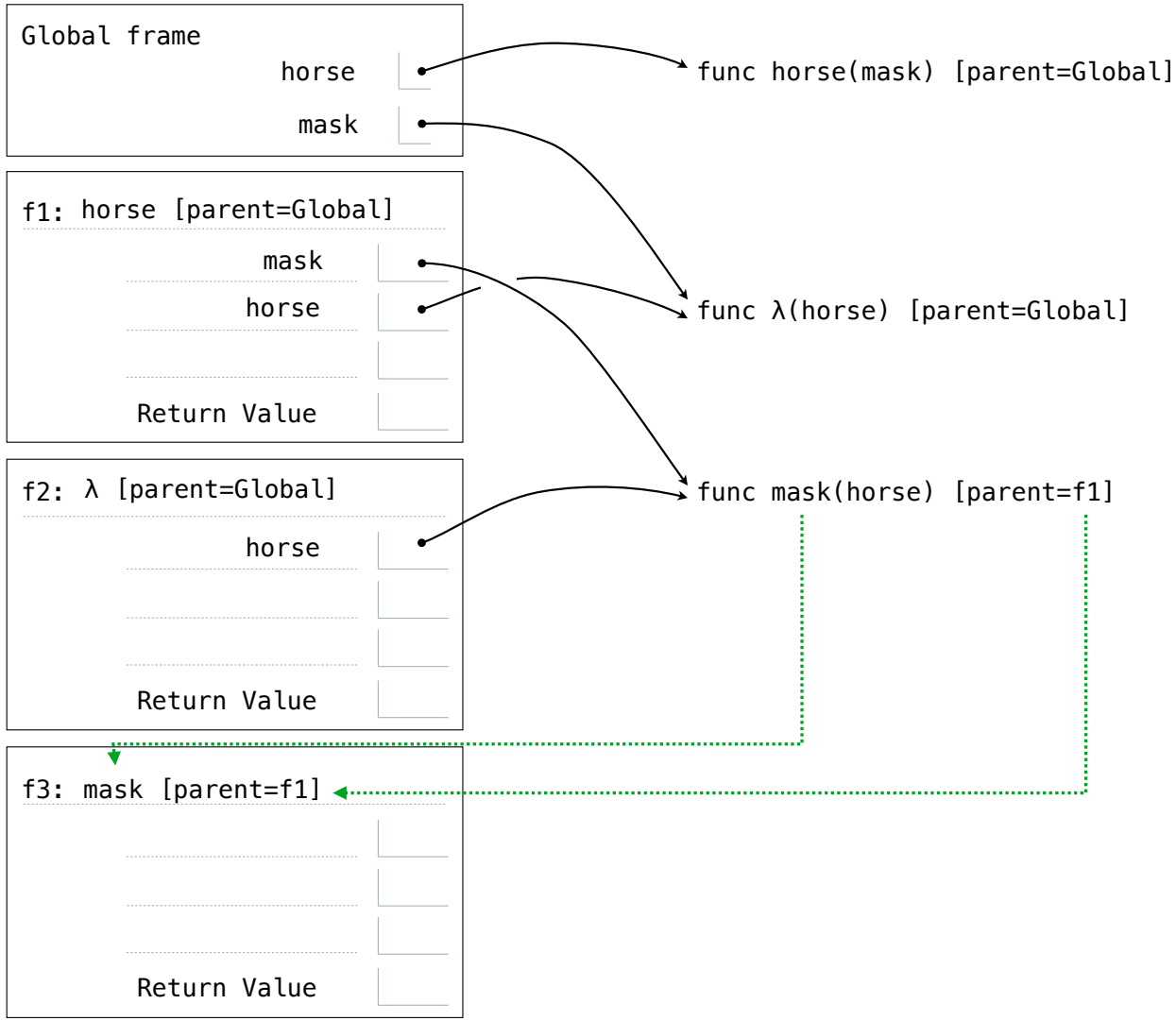
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



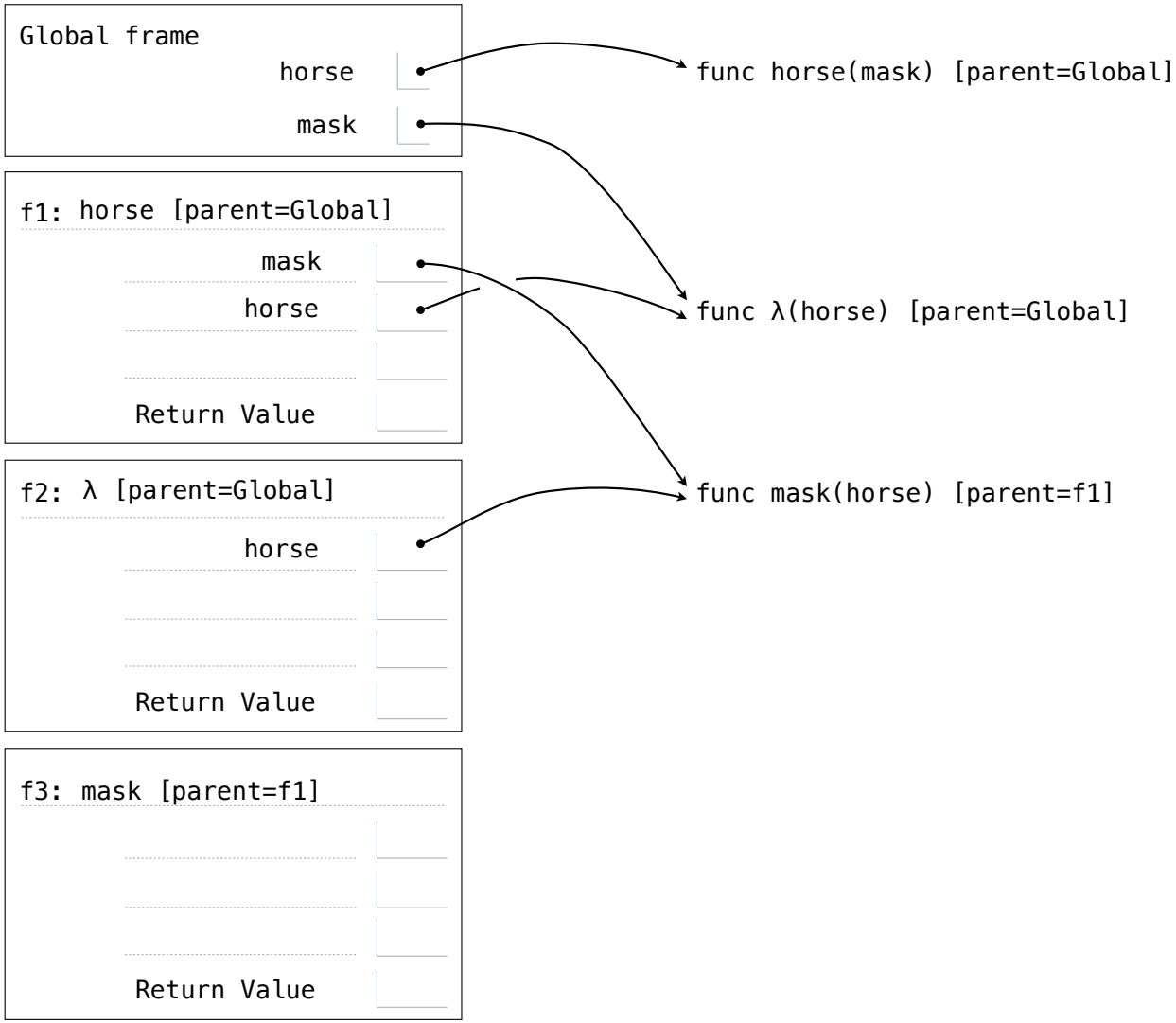
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



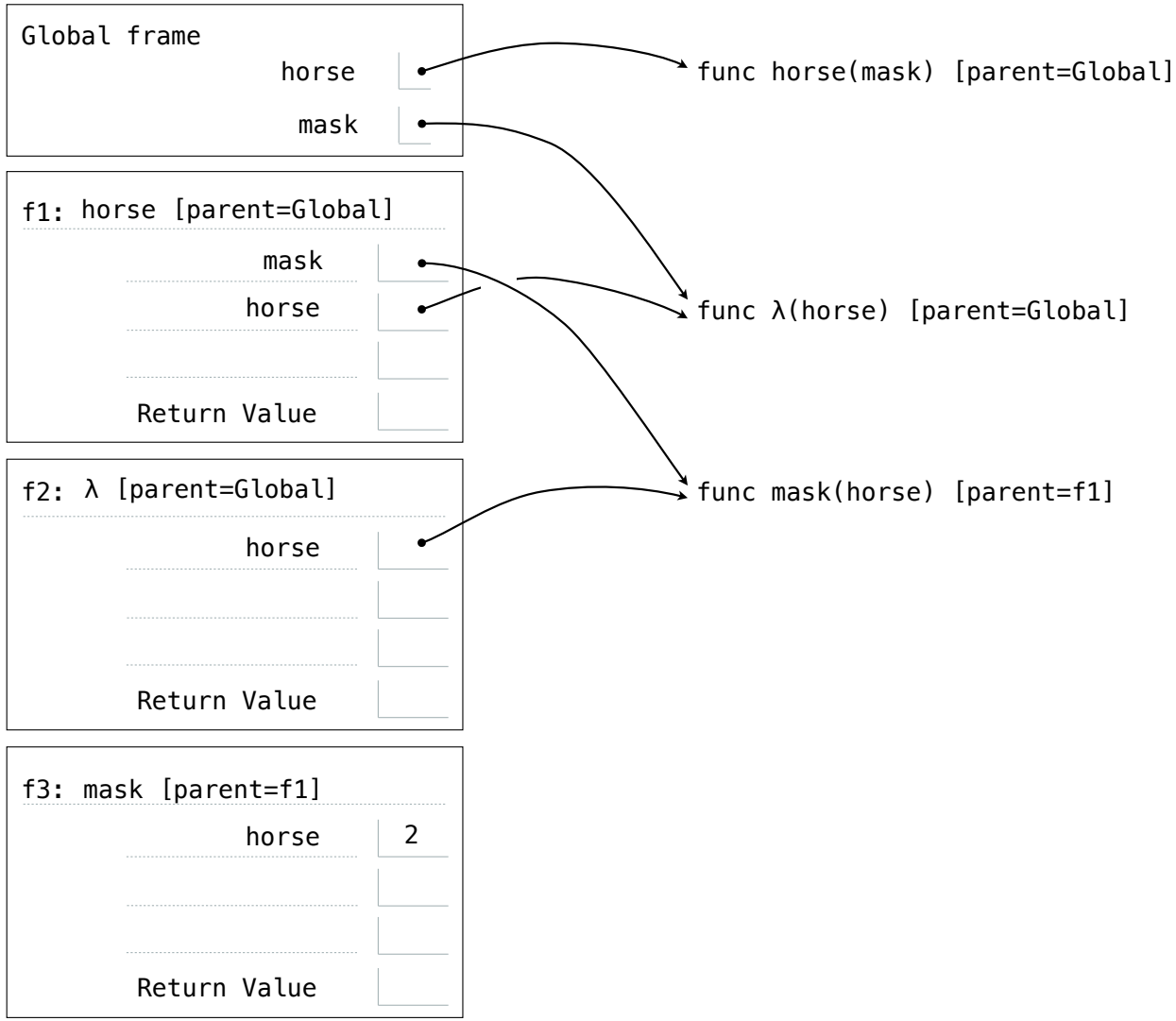
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



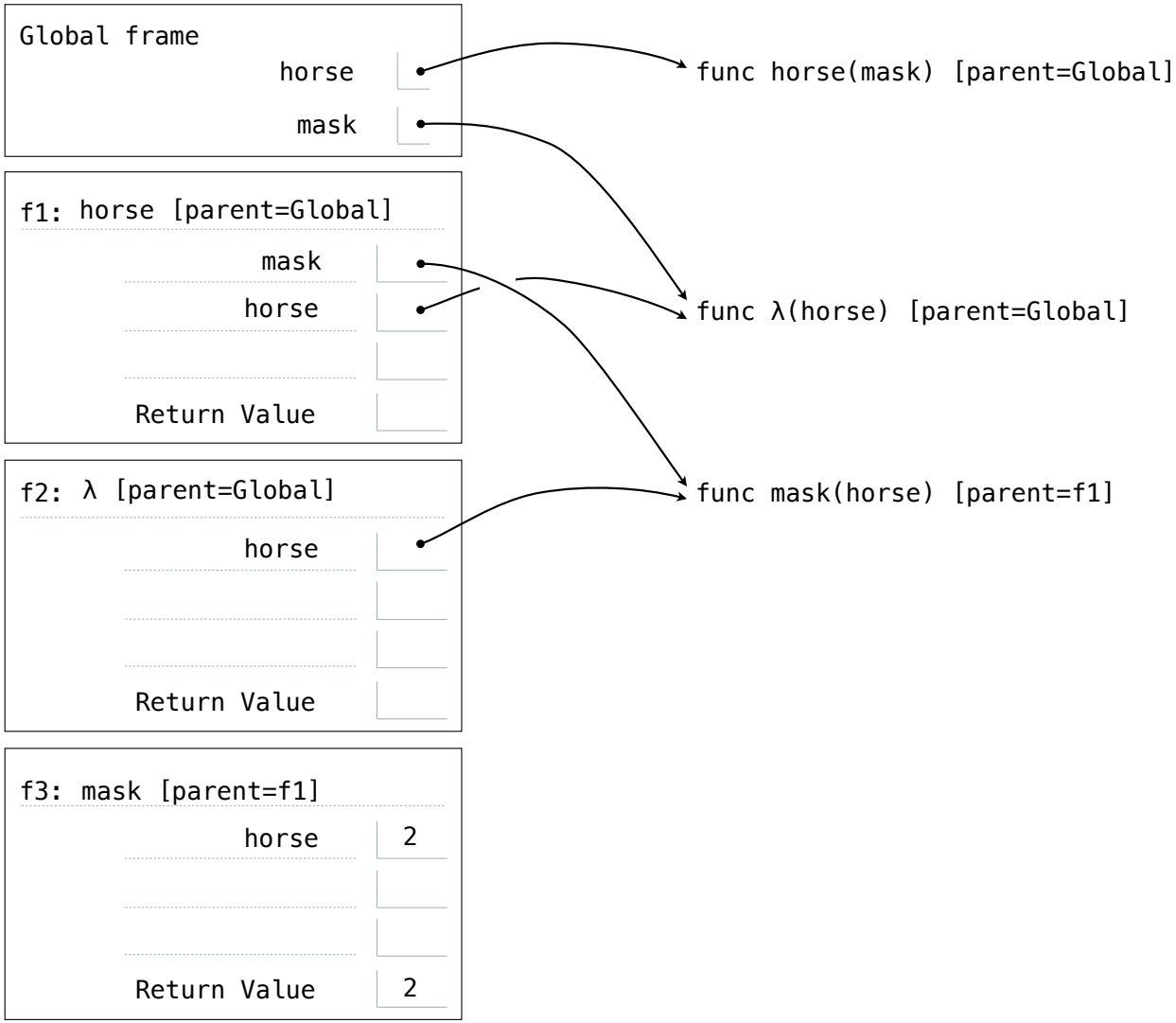
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



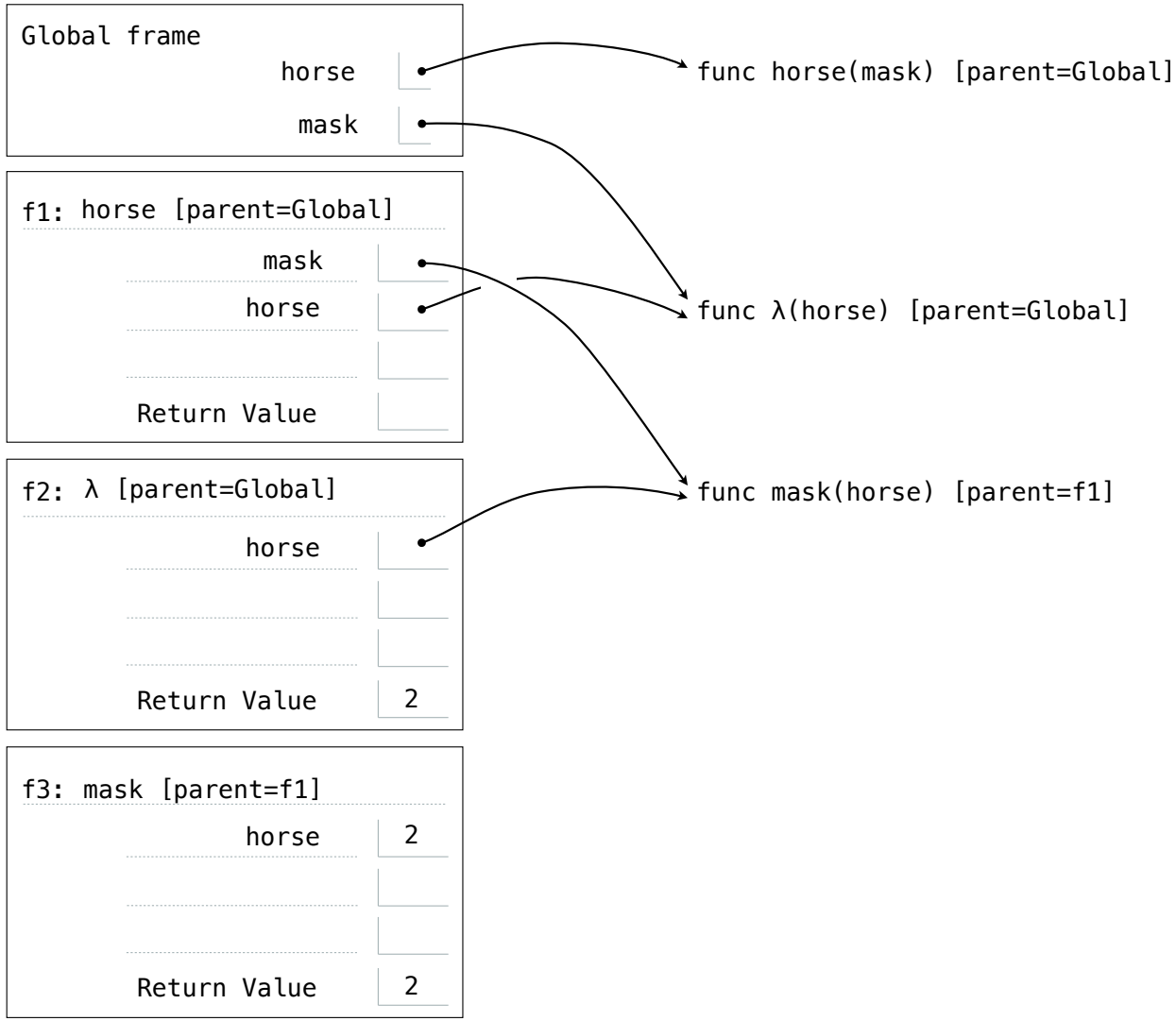
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



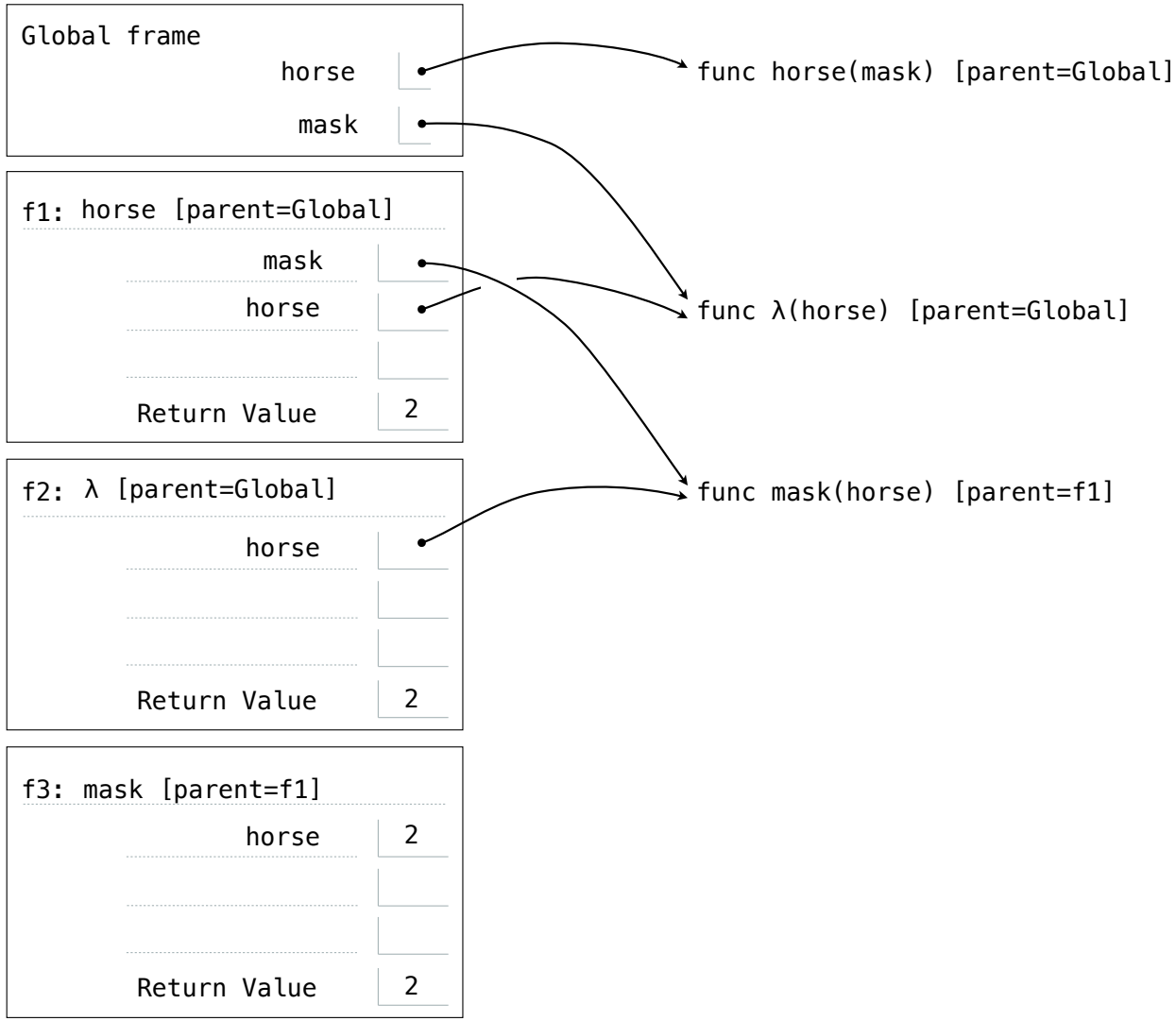
```
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

horse(mask)
```



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

mask = lambda horse: horse(2)

horse(mask)
```



```
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):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ______________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ______________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

Read the description

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ______________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

Read the description

Verify the examples & pick a simple one

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ______________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

Read the description

Verify the examples & pick a simple one

Read the template

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ______________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

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.

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ________________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

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

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ________________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

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

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ________________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

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?

## Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

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

    while ________________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

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):
    """Return all digits of non-negative N
    that are DIGIT, for some
    non-negative DIGIT less than 10.

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

    while ________________________________:

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

        if ______________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

231

3

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.

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

    while ________________________________:

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

        if ________________________________:

            kept = ______________________

            digits = ____________________

    return ________________________________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are DIGIT, for some
    non-negative DIGIT 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 ____________________________:

            kept = ______________________

            digits = ____________________

    return ______________________________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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 = ____________________

            digits = ____________________

    return ____________________________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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 = ____________________

            digits = ____________________

    return ____________kept____________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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 + last____

            digits = ____________

    return ____kept____
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____

            digits = ____________________

    return ____________kept____________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____

            digits = ____________________

    return ____________kept____________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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 = ____________________

    return ____________kept____________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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 = ____________________

    return ____________kept____________
```

231

4

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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 = ____________________

    return ____________kept____________
```

231

4

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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 + 1____

    return ____kept____
```

231

4

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.

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

    while ____________________________:

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

        if ____________________________:

            kept = ____________________________

            digits = ____________________________

    return ____________________________
```

231

4

n > 0

last != digit

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

digits + 1

kept

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____ + ____last____

            digits = ____________________

    return ____kept____
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____

            digits = ____________________

    return ____________kept____________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____

            digits = ____________________

    return ____kept * 10____________________
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____

            digits = ____digits + 1____

    return ____kept * 10____
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____

            digits = ____digits + 1____

    return ____kept * 10 ** (digits-1)____
```

231

3

21

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):
    """Return all digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT 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____

            digits = ____digits + 1____

    return ____round(kept * 10 ** (digits-1))____
```

231

3

21

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