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

Kaylee 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

Kaylee 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

Kaylee 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

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

Kaylee 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

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

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

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

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

Fall 2017 Winners
Alex Yu and Tanmay Khattar
James Li
Justin Yokota

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

Fall 2017 Winners

Alex Yu and Tanmay Khattar
James Li
Justin Yokota

Spring 2018 Winners

Eric James Michaud
Ziyu Dong
Xuhui Zhou

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

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

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

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

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

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

Jet Situ and Lucas Schaberg
Anthony Han and Hongyi Huang
Arthur Pan and Qingyuan Liu

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

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

Jet Situ and Lucas Schaberg
Anthony Han and Hongyi Huang
Arthur Pan and Qingyuan Liu

Spring 2020 Winners

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

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

Jet Situ and Lucas Schaberg
Anthony Han and Hongyi Huang
Arthur Pan and Qingyuan Liu

Spring 2020 Winners

Andy Dong
Theodor Sion and Anish Kar
Shaun Diem-Lane

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

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

Jet Situ and Lucas Schaberg
Anthony Han and Hongyi Huang
Arthur Pan and Qingyuan Liu

Spring 2020 Winners

Andy Dong
Theodor Sion and Anish Kar
Shaun Diem-Lane

Fall 2020 Winners

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
Jet Situ and Lucas Schaberg
Anthony Han and Hongyi Huang
Arthur Pan and Qingyuan Liu

Spring 2020 Winners
Andy Dong
Theodor Sion and Anish Kar
Shaun Diem-Lane

Fall 2020 Winners

Describing Functions

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

mystery1 prints ______ less than n ______ .

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

One approach:

mystery1 prints ______ less than n ______ .

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

One approach:

1. Read the code

mystery1 prints ______ less than n ______ .

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

One approach:

1. Read the code
2. Read the description options

mystery1 prints ______ less than n ______ .

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

mystery1 prints ______ less than n ______ .

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

mystery1 prints ______ less than n ______ .

mystery1 prints all odd numbers less than n that George likes.

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

```
likes = is_prime
n = 8
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

mystery1 prints ______ less than n ______ .

mystery1 prints all odd numbers less than n that George likes.

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

```
likes = is_prime
n = 8
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

mystery1 prints ______ less than n ______ .

~~**mystery1** prints all odd numbers less than n that George likes.~~

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

```
likes = is_prime
n = 8
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

all odd numbers

mystery1 prints ______ less than n ______ .

~~**mystery1** prints all odd numbers less than n that George likes.~~

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    while k < n:
        if likes(n):
            print(k)
        k = k + 2
```

```
likes = is_prime
n = 8
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

all odd numbers ▼ but only if George likes n ▼

mystery1 prints ______ less than n ______ .

~~**mystery1** prints all odd numbers less than n that George likes.~~

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery2(n):
    i, j, k = 0, None, None
    while i < n:
        if likes(i):
            if j != None and (k == None or i - j < k):
                k = i - j
            j = i
        i = i + 1
    return k
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

mystery 2 returns ______ or returns None if ______ .

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery2(n):
    i, j, k = 0, None, None
    while i < n:
        if likes(i):
            if j != None and (k == None or i - j < k):
                k = i - j
            j = i
        i = i + 1
    return k
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

the smallest difference between
two positive integers below n
that George likes

mystery 2 returns ______ or returns None if ______ .

Boolean Favorites

```
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery2(n):
    i, j, k = 0, None, None
    while i < n:
        if likes(i):
            if j != None and (k == None or i - j < k):
                k = i - j
            j = i
        i = i + 1
    return k
```

One approach:

1. Read the code
2. Read the description options
3. Consider an example

the smallest difference between two positive integers below n that George likes

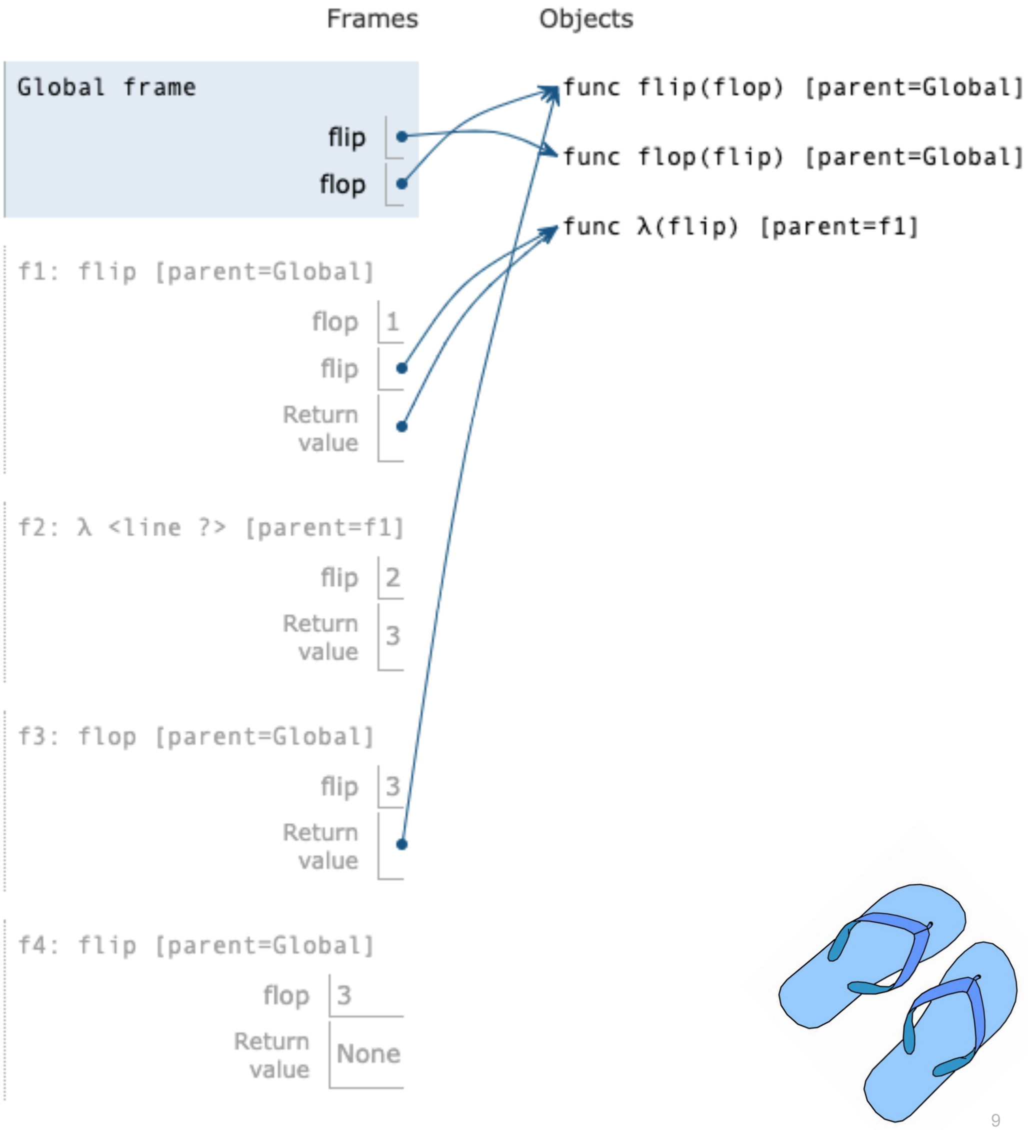
There are no two such integers

mystery 2 returns ______ or returns None if ______ .

Generating Environment Diagram

A Day at the Beach

```
def flip(flop):
    if ______:

        ______
    flip = ______
    return flip

def flop(flip):
    return flop


______


flip(____)(3)
```



Frames

Objects

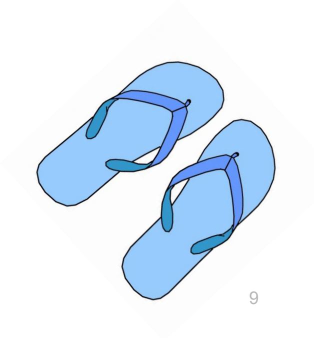
Global frame
flip: func flip(flop) [parent=Global]
flop: func flop(flip) [parent=Global]

f1: flip [parent=Global]
flop: 1
flip: func λ(flip) [parent=f1]
Return value: func λ(flip) [parent=f1]

f2: λ <line ?> [parent=f1]
flip: 2
Return value: 3

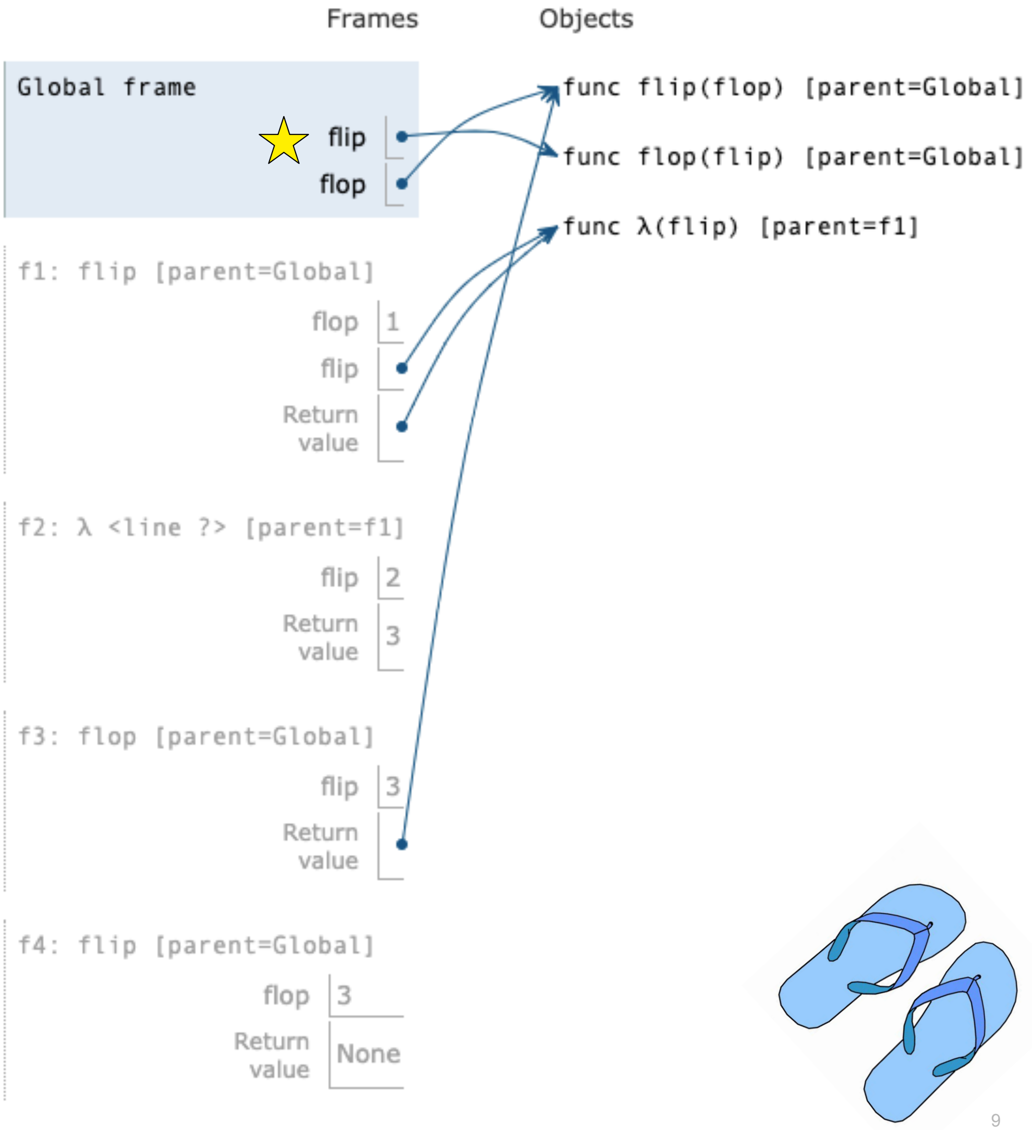
f3: flop [parent=Global]
flip: 3
Return value: func flip(flop) [parent=Global]

f4: flip [parent=Global]
flop: 3
Return value: None



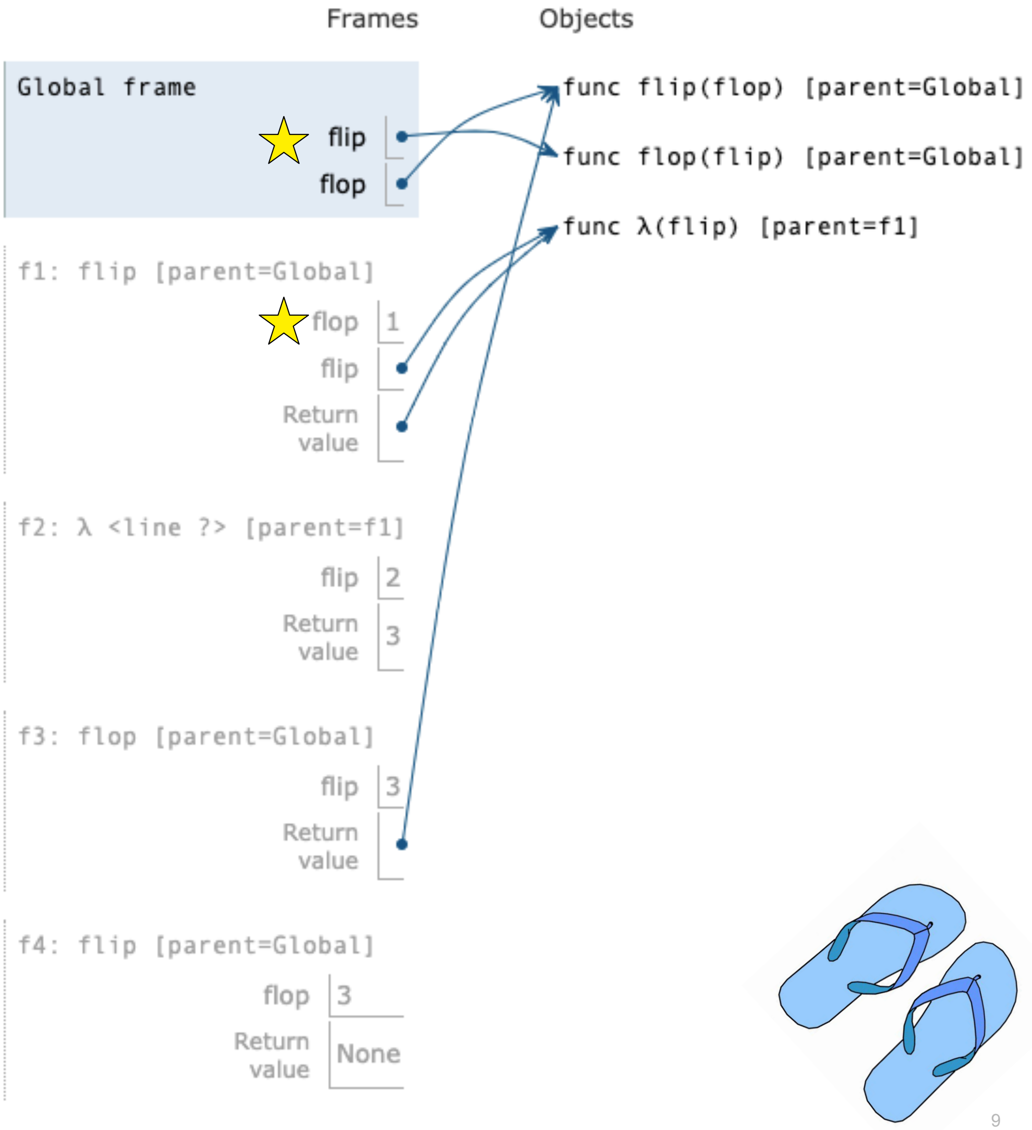
A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



Frames

Objects

Global frame

☆ flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

☆ flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

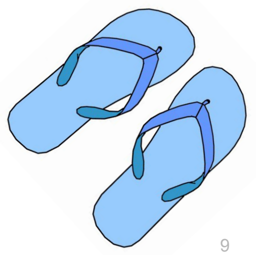
flip 3

Return value

f4: flip [parent=Global]

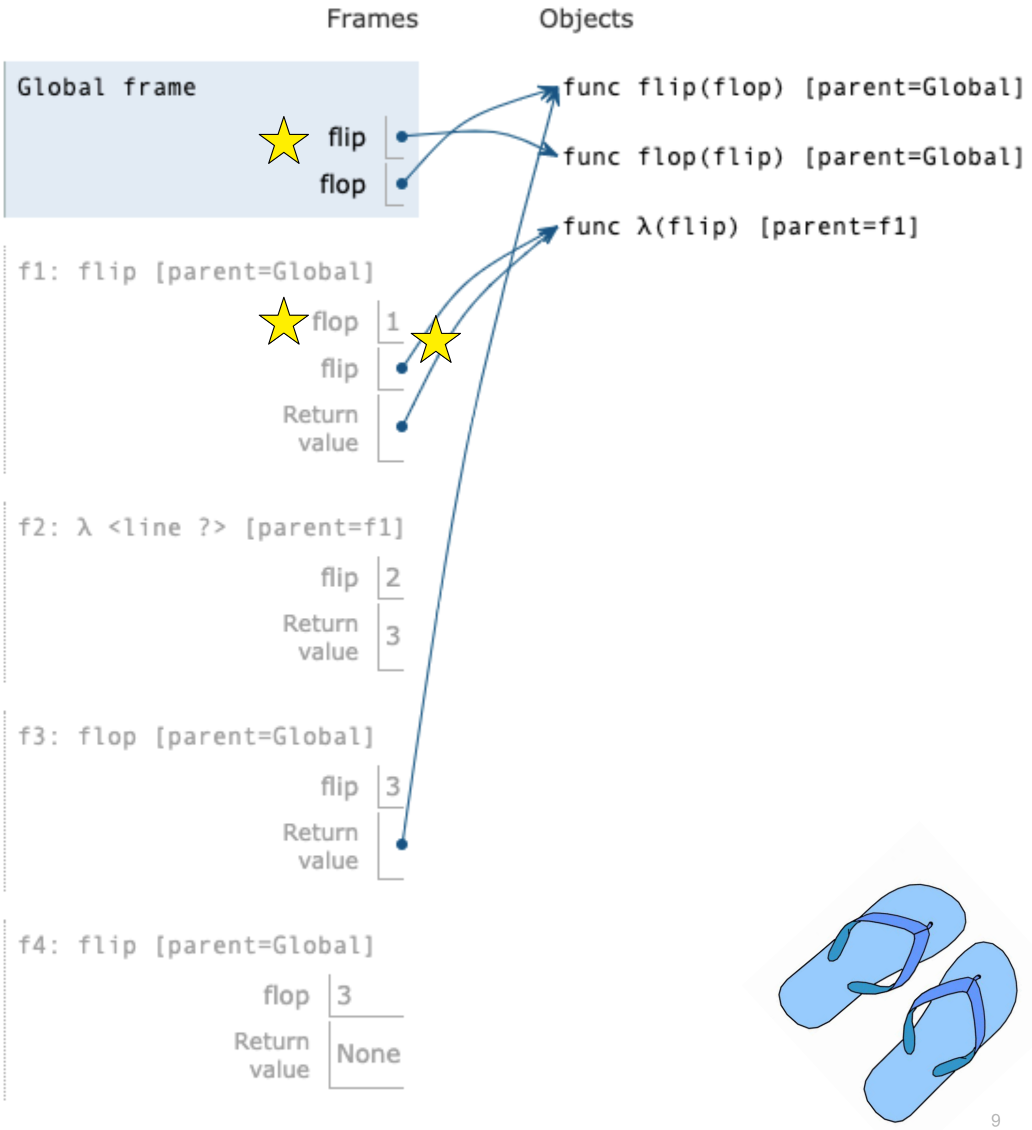
flop 3

Return value None



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



Frames

Objects

Global frame

☆ flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

☆ flop 1

flip ☆

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value

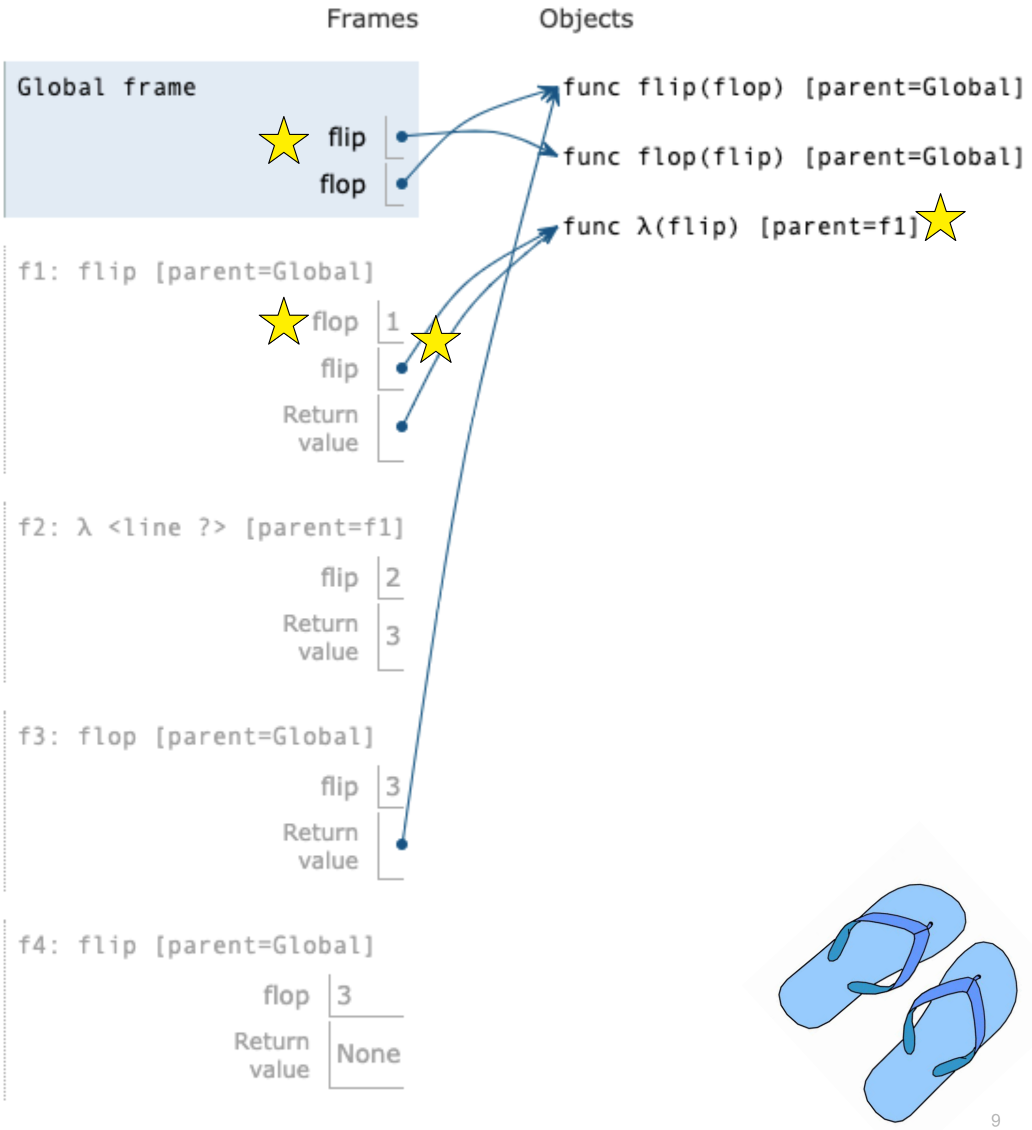
f4: flip [parent=Global]

flop 3

Return value None

A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

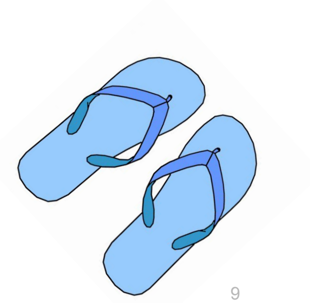
flip 3

Return value

f4: flip [parent=Global]

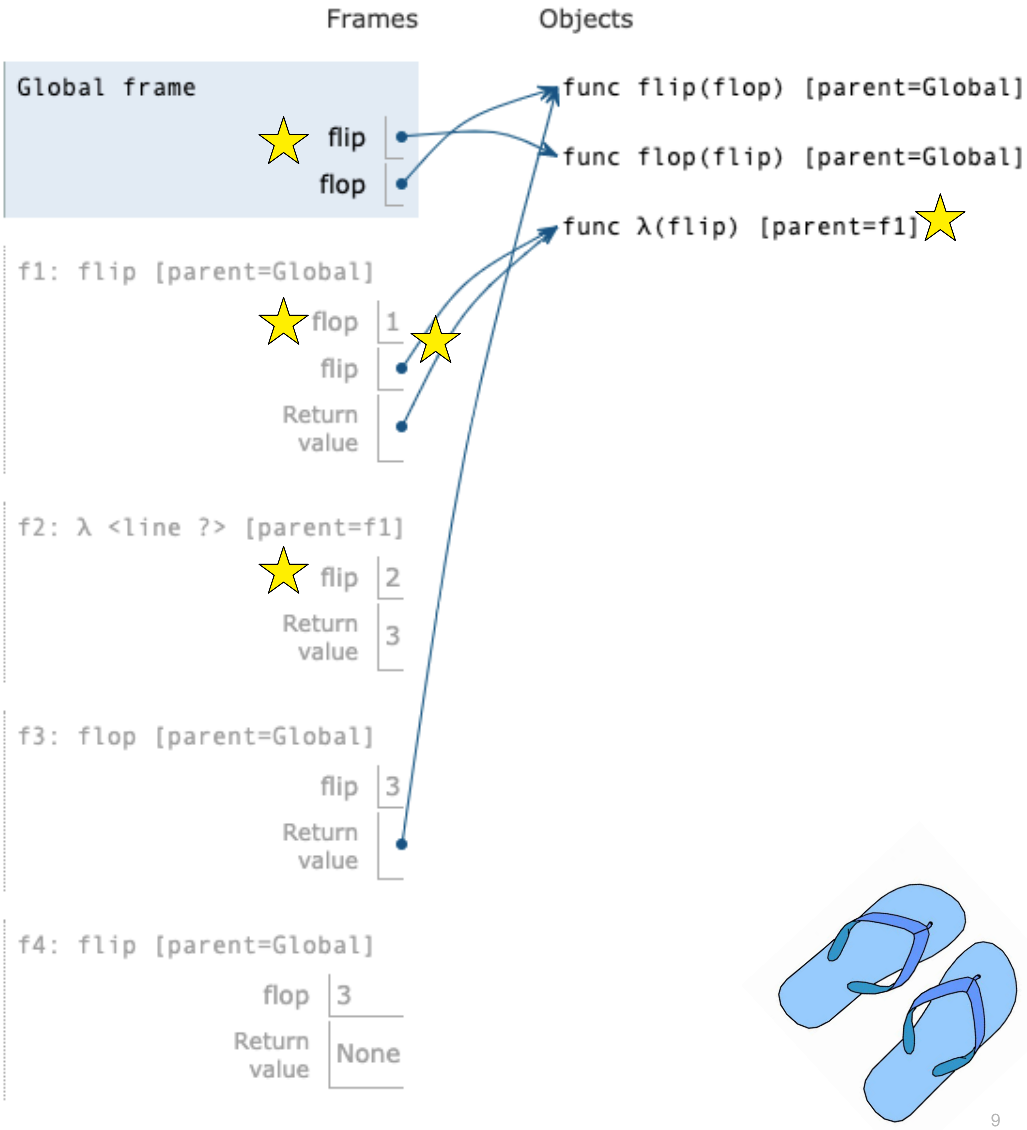
flop 3

Return value None



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```

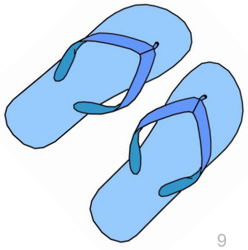


Frames

Objects

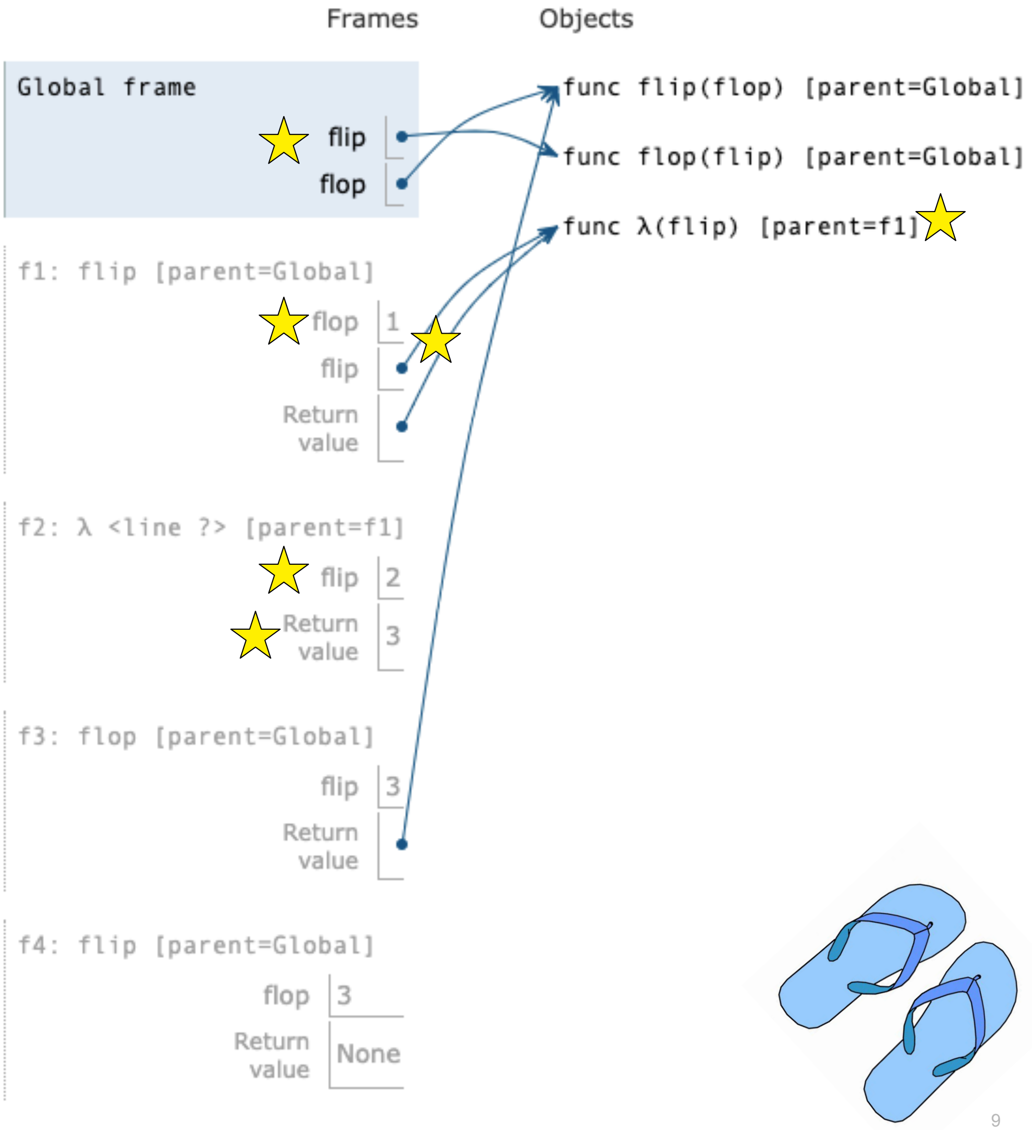
```
Global frame
        flip  •
        flop  •

f1: flip [parent=Global]
        flop  1
        flip  •
      Return
       value  •

f2: λ <line ?> [parent=f1]
        flip  2
      Return
       value  3

f3: flop [parent=Global]
        flip  3
      Return
       value  •

f4: flip [parent=Global]
        flop  3
      Return
       value  None
```

```
func flip(flop) [parent=Global]
func flop(flip) [parent=Global]
func λ(flip) [parent=f1]
```



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

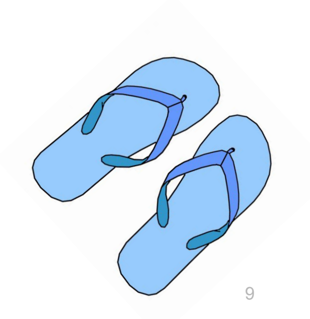
flip 3

Return value

f4: flip [parent=Global]

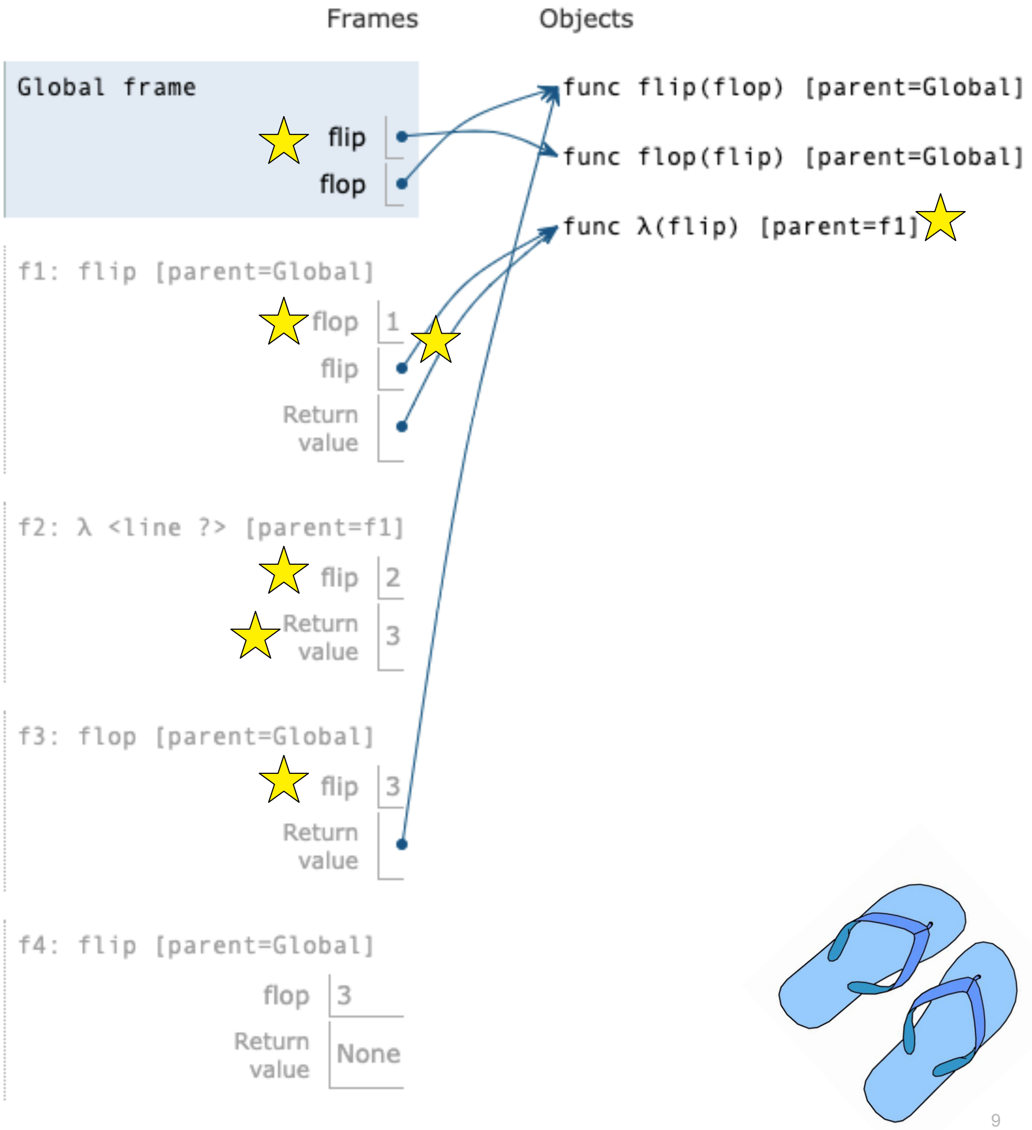
flop 3

Return value None



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

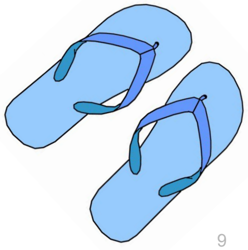
flip 3

Return value

f4: flip [parent=Global]

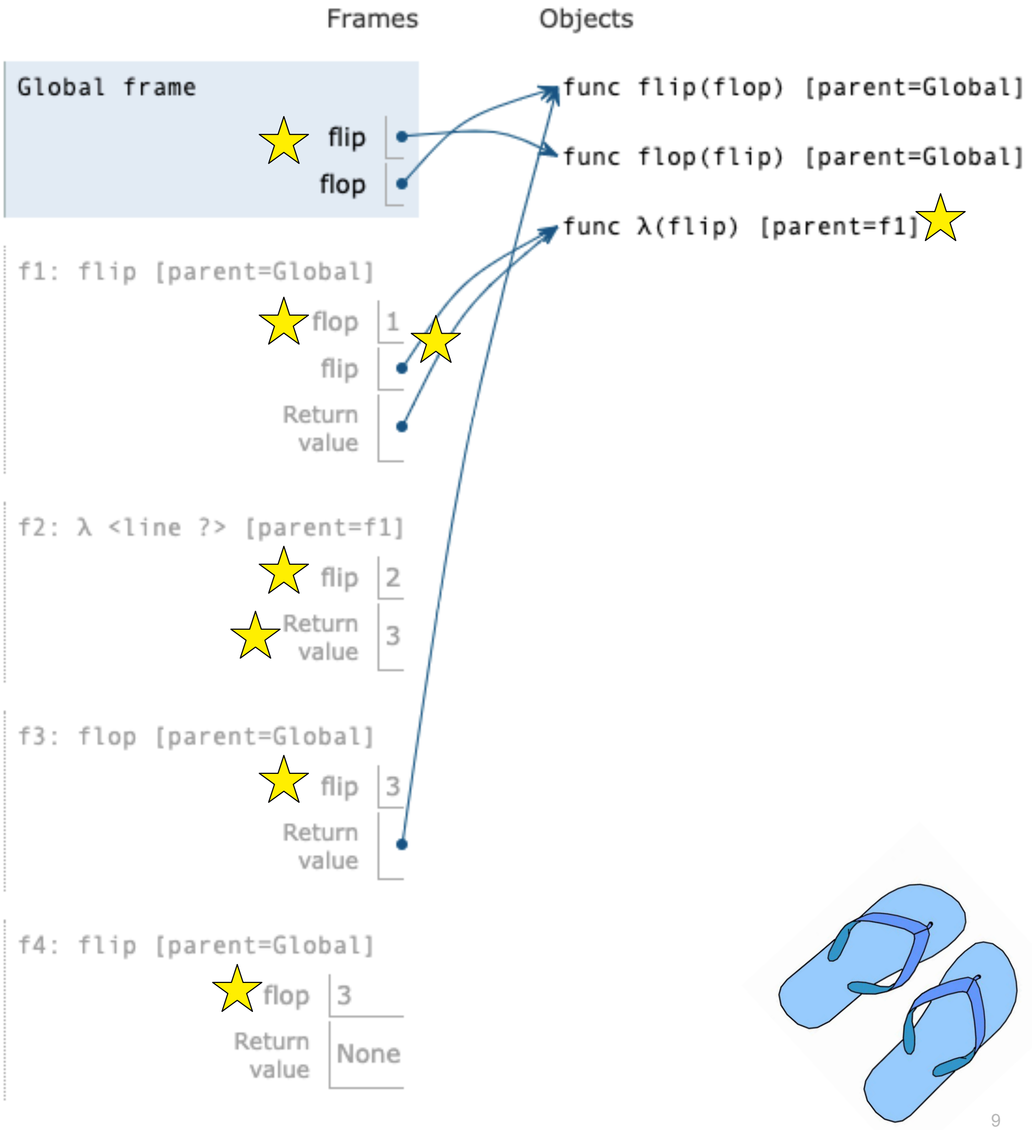
flop 3

Return value None



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value

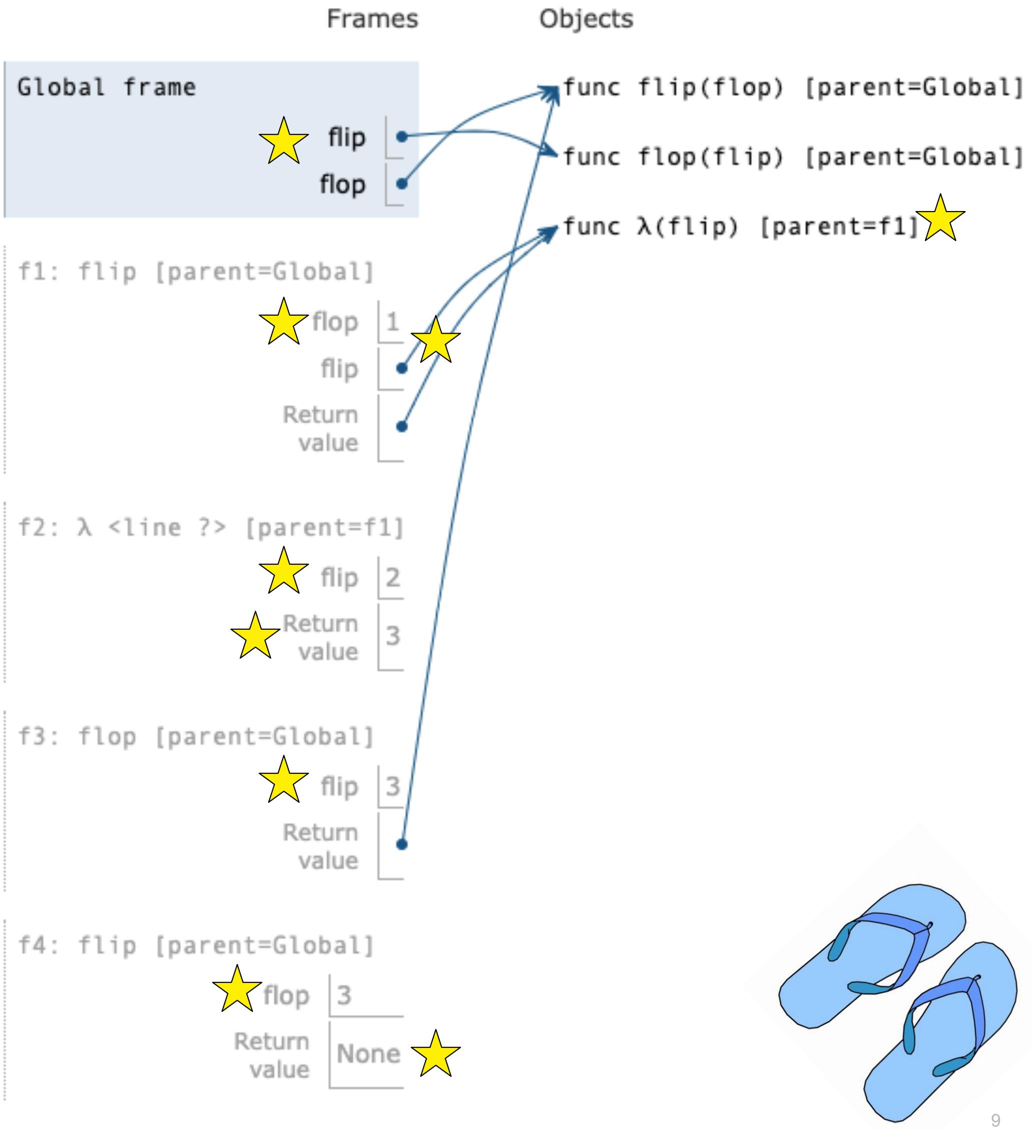
f4: flip [parent=Global]

flop 3

Return value None

A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop


______


flip(____)(3)
```



Frames

Objects

Global frame

☆ flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1] ☆

f1: flip [parent=Global]

☆ flop 1

flip

Return value ☆

f2: λ <line ?> [parent=f1]

☆ flip 2

☆ Return value 3

f3: flop [parent=Global]

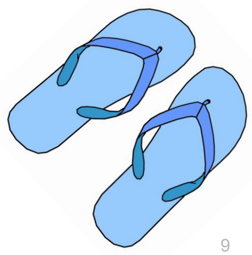
☆ flip 3

Return value

f4: flip [parent=Global]

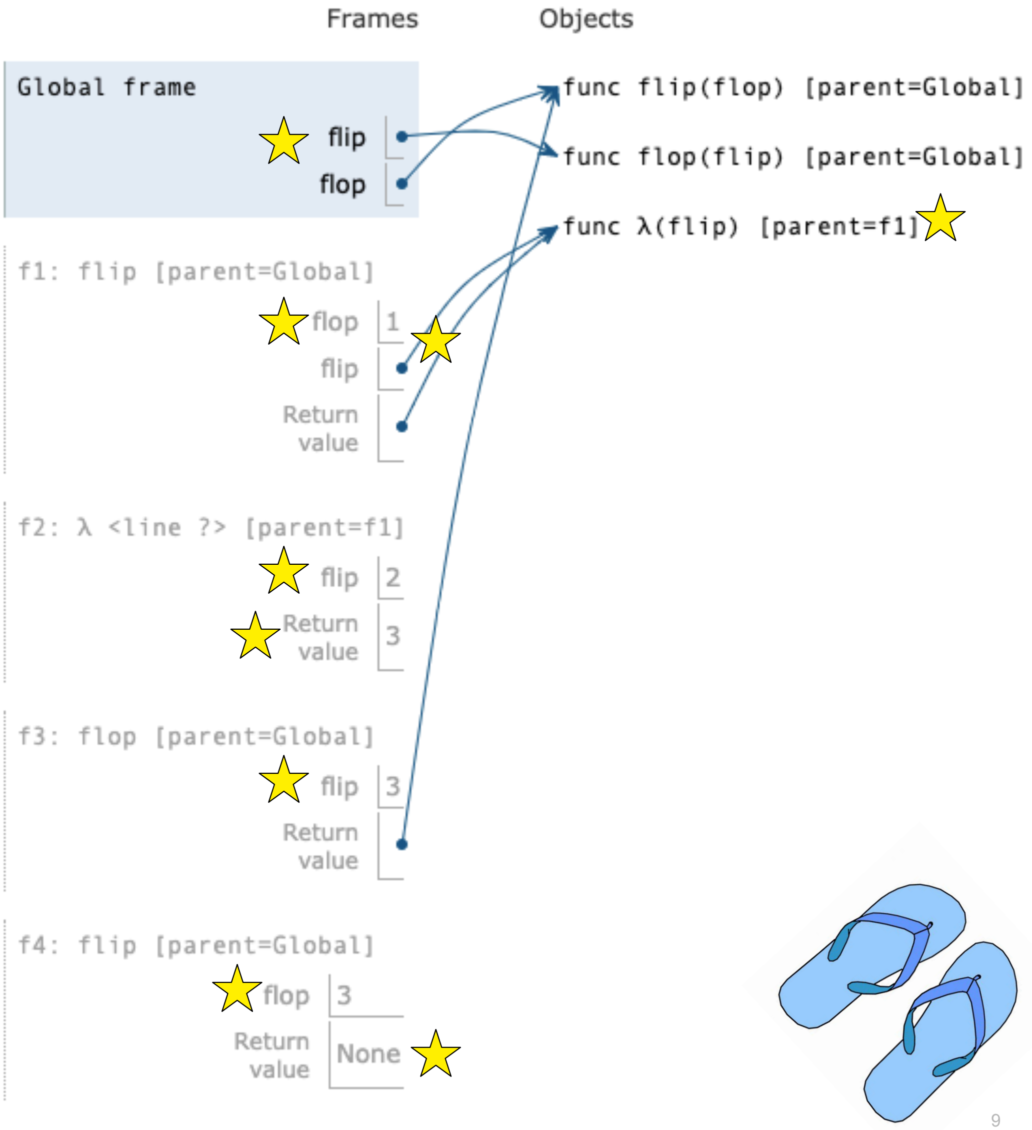
☆ flop 3

Return value None ☆



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip
______

flip(____)(3)
```



Frames

Objects

Global frame
flip
flop

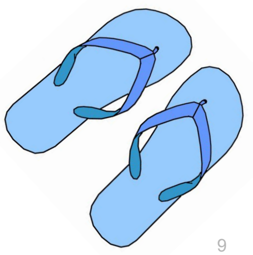
func flip(flop) [parent=Global]
func flop(flip) [parent=Global]
func λ(flip) [parent=f1]

f1: flip [parent=Global]
flop 1
flip
Return value

f2: λ <line ?> [parent=f1]
flip 2
Return value 3

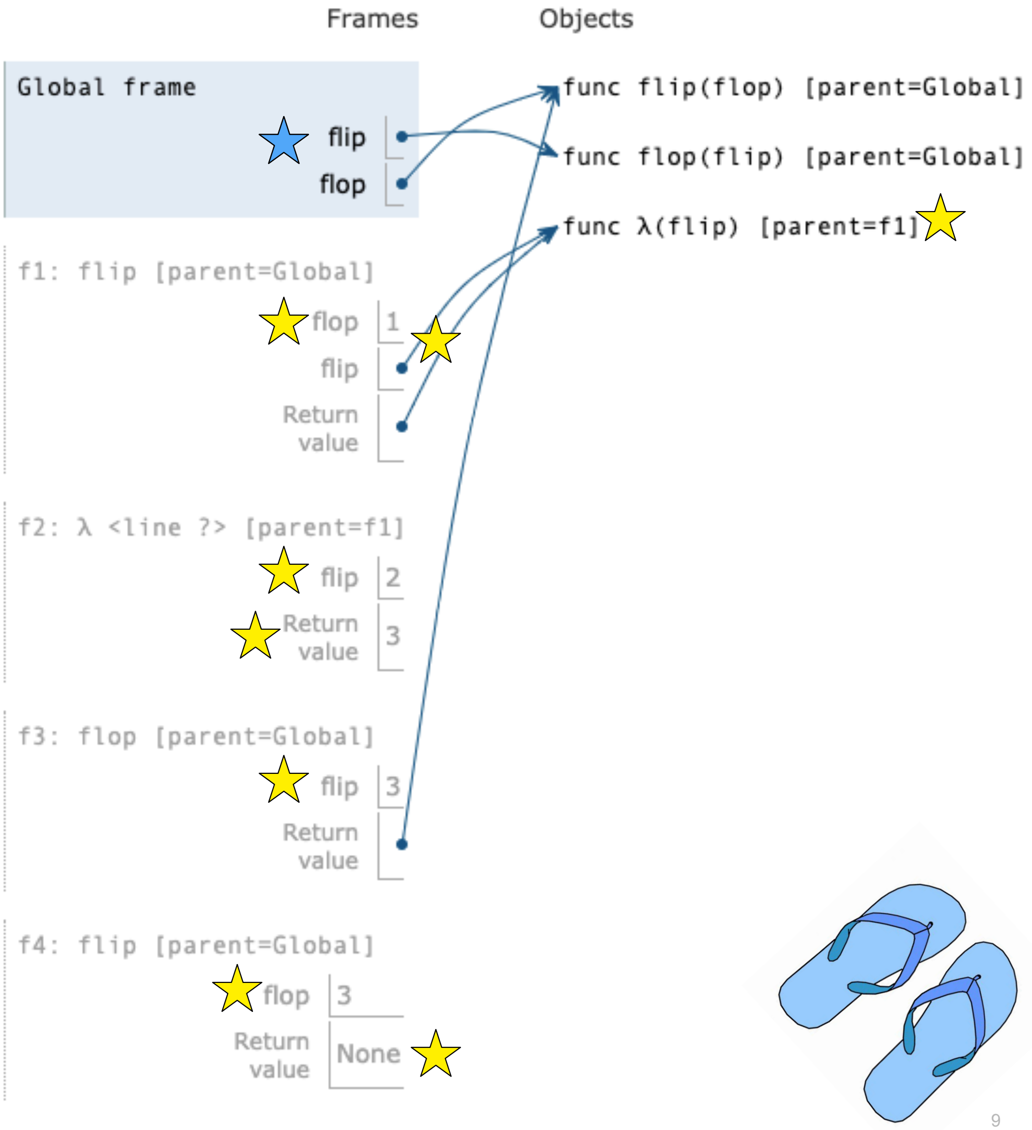
f3: flop [parent=Global]
flip 3
Return value

f4: flip [parent=Global]
flop 3
Return value None



A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip
______

flip(____)(3)
```



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

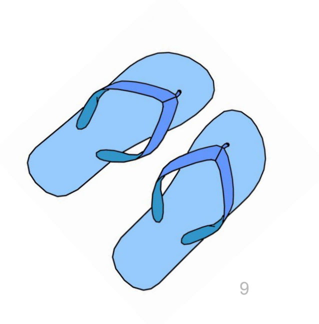
flip 3

Return value

f4: flip [parent=Global]

flop 3

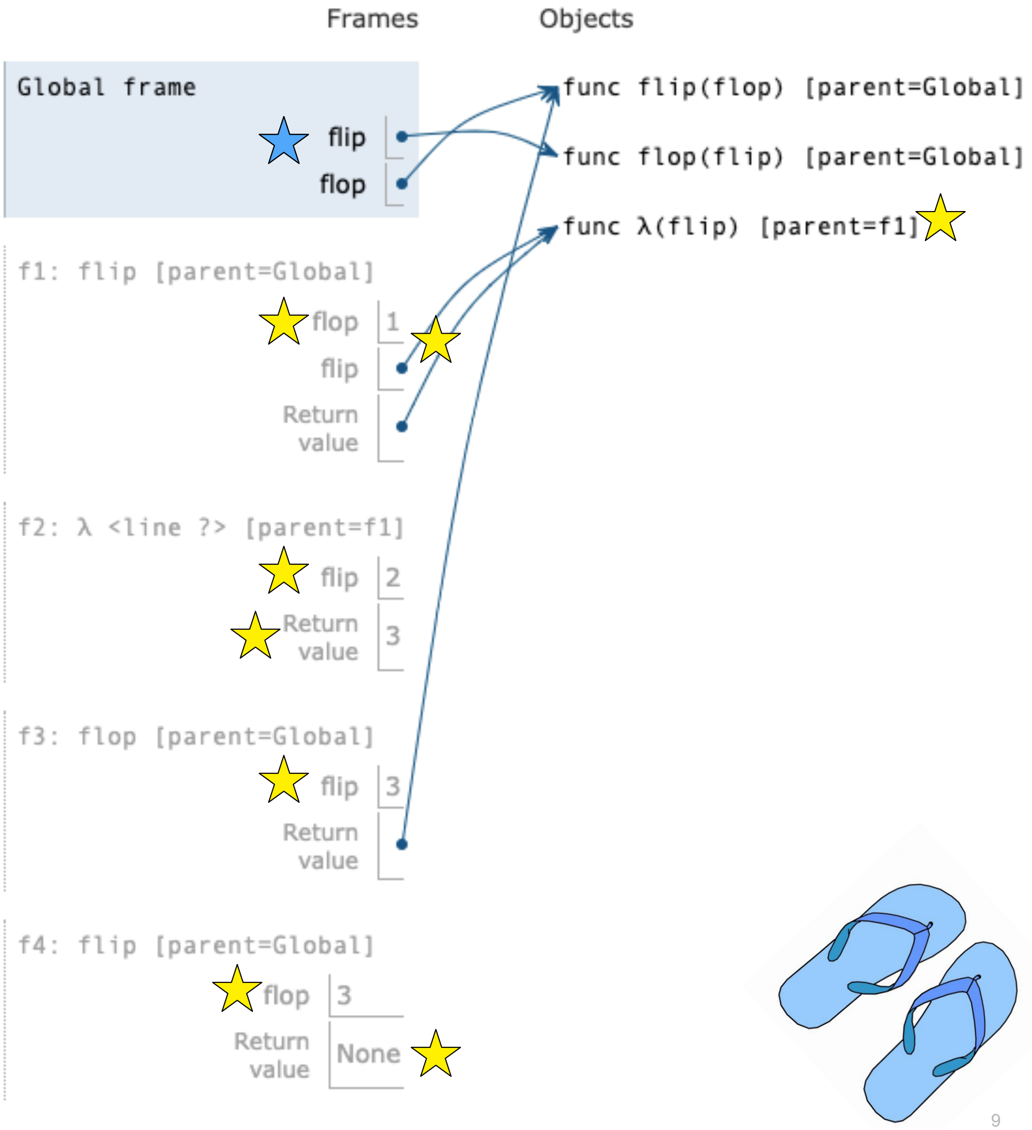
Return value None

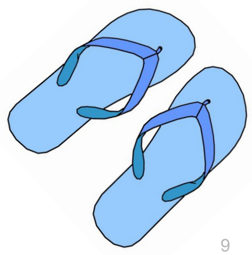


A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip
______

flip(____)(3)
```

flop(1)

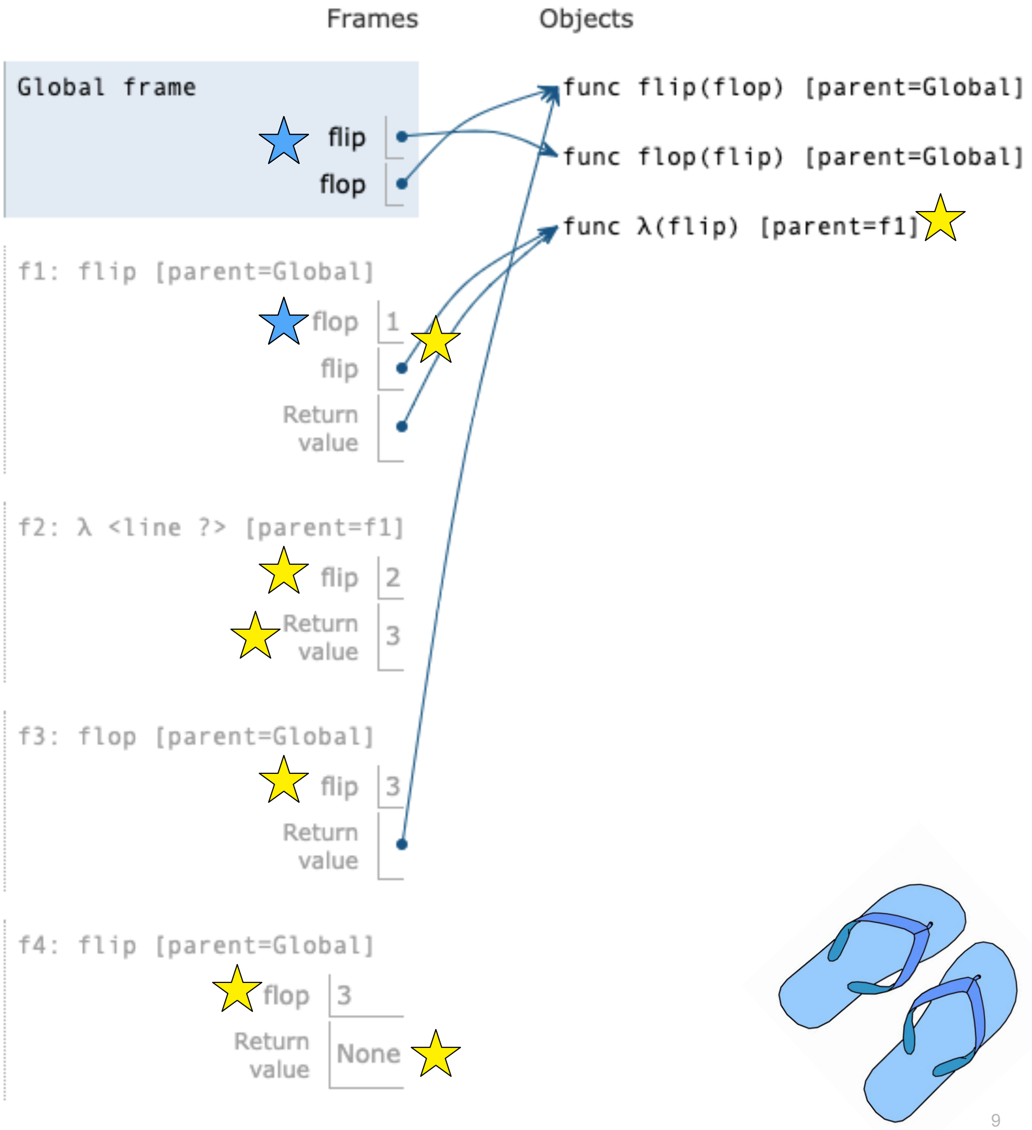




A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip
______

flip(____)(3)
```

flop(1)



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value

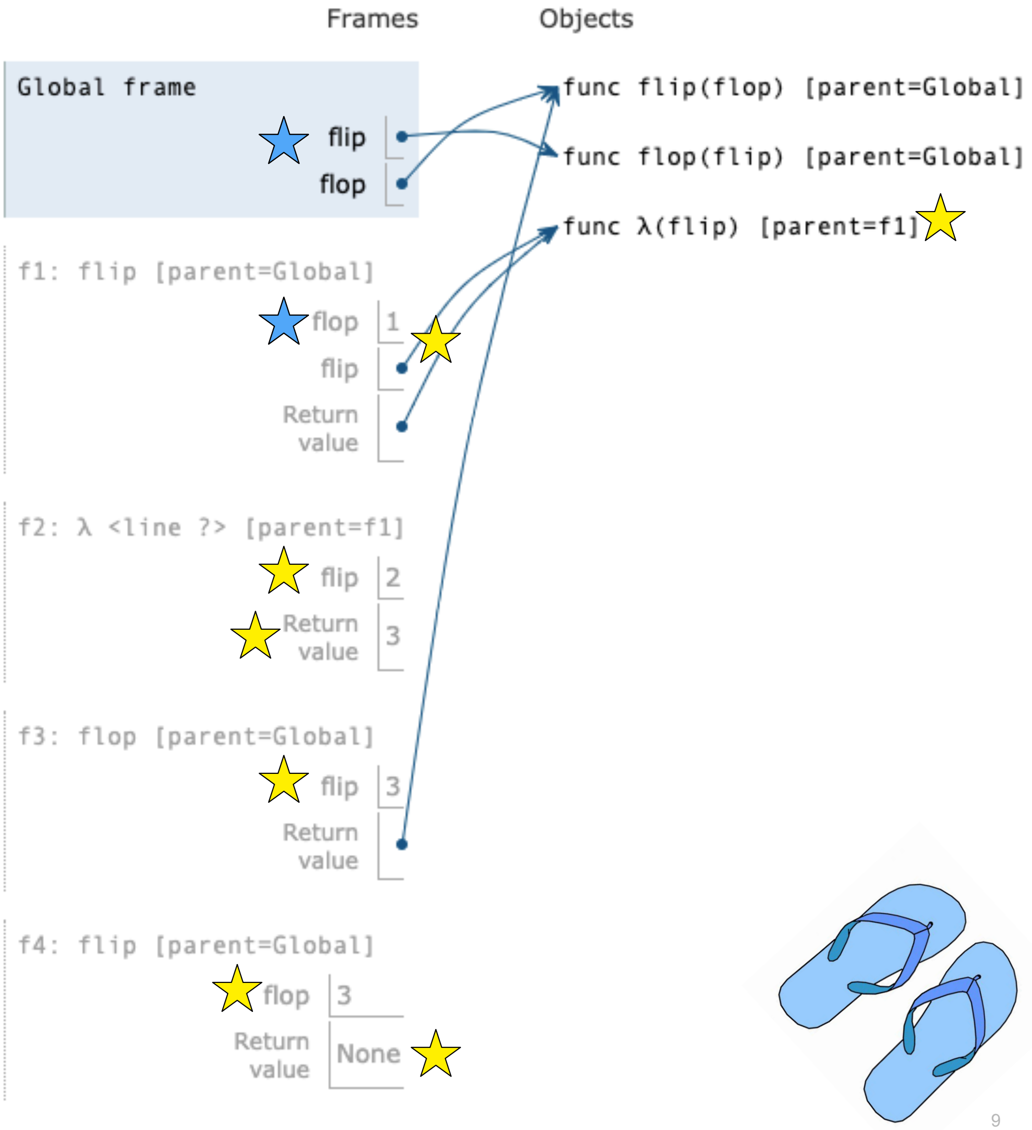
f4: flip [parent=Global]

flop 3

Return value None

A Day at the Beach

```
def flip(flop):
    if ______:          not true for flop == 1
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip
______

flip(____)(3)
     flop(1)
```



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value

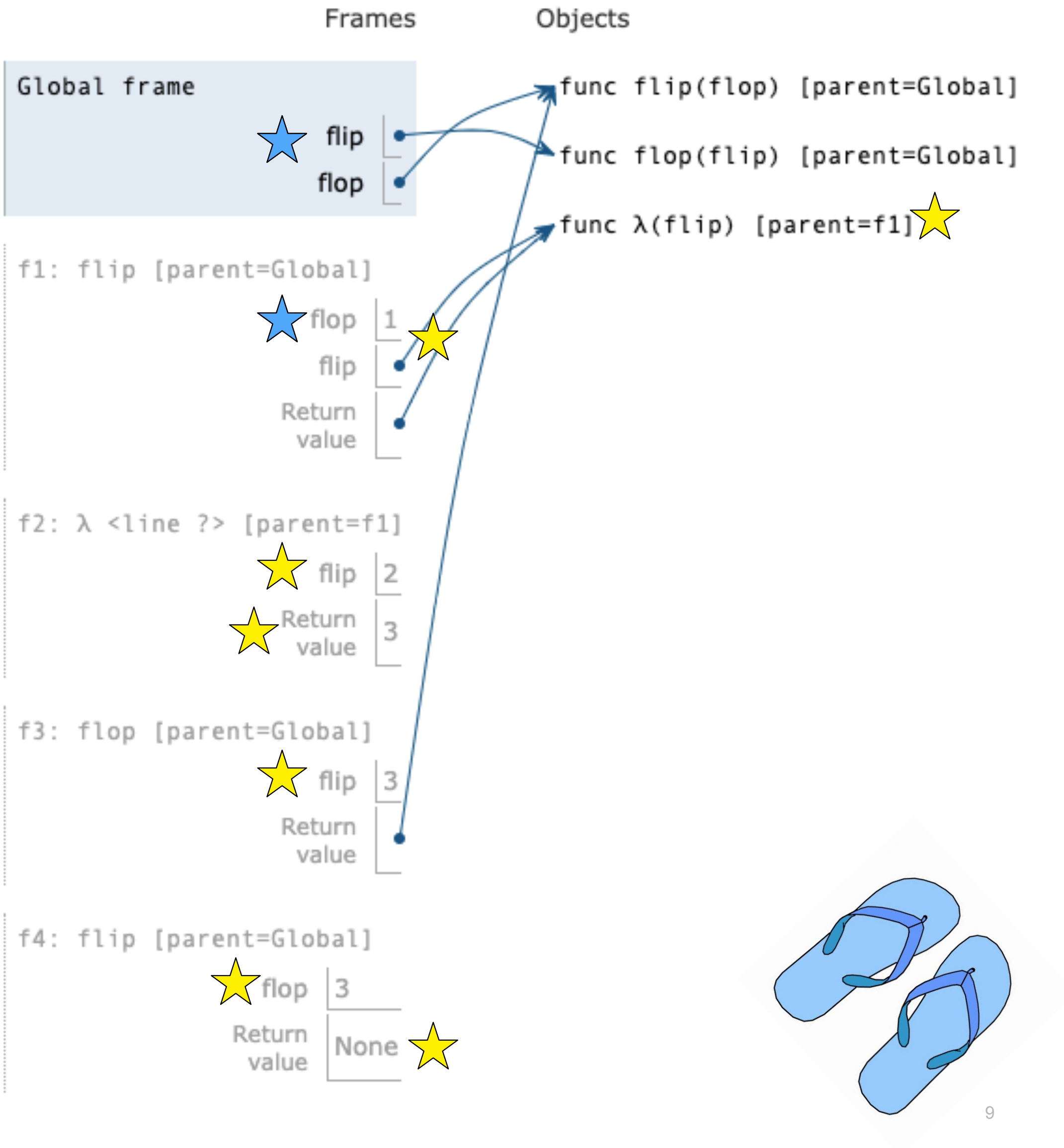
f4: flip [parent=Global]

flop 3

Return value None

A Day at the Beach

```
def flip(flop):
    if ______:         not true for flop == 1
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
      flop(1)
```



Frames

Objects

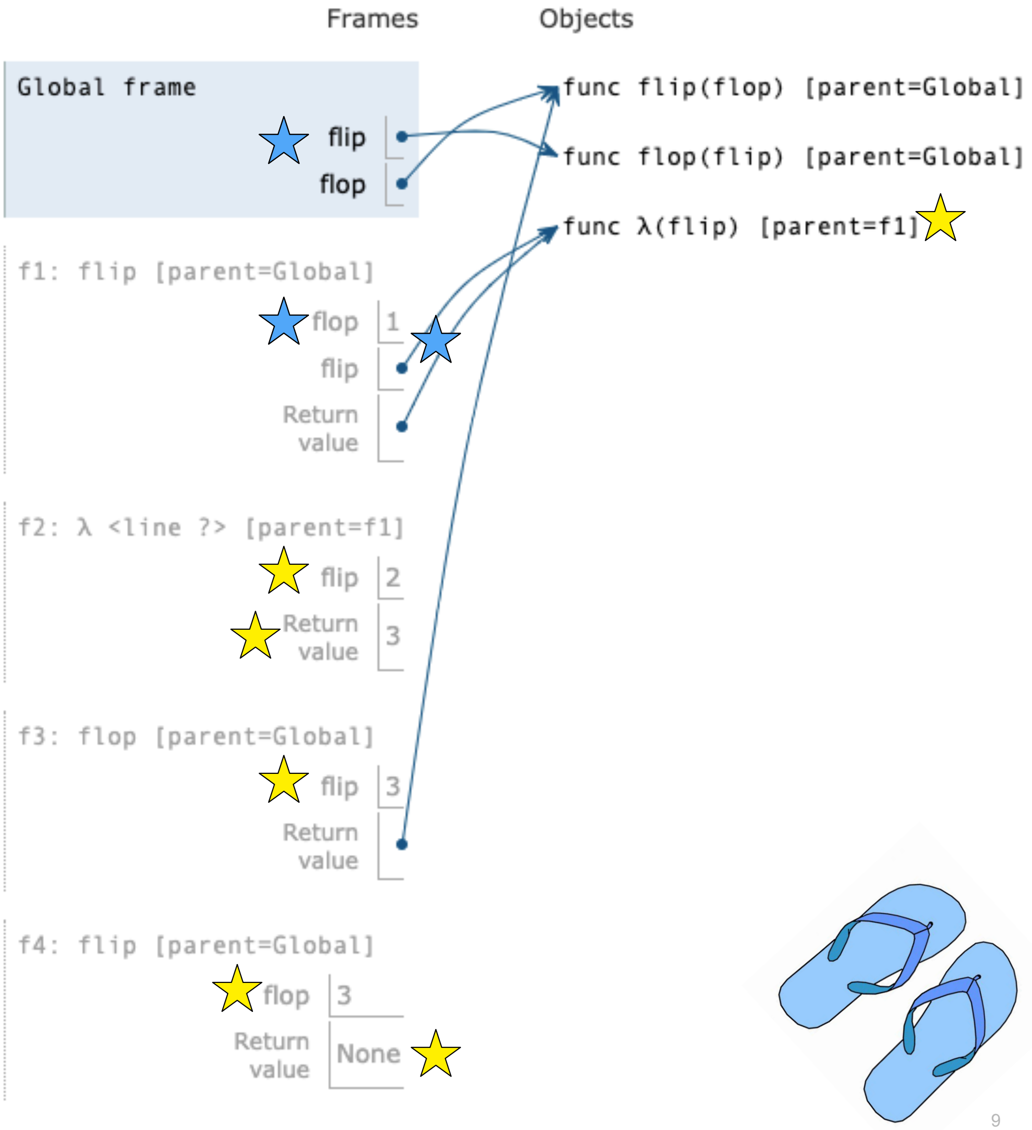
```
Global frame
        flip  •   func flip(flop) [parent=Global]
        flop  •   func flop(flip) [parent=Global]
                  func λ(flip) [parent=f1]

f1: flip [parent=Global]
        flop  1
        flip  •
      Return  •
       value

f2: λ <line ?> [parent=f1]
        flip  2
      Return  3
       value

f3: flop [parent=Global]
        flip  3
      Return  •
       value

f4: flip [parent=Global]
        flop  3
      Return  None
       value
```

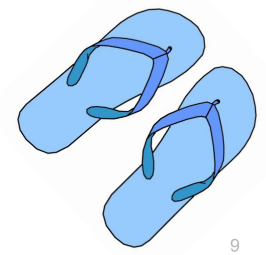
A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
```

not true for flop == 1

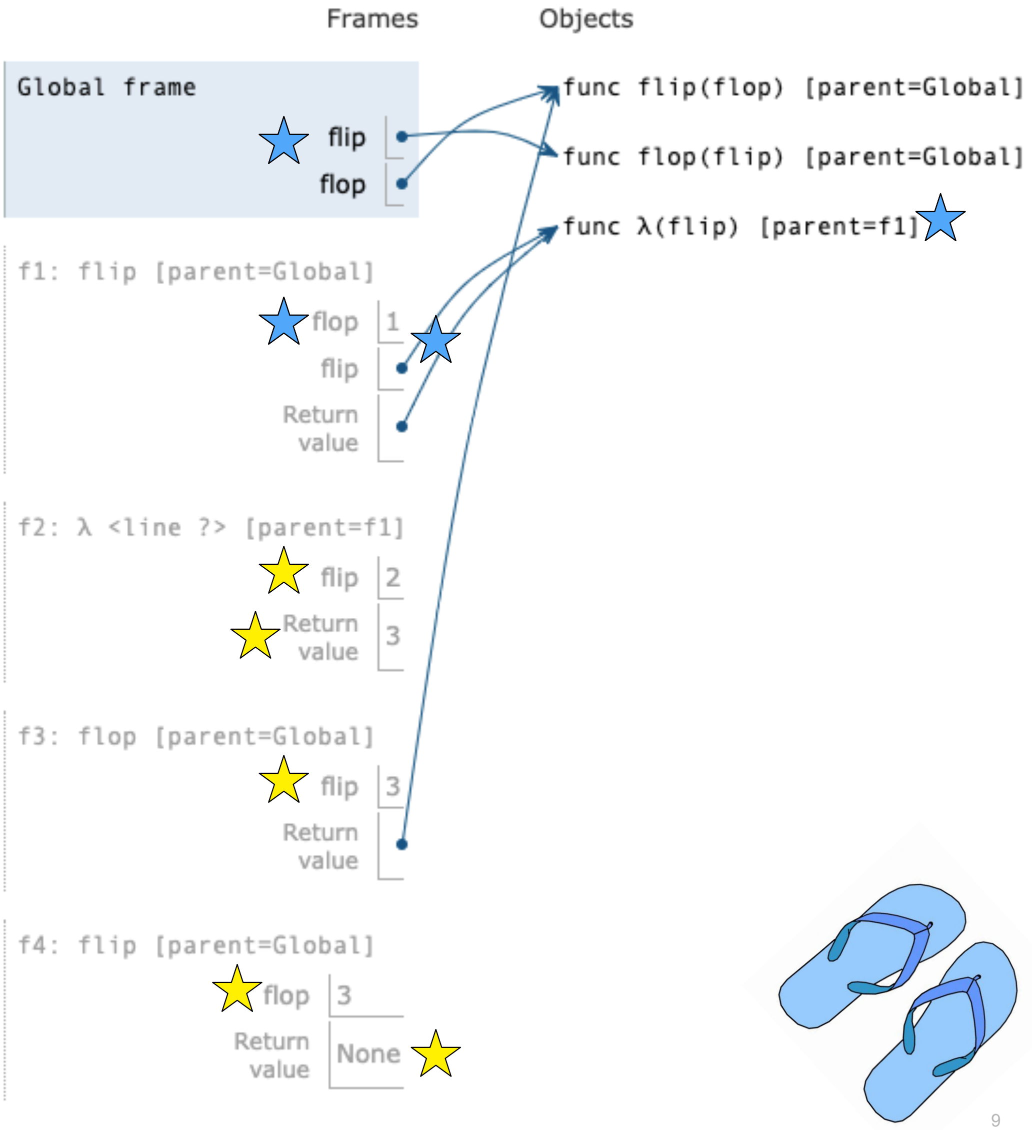
flop(1)





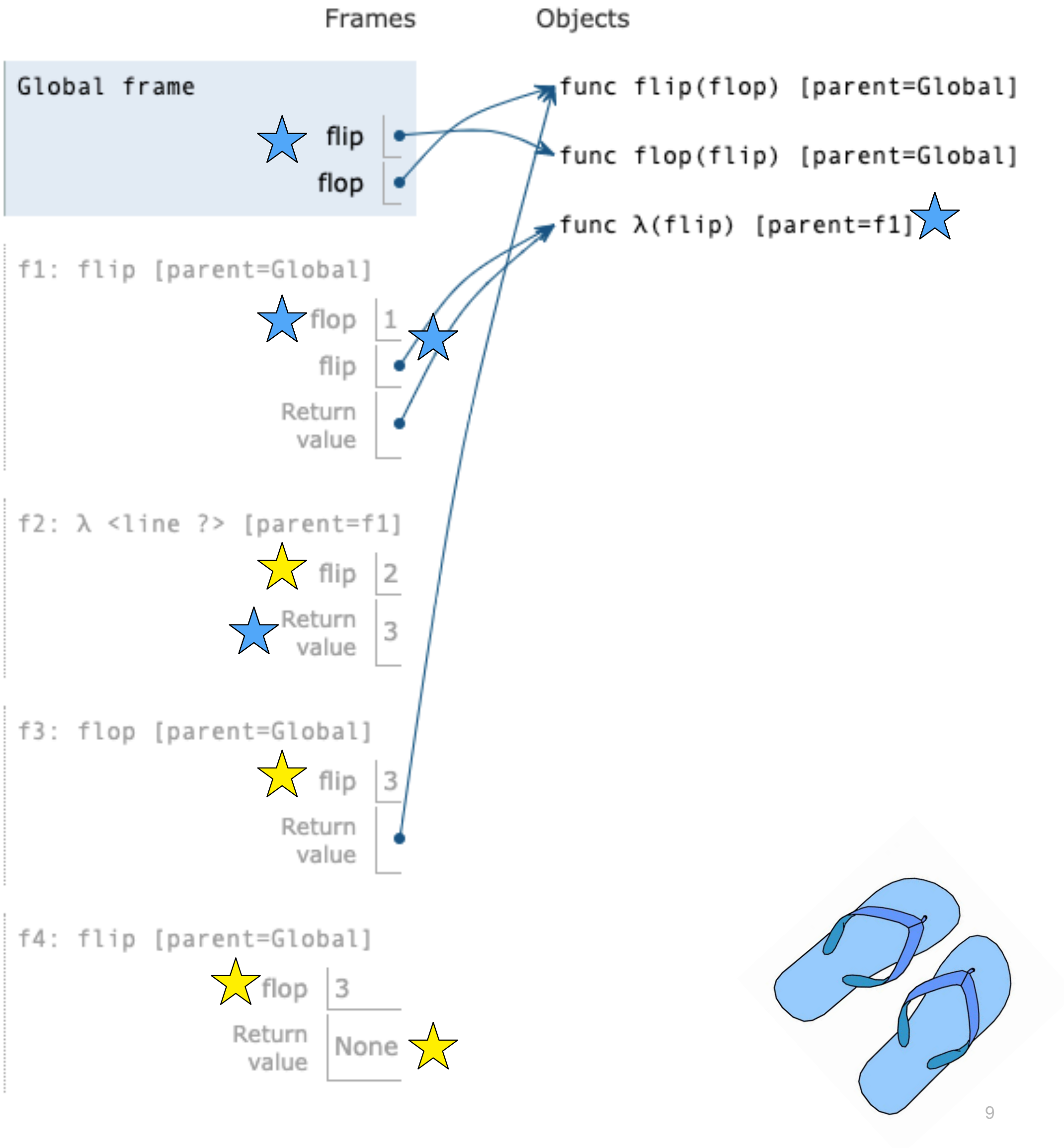
A Day at the Beach

```
def flip(flop):
    if ______:          not true for flop == 1
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
      flop(1)
```



A Day at the Beach

```
def flip(flop):
    if ______:             not true for flop == 1
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
       flop(1)
```

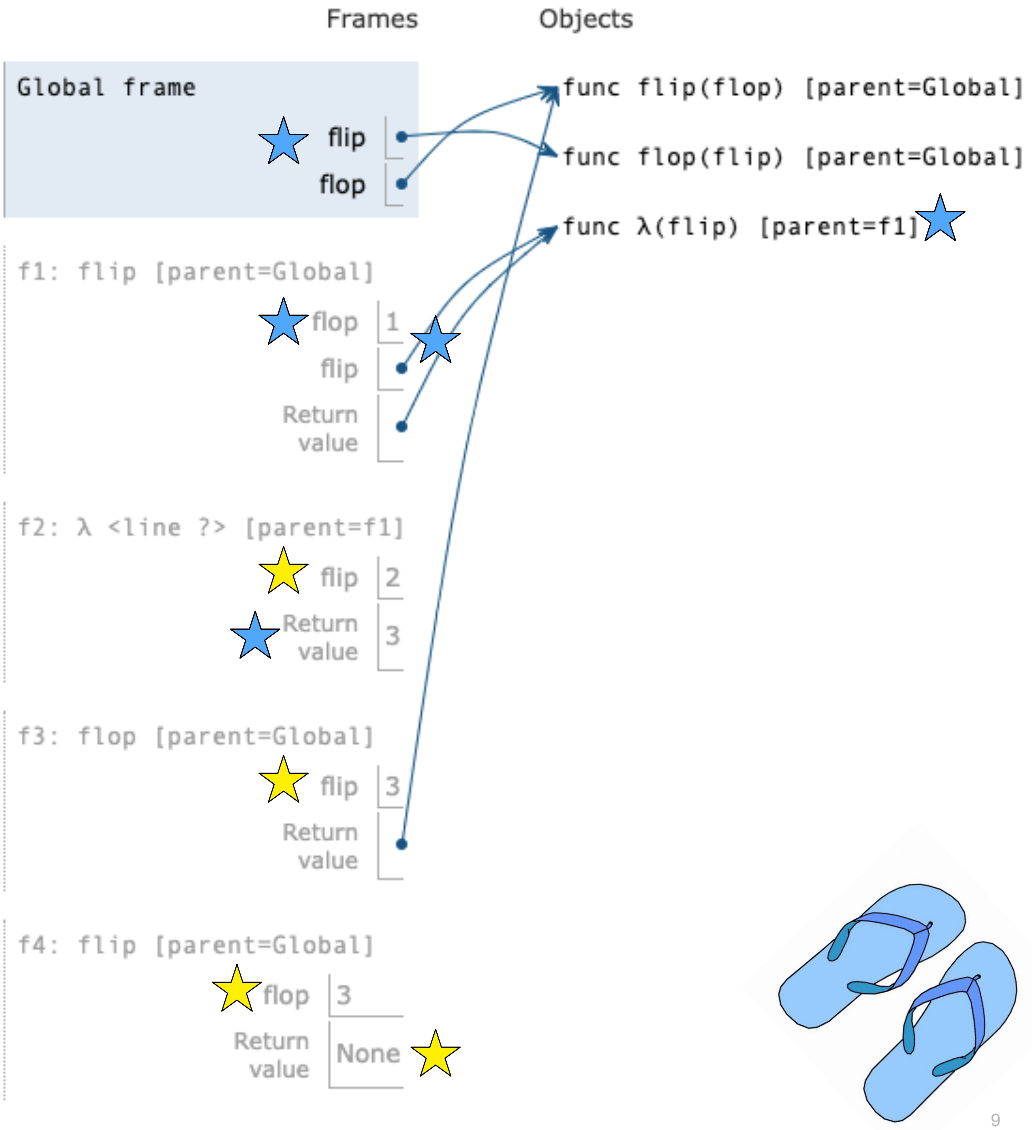


A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
```

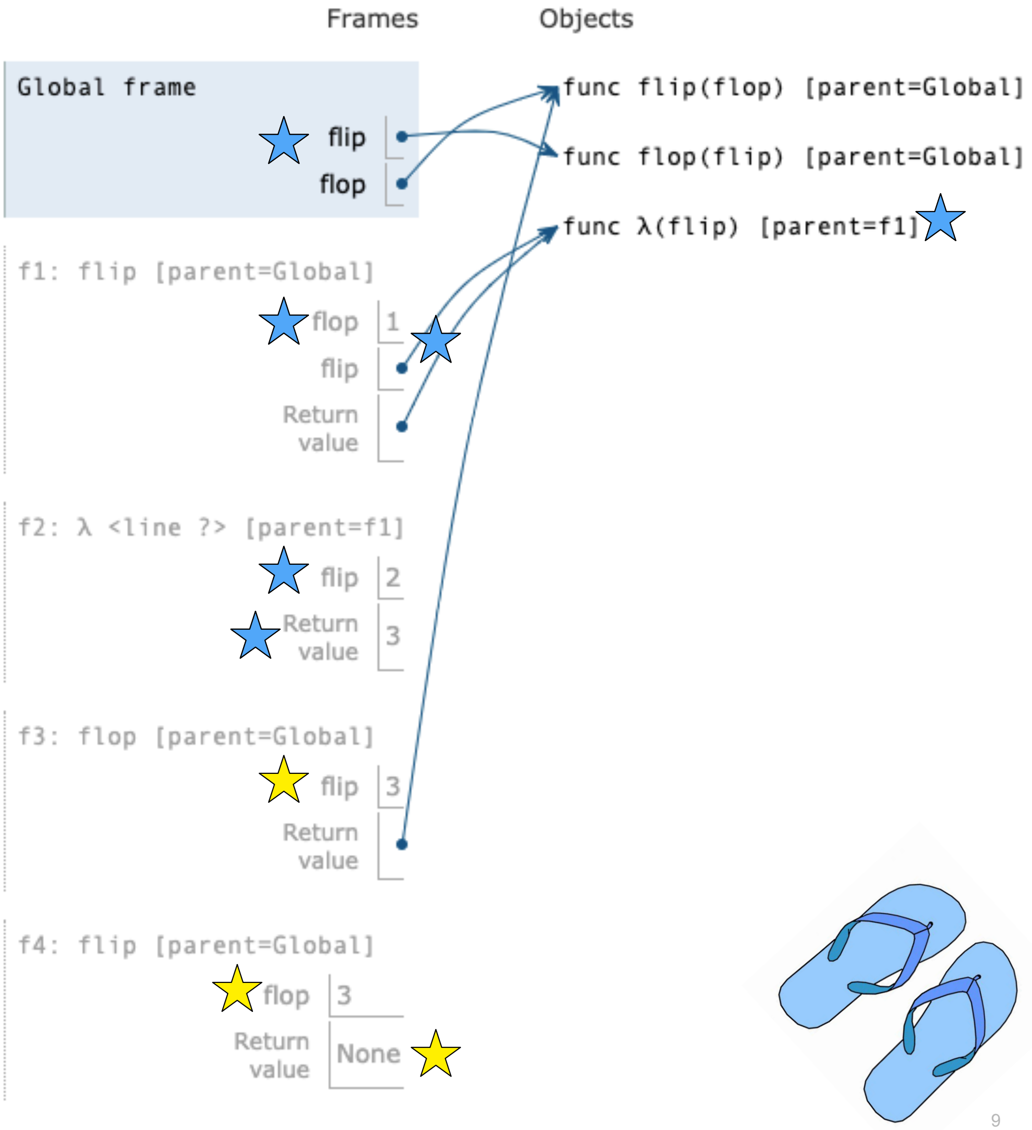
not true for flop == 1

flop(1)(2)



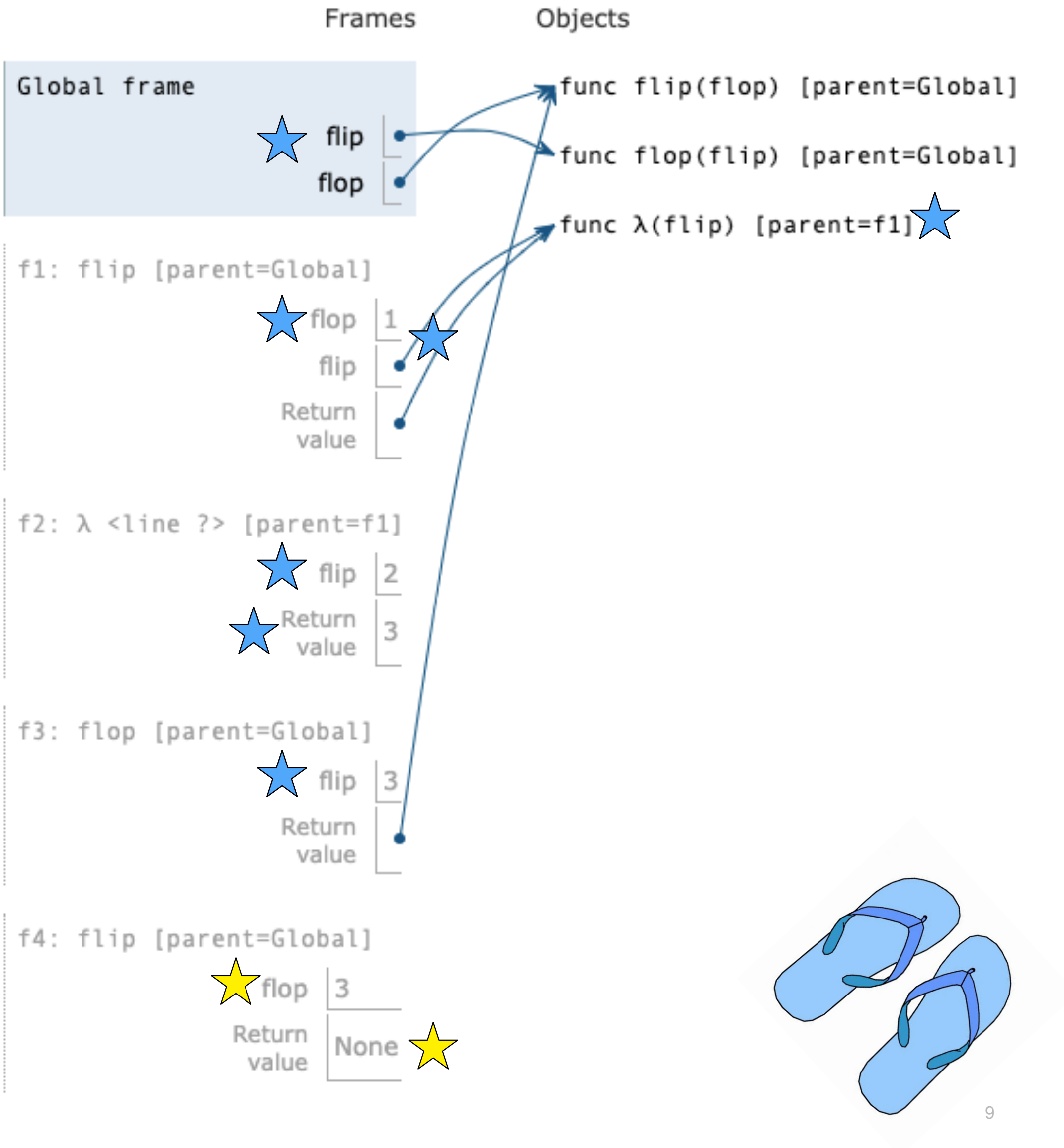
A Day at the Beach

```
def flip(flop):
    if ______:          not true for flop == 1
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
     flop(1)(2)
```



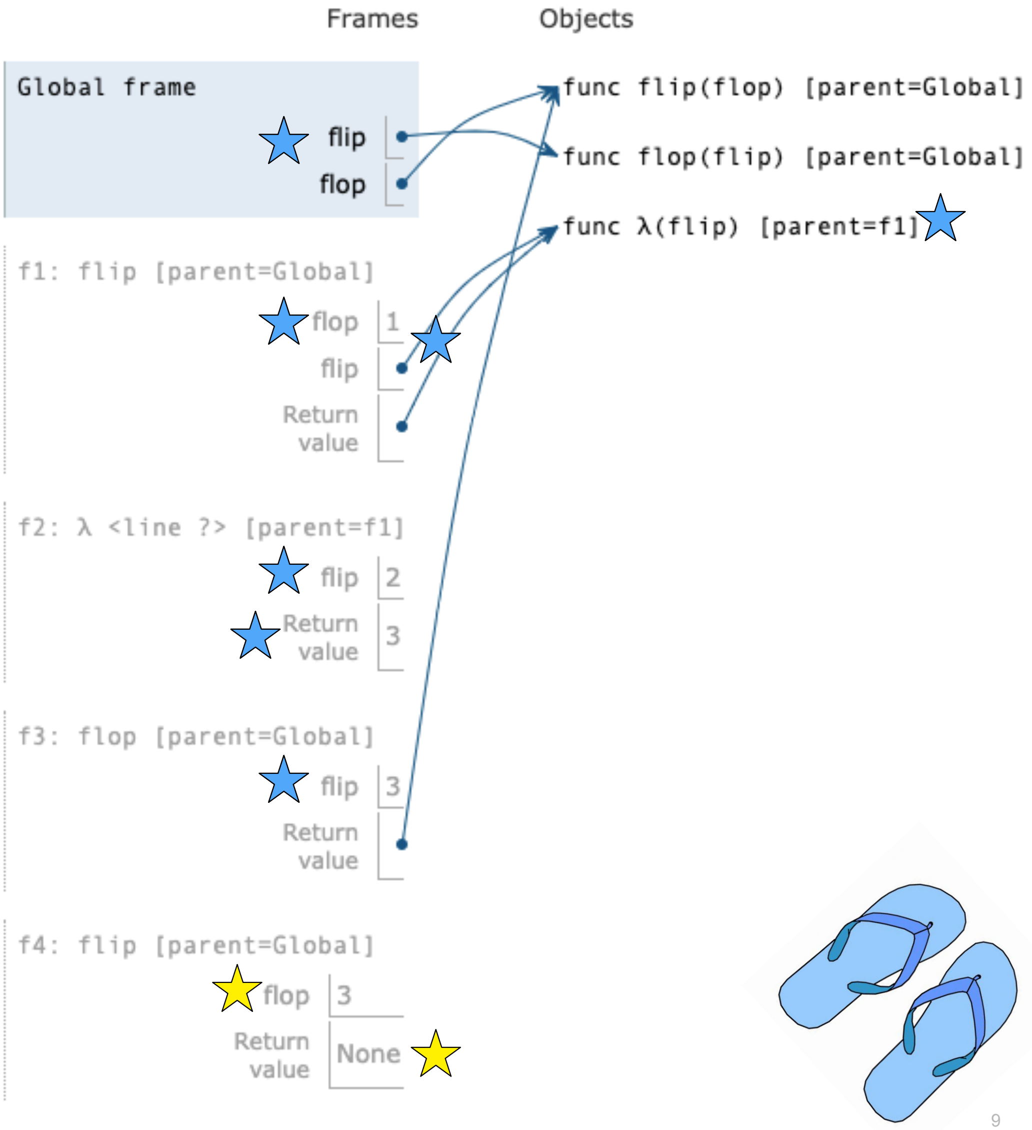
A Day at the Beach

```
def flip(flop):
    if ______:        not true for flop == 1
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
     flop(1)(2)
```



A Day at the Beach

```
def flip(flop):
    if ______:          not true for flop == 1

        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)

        flop(1)(2)
```



Frames

Objects

Global frame
flip
flop

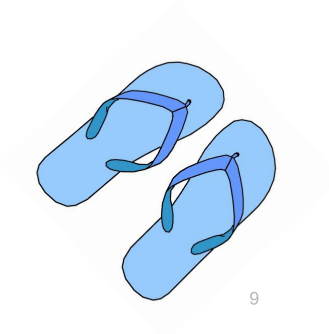
func flip(flop) [parent=Global]
func flop(flip) [parent=Global]
func λ(flip) [parent=f1]

f1: flip [parent=Global]
flop 1
flip
Return value

f2: λ <line ?> [parent=f1]
flip 2
Return value 3

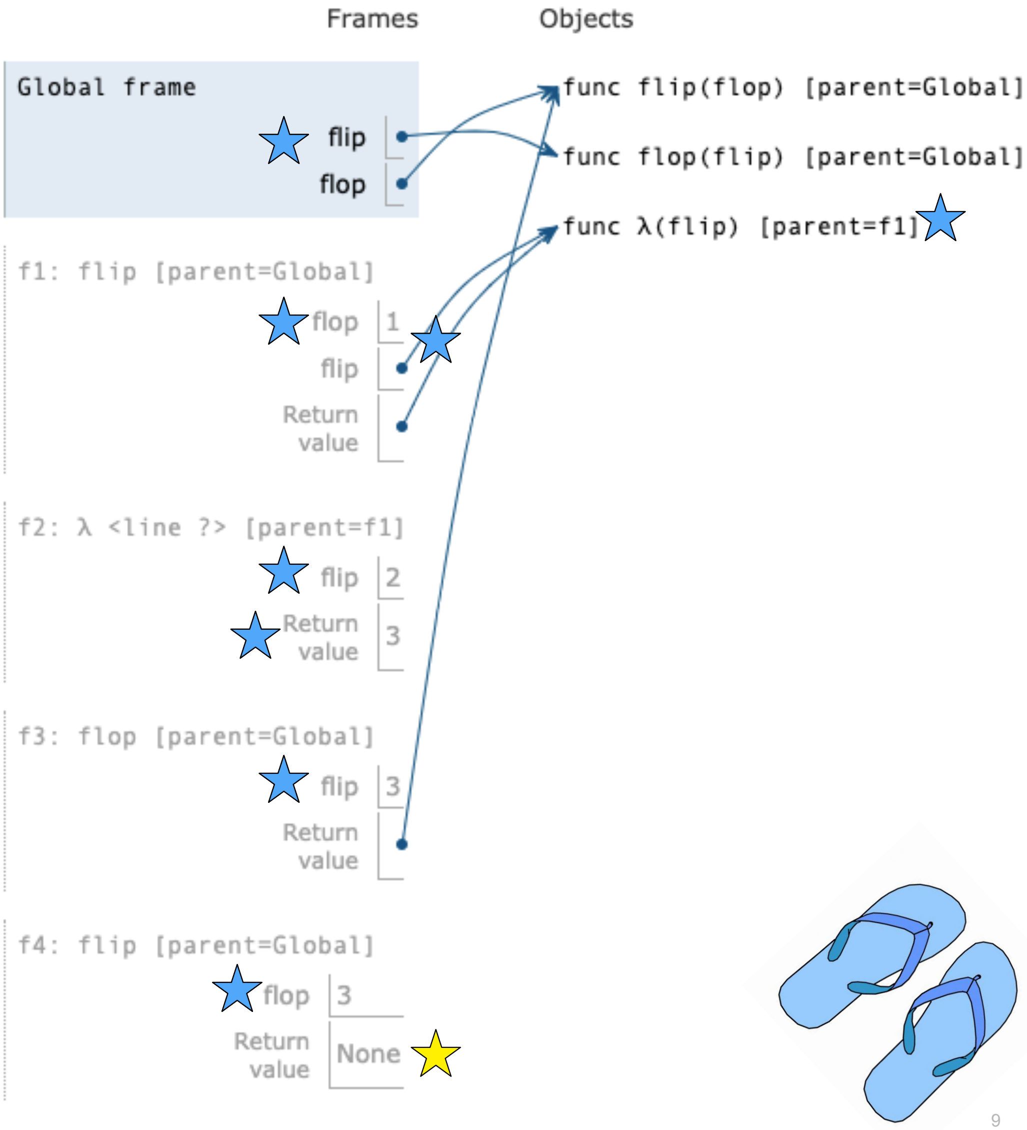
f3: flop [parent=Global]
flip 3
Return value

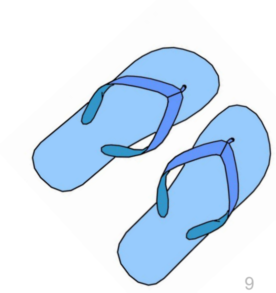
f4: flip [parent=Global]
flop 3
Return value None



A Day at the Beach

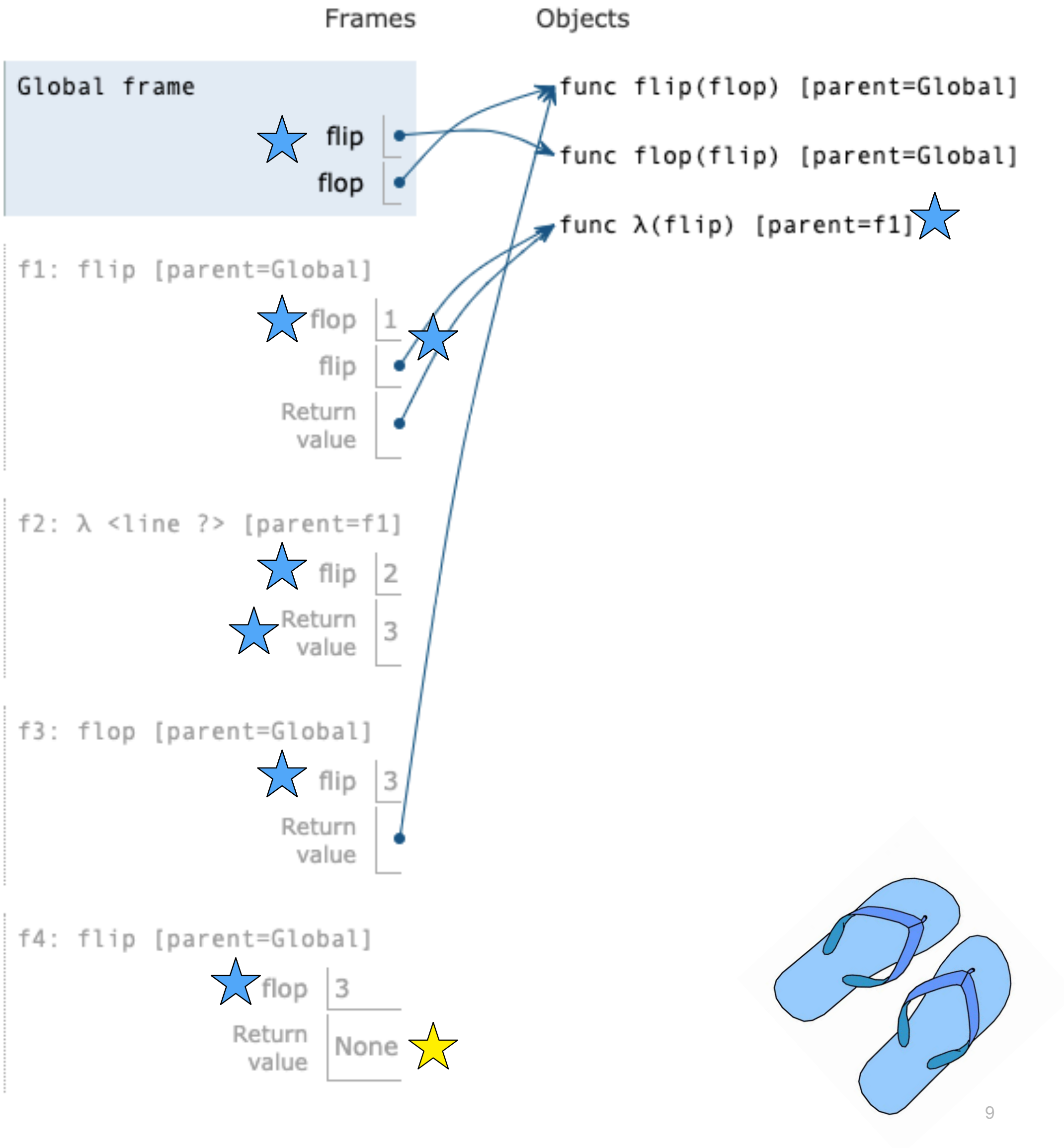
```
def flip(flop):
    if ______:            not true for flop == 1
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
     flop(1)(2)
```


Frames
Objects
Global frame
flip
flop
func flip(flop) [parent=Global]
func flop(flip) [parent=Global]
func λ(flip) [parent=f1]
f1: flip [parent=Global]
flop 1
flip
Return value
f2: λ <line ?> [parent=f1]
flip 2
Return value 3
f3: flop [parent=Global]
flip 3
Return value
f4: flip [parent=Global]
flop 3
Return value None



A Day at the Beach

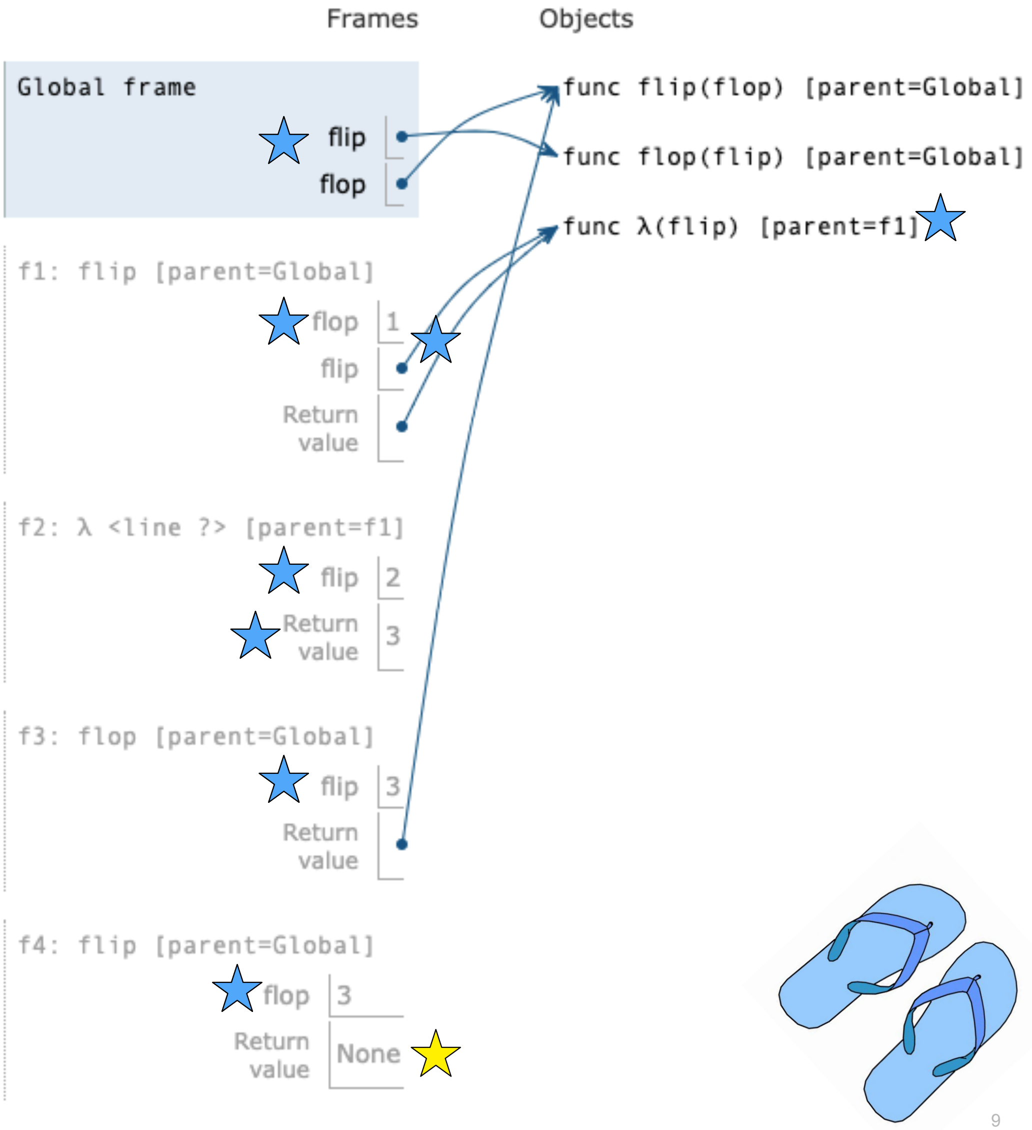
```
def flip(flop):
    if ______:        not true for flop == 1
                          true for flop == 3
       ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
     flop(1)(2)
```



A Day at the Beach

```
def flip(flop):
    if flop>2:        not true for flop == 1
                          true for flop == 3
        ______
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
```

flop(1)(2)



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value

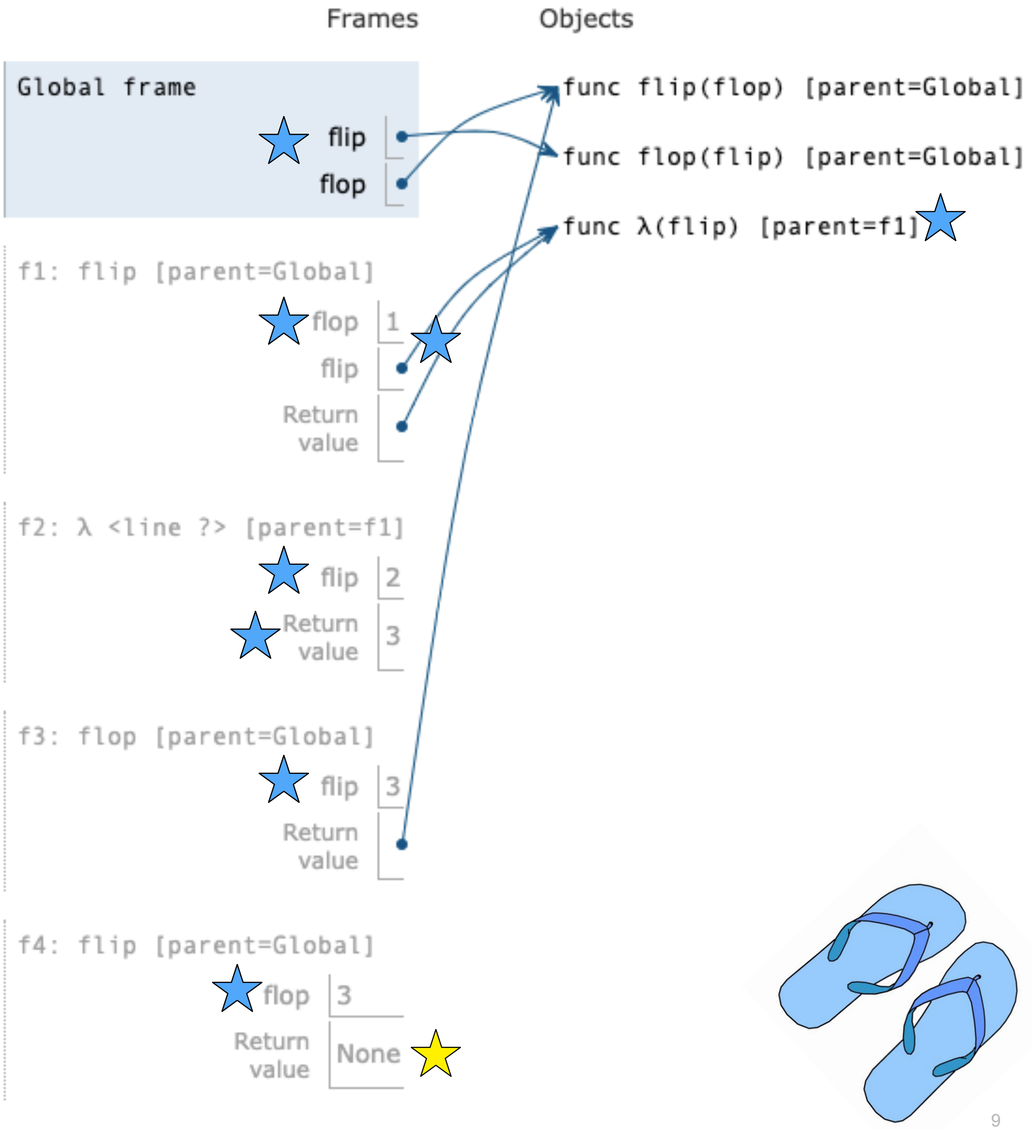
f4: flip [parent=Global]

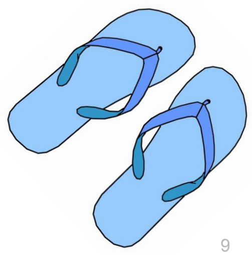
flop 3

Return value None

A Day at the Beach

```
def flip(flop):
    if flop>2:          <- not true for flop == 1
                           true for flop == 3
        return None
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
      ^
      flop(1)(2)
```





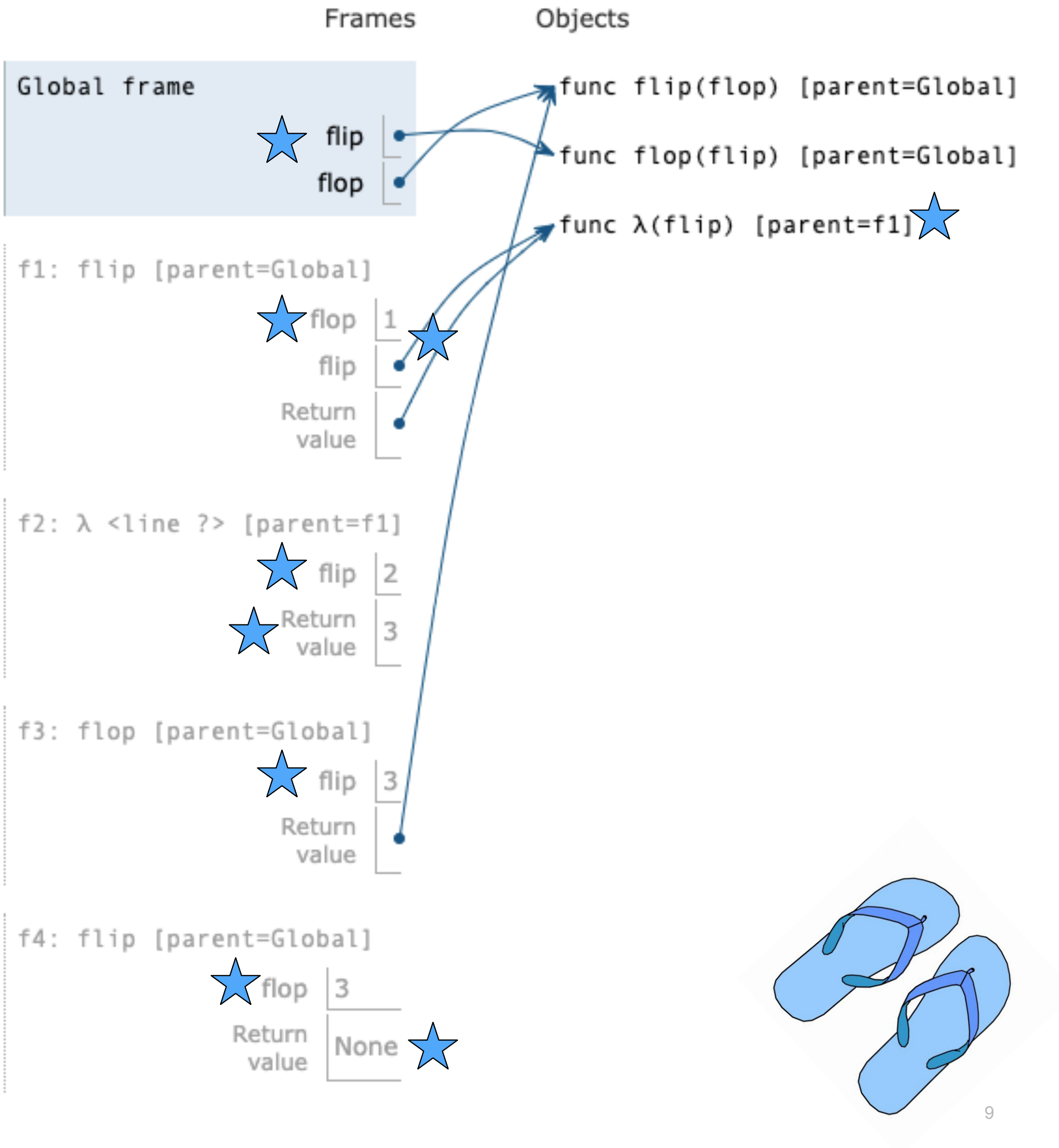
A Day at the Beach

```
def flip(flop):
    if flop>2:
        return None
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
```

not true for flop == 1

true for flop == 3

flop(1)(2)



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line ?> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value

f4: flip [parent=Global]

flop 3

Return value None

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

231

3

```
def remove(n, digit):
    """Return all digits of non-negative N
          are        IT, 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):
    """Retu n all digits of non-negative N
       that are     IT, for some
       non-nega     IT 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 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 _______________________________:

            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):
    """Retu n all  digits of non-negative N
         are       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 = ____________________

            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):
    """Retu n al  digits of non-negative N
       are    IT, for some
         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 + 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):
    """Retu n all digits of non-negative N
    231 are 3 IT, 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

    21      digits = ____________

    return _______kept_______
```

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 digits of non-negative N
           are      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 = ______________

    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):
    """Retu  n all  digits of non-negative N
             are       IT, for some
       non-nega      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

1

+ 20

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 = ______digits + 1______

    return ______kept______
```

231

3

21

$$\begin{array}{r} 1 \\ +\ 20 \\ \hline 21 \end{array}$$

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

3

21

1

+ 20

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):
    """Retu  n all  digits of non-negative N
           are        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

21

```
   1
+ 20
----
  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

            digits = ______digits + 1______

    return ______kept______
```

231

4

231

  1
+ 20
―――
 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

            digits = ____digits + 1____

    return ________kept________
```

231

4

231

```
   1         1
+ 20      + 30
         + 200
----      -----
  21       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 digits of non-negative N
           are       IT, for some
      n    ega       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 + 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):
    """Retu n all 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 = ____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):
    """Retu    n all    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 = ____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):
    """Retu n all digits of non-negative N
          are        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 = ___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):
    """Retu n all digits of non-negative N
         are       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 = 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

Decorators

Function Decorators

(Demo)

Function Decorators

(Demo)

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

Function Decorators

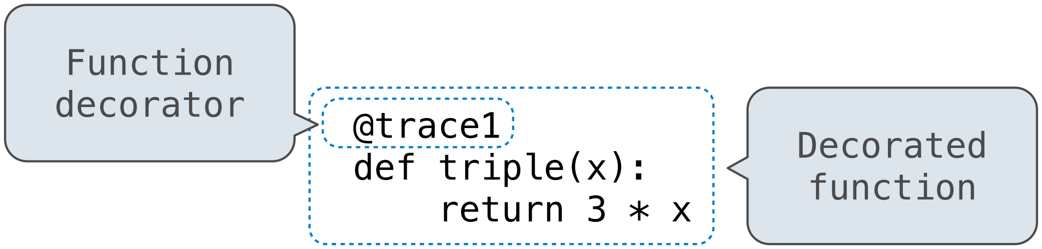
(Demo)

Function decorator

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

Function Decorators

(Demo)



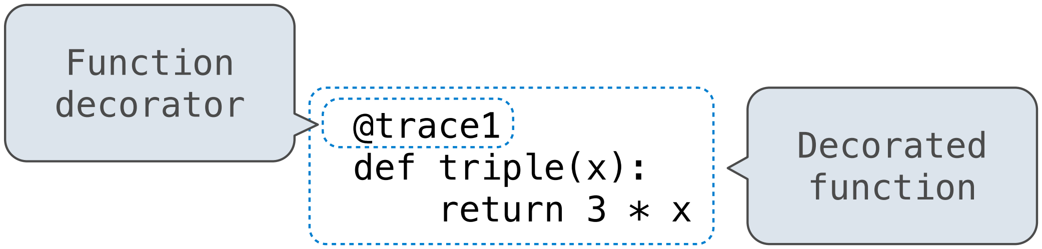
Function decorator

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

Decorated function

Function Decorators

(Demo)



Function decorator

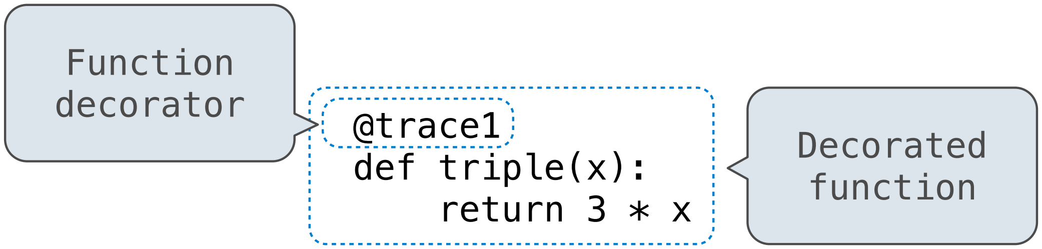
```
@trace1
def triple(x):
    return 3 * x
```

Decorated function

is identical to

Function Decorators

(Demo)



Function decorator

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

Decorated function

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)
```