Function Examples
Announcements
Hog Contest Rules

[link: 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
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
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
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

[link](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

https://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
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
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
Sume 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

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

**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 ______ .
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 ______ .

One approach:
Boolean Favorites

```python
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 _____ .
```

One approach:
1. Read the code
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

mystery1 prints _____ less than n _____.

One approach:
1. Read the code
2. Read the description options
3. Consider an example
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 _____ .

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

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

def mystery1(n):
    likes = is_prime
    n = 8
    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.
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""
    ...

def mystery1(n):
    k = 1
    n = 8
    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

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

def mystery1(n):
    likes = is_prime
    k = 1
    n = 8
    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

all odd numbers

_mystery1_ prints _____ less than n ______ .

_mystery1_ prints all odd numbers less than n that George likes.
def likes(n):
    """Returns whether George Boole likes the non-negative integer n."""

    def mystery1(n):
        likes = is_prime
        n = 8
        k = 1
        while k < n:
            if likes(n):
                print(k)
            k = k + 2

        mystery1 prints all odd numbers but only if George likes n

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

```python
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

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

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

mystery 2 returns _____ or returns None if _____.

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

    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

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

def flop(flip):
    return flop

______

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

def flop(flip):
    return flop

_____

flip(____)(3)
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______
flip(____)(3)
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______
flip(____)(3)
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)
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)
A Day at the Beach

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

def flop(flip):
    return flop

_____

flip(____)(3)
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
def flip(flop):
    if _____:
        _____
    flip = _____
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
flop(1)
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)
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)
A Day at the Beach

```python
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)
```

Lambda flip: 3
A Day at the Beach

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

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)

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

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
A Day at the Beach

def flip(flop):
    if _____:  # not true for flop == 1
        _____
        flip = _____ flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)

flop(____)(3)  # flop(1)(2)
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

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

def flop(flip):
    return flop

flip, flop = flop, flip

flip(__)(3)
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 = ______
    return flip

def flop(flip):
    return flop

______
flip, flop = flop, flip

flip(____)(3)

flop(1)(2)
A Day at the Beach

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

def flop(flip):
    return flop

flip, flop = flop, flip

flip(____)(3)
```

not true for flop == 1
true for flop == 3

lambda flip: 3

true for flop == 3
A Day at the Beach

def flip(flop):
    if flop > 2:
        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)

flop(____)(3)

flop(1)(2)
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)

flop(1)(2)
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 ______________________________
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

```python
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

```python
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

```python
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 _______________________________
Implementing a Function

```python
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

```python
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

```python
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 = last
            digits = digits * 10 + kept
    return digits
```

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

```python
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 _______________________________
```

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

```python
def remove(n, digit):
    """Return all digits of non-negative N except DIGIT, for some non-negative DIGIT less than 10.""

keep, digits = 0, 0

while n > 0:
    n, last = n // 10, n % 10
    if last != digit:
        keep = ...  # Annotate names with values from your chosen example
    digits = ...  # Write code to compute the result

return ...  # Did you really return the right thing?
```

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

```python
def remove(n, digit):
    """Return all digits of non-negative N except for some non-negative DIGIT less than 10."
    kept, digits = 0, 0
    while n > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = kept + last
    digits = kept
    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):
    """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 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 = kept

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

```python
def remove(n, digit):
    """Return all digits of non-negative N except 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*10
        digits = kept
    return digits
```

1. Read the description
2. Verify the examples & pick a simple one
3. Read the template
4. Implement without the template, then change your implementation to match the template. **OR**
5. If the template is helpful, use it.
6. Annotate names with values from your chosen example
7. Write code to compute the result
8. Did you really return the right thing?
9. 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 = kept
    return digits

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
    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

```python
def remove(n, digit):
    """Return all digits of non-negative N excluding those equal to a non-negative DIGIT less than 10."
    kept, digits = 0, 0
    n, last = n // 10, n % 10
    if last != digit:
        kept = kept + last
    digits = digits + 1
    return kept

>>> remove(231, 3)
21
>>> remove(243132, 2)
4313
```

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

```python
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 = digits + 1
    return kept * 10**digits
```

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 = digits + 1
    return kept * 10**digits

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
    # except for any 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*10**digits
    digits = digits + 1

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

```python
def remove(n, digit):
    """Return all digits of non-negative N
    except for some 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 = kept
    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

```python
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 * 10 + kept
return digits
```

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 ________________________________:
            _______________________
        _______________________
    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

```python
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
```

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

```python
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 = _____________________
    kept = _______________________
    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

```python
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))
```

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)

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

(Demo)

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

Function decorator

Decorated function
Function Decorators

(Demo)

Function decorator

@trace

def triple(x):
    return 3 * x

Decorated function

is identical to
Function Decorators

(Demo)

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

is identical to

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

(Demo)

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

is identical to

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

Why not just use this?