Function Examples
Announcements
Hog Contest Rules

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

Fall 2011 Winners
Keegan Mann
Yan Duan & Ziming Li
Brian Prike & Zhenghao Qian
Parker Schuh & Robert Chatham

Fall 2012 Winners
Chenyang Yuan
Joseph Hui

Fall 2013 Winners
Paul Bramsen
Sam Kumar & Kangsik Lee
Kevin Chen

Fall 2014 Winners
Alan Tong & Elaine Zhao
Zhenyang Zhang
Adam Robert Villaflor & Joany Gao
Zhen Qin & Dian Chen
Zizheng Tai & Yihe Li

[cs61a.org/proj/hog_contest]
Hog Contest Winners

**Spring 2015 Winners**
Sinho Chewi & Alexander Nguyen Tran
Zhaoxi Li
Stella Tao and Yao Ge

**Fall 2015 Winners**
Micah Carroll & Vasilis Oikonomou
Matthew Wu
Anthony Yeung and Alexander Dai

**Spring 2016 Winners**
Michael McDonald and Tianrui Chen
Andrei Kassiantchouk
Benjamin Krieges

**Fall 2016 Winners**
Cindy Jin and Sunjoon Lee
Anny Patino and Christian Vasquez
Asana Choudhury and Jenna Wen
Michelle Lee and Nicholas Chew

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

**Spring 2018 Winners**
Eric James Michaud
Ziyu Dong
Xuhui Zhou

**Fall 2018 Winners**
Rahul Arya
Jonathan Bodine
Sumer Kohli and Neelesh Ramachandran

**Fall 2019 Winners**
Currying
Function Currying

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

```python
>>> make_adder(2)(3)
5
```

There's a general relationship between these functions (Demo)

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

(Demo)

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

is identical to

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

Why not just use this?
Review
What Would Python Display?

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

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

```
from operator import add, mul

def square(x):
    return mul(x, x)
```

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

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

```
Names in nested def statements can refer to their enclosing scope
```

```
This expression           Evaluates to
5                           5
print(5)                   None
print(print(5))            None
(\textit{delay(\textit{delay})})(6)()           6
print(\textit{delay(print)})(4))                  None
```

```
Interactive Output
5
5
5
6
delayed delayed delayed delayed
4
None
```
What Would Python Print?

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

```python
from operator import add, mul

def square(x):
    return mul(x, x)

def pirate(arggg):
    print('matey')

def plunder(arggg):
    return arggg
    return plunder
```

A function that always returns the identity function

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(pirate(3)(square)(4), 1)</td>
<td>17</td>
<td>Matey</td>
</tr>
<tr>
<td>func square(x)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pirate(pirate(pirate))(5)(7)</td>
<td>Error</td>
<td>Matey</td>
</tr>
<tr>
<td>Identity function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Error</td>
</tr>
</tbody>
</table>

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
```python
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse

def mask(horse):
    return horse(2)
```

Diagram:
- **Global Frame**:
  - `horse`
  - `mask`

- **f1: horse [parent=Global]**:
  - Return Value: 2

- **f2: λ [parent=Global]**:
  - Return Value: 2

- **f3: mask [parent=f1]**:
  - horse: 2
  - Return Value: 2
Implementing Functions
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/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