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
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
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
Hog Contest Rules

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

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

cs61a.org/proj/hog_contest
Hog Contest Rules

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

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

Fall 2012 Winners
Chenyang Yuan
Joseph Hui

cs61a.org/proj/hog_contest
Hog Contest Rules

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

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

Fall 2012 Winners
Chenyang Yuan
Joseph Hui

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

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

>>> make_adder(2)(3)
5
>>> add(2, 3)
5
Function Currying

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

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

There's a general relationship between these functions!
Function Currying

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

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

There's a general relationship between these functions

(Demo)
Function Currying

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

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

There's a general relationship between these functions

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

(Demo)
Function Decorators

(Demo)

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

(Demo)

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

(Demo)

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

Diagram:

- **Function decorator**
- **@trace1**
- **def triple(x):**
  - **return 3 * x**
- **Decorated function**
Function Decorators

(Demo)

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

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?
Review
What Would Python Display?
What Would Python Display?

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

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

```python
from operator import add, mul

def square(x):
    return mul(x, x)
```
What Would Python Display?

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

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>from operator import add, mul</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>def square(x): return mul(x, x)</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What Would Python Display?

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)
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

10
What Would Python Display?

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

```
from operator import add, mul

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
What Would Python Display?

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)
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

print(5)
What Would Python Display?

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)

print(5)

This expression       | Evaluates to | Interactive Output
----------------------|--------------|-------------------
5                     | 5            | 5                 
print(5)              | None         | 5                 
```
What Would Python Display?

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

```python
from operator import add, mul

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
</tbody>
</table>

10
What Would Python Display?

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)
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

This expression evaluates to 5, and the interactive output is also 5.
What Would Python Display?

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)

print(5)  # None
5
print(5)  # None 5
5
print(print(5))  # None
None
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
What Would Python Display?

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)

print(print(5))
None

print(5)
5

print(print(5))
None
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5</td>
</tr>
</tbody>
</table>
What Would Python Display?

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)

print(print(5))
None

print(5)
None

print(print(5))
None

print(print(5))
None

print(print(5))
None
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
</tbody>
</table>
What Would Python Display?

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

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
</tbody>
</table>

```
def delay(arg):
    print('delayed')
def g():
    return arg
return g
```
What Would Python Display?

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)
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
</tbody>
</table>

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

Names in nested def statements can refer to their enclosing scope.
What Would Python Display?

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

```
from operator import add, mul

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

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

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

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
</tbody>
</table>
What Would Python Display?

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

```
from operator import add, mul

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

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

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
<tr>
<td>delay(delay)()(6)()</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
What Would Python Display?

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

```
from operator import add, mul

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

```

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

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

def g():
    return arg

```

Names in nested def statements can refer to their enclosing scope

```
print(print(5))

```

None

```
delay(delay)()(6)()(

```

None

This expression | Evaluates to | Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5
print(print(5)) | None | 5
None | None | 5
```
What Would Python Display?

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

```
from operator import add, mul

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

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

def g():
    return arg

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>delay(delay)()(6)()</td>
<td>None</td>
<td>5 None</td>
</tr>
</tbody>
</table>
What Would Python Display?

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

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

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

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

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

This expression | Evaluates to | Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5
print(print(5)) | None | None
delay(delay)()(6)() | 5 | 5

Interactive Output

None

Names in nested def statements can refer to their enclosing scope.
What Would Python Display?

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

```
from operator import add, mul

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

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

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

def g():
    return arg

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>delay(delay)()</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>delay(delay)()(6)(6)</td>
<td>None</td>
<td>5</td>
</tr>
</tbody>
</table>
What Would Python Display?

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)

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

def g():
    return arg

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
<tr>
<td>delay(delay)()</td>
<td>delayed</td>
<td></td>
</tr>
</tbody>
</table>
What Would Python Display?

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

```
from operator import add, mul

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

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

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

Names in nested def statements can refer to their enclosing scope

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
<tr>
<td>delay(delay)()()</td>
<td>delayed delayed</td>
<td></td>
</tr>
</tbody>
</table>
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

def g():
    return g

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

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

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

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>(delay(delay))()()()</td>
<td>None</td>
<td>delayed delayed delayed</td>
</tr>
</tbody>
</table>

5

6
What Would Python Display?

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)
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

def g():
    return arg
return g
```

### This expression | Evaluates to | Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5 None
print(print(5)) | None | 5 None
delay(delay)()() | 6 | delayed delayed 6
What Would Python Display?

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

```
from operator import add, mul

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

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

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

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>print(5)</td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
<tr>
<td>delay(delay)()(6)()</td>
<td>6</td>
<td>delayed delayed</td>
</tr>
<tr>
<td>print(delay(print)()())</td>
<td>None</td>
<td>5</td>
</tr>
</tbody>
</table>
What Would Python Display?

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

```
from operator import add, mul

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

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

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

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

```
print(print(5))
```

```
5
print(5)
```

```
print(print(5))
```

```
5
None
```

```
This expression | Evaluates to | Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5 None
delay(print)()()() | 6 | delayed delayed 6
print(delay(print)()()) | | delayed
```

```
This expression | Evaluates to | Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5 None
delay(print)()()() | 6 | delayed delayed 6
print(delay(print)()()) | | delayed
```
What Would Python Display?

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

```
from operator import add, mul

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

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

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

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

```
This expression | Evaluates to | Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5
print(print(5)) | None | 5 None
delay(delay)()(6)() | 6 | delayed 6
delay(print)()(4)) | | delayed 4
```
What Would Python Display?

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

```
from operator import add, mul

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

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

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

Names in nested def statements can refer to their enclosing scope

def g():
    return arg
```

This expression  |  Evaluates to  |  Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5 None
print(print(5)) | None | 5 None
(delay(delay)()(6)()) | 6 | delayed delayed 6
delay(print)()() | 4 | delayed None
What Would Python Display?

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

def g():
    return delay

print(5)
print(print(5))
print(delay(delay)()(6))
print(delay(print)()(4))
```

### This expression | Evaluates to | Interactive Output
--- | --- | ---
5 | 5 | 5
print(5) | None | 5
print(print(5)) | None | 5 None
delay(delay)()(6)() | 6 | delayed
delay(print)()(4) | None | delayed 4 None
What Would Python Print?

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

```python
from operator import add, mul

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
</table>
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
```

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>from operator import add, mul</td>
<td></td>
<td></td>
</tr>
<tr>
<td>def square(x):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return mul(x, x)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>def pirate(arggg):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>print('matey')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>def plunder(arggg):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return arggg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return plunder</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What Would Python Print?

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

```
from operator import add, mul
def square(x):
    return mul(x, x)
def pirate(arggg):
    print('matey')
def plunder(arggg):
    return arggg
    return plunder

add(pirate(3)(square)(4), 1)
```

Interactive Output

```
11
```
What Would Python Print?

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

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

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

add(pirate(3)(square)(4), 1)
```

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

```
from operator import add, mul
def square(x):
    return mul(x, x)
def pirate(arggg):
    print('matey')
def plunder(arggg):
    return arggg
    return plunder
```

A function that always returns the identity function

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

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

<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>11</td>
<td>11</td>
</tr>
</tbody>
</table>
What Would Python Print?

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

```
from operator import add, mul

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

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

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

A function that always returns the identity function

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add(pirate(3)(square)(4), 1)</code></td>
<td>Matey</td>
<td></td>
</tr>
</tbody>
</table>

A function that always returns the identity function:

```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 name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

```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

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

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

```

<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>Matey</td>
<td></td>
</tr>
</tbody>
</table>
```

The print function returns None. It also displays its arguments (separated by spaces) when it is called.
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>func square(x)</td>
<td>Matey</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.
What Would Python Print?

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

```
from operator import add, mul
def square(x):
    return mul(x, x)
def pirate(arggg):
    print('matey')
def plunder(arggg):
    return arggg
return plunder
```

This expression
```
add(pirate(3)(square)(4), 1)
```
Evaluates to
```
func square(x)
```
Interactive Output
Matey

A function that always returns the identity function

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

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

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

A function that always returns the identity function

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

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

<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>func square(x)</td>
<td>16</td>
</tr>
</tbody>
</table>

Matey
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

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

```
from operator import add, mul

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

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

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

<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 17</td>
</tr>
<tr>
<td>func square(x)</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

A function that always returns the identity function.

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

```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

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

<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 17</td>
</tr>
</tbody>
</table>

```
```
What Would Python Print?

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

```
from operator import add, mul

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

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

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

A function that always returns the identity function

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

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

<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 17</td>
</tr>
<tr>
<td>func square(x)</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>pirate(pirate(pirate))(5)(7)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
```

<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 17</td>
</tr>
<tr>
<td><code>func square(x)</code></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>pirate(pirate(pirate))(5)(7)</td>
<td>Identity function</td>
<td></td>
</tr>
</tbody>
</table>

A function that always returns the identity function

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

```
from operator import add, mul

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

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

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

A function that always returns the identity function

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

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

<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 17</td>
</tr>
<tr>
<td>pirate(pirate(pirate))(5)(7)</td>
<td>Identity function</td>
<td>Matey</td>
</tr>
</tbody>
</table>
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

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

<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 17</td>
</tr>
<tr>
<td><code>func square(x)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>16</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>pirate(pirate(pirate)))(5)(7)</code></td>
<td></td>
<td>Matey Matey</td>
</tr>
<tr>
<td>Identity function</td>
<td></td>
<td></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.
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
```

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

A function that always returns the identity function

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

```
from operator import add, mul
def square(x):
    return mul(x, x)
def pirate(arggg):
    print('matey')
def plunder(arggg):
    return arggg
return plunder
```

A function that always returns the identity function

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

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

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
What Would Python Print?

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

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 17</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></td>
<td>Matey</td>
</tr>
<tr>
<td>Identity function</td>
<td></td>
<td>Matey</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.

from operator import add, mul
def square(x):
    return mul(x, x)
def pirate(arggg):
    print('matey')
def plunder(arggg):
    return arggg
    return plunder
What Would Python Print?

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

```
from operator import add, mul

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

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

def plunder(arggg):
    return arggg

    return plunder
```

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></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>func square(x)</td>
<td>16</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>Error</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></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.
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

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

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)

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

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    return horse

def mask(horse):
    return horse

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return mask(horse)

mask = lambda horse: horse(2)
horse(mask)
```python
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)
horse(mask)
```
```python
def horse(mask):
    horse = mask

def mask(horse):
    return horse

mask = lambda horse: horse(2)
horse(mask)
```

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

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse

mask = lambda horse: horse(2)
horse(mask)
```python
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse

mask = lambda horse: horse(2)
horse(mask)
```
```python
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse

mask = lambda horse: horse(2)
horse(mask)
```
```python
def horse(mask):
    horse = mask
    return horse

def mask(horse):
    return horse(mask)

mask = lambda horse: horse(2)
horse(mask)
```
def horse(mask):
    horse = mask

def mask(horse):
    return horse

mask = lambda horse: horse(2)

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

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    return horse

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    return horse

def mask(horse):
    return horse(mask)

mask = lambda horse: horse(2)
horse(mask)
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse

mask = lambda horse: horse(2)
horse(mask)
```python
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse

def f1():
    mask(horse)

mask = lambda horse: horse(2)
horse(mask)
```

```
Return Value 2
```

```
Return Value 2
```

```
Return Value 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 _______________________________
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."
    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

```
>>> 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 _______________________________
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

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 _______________________________
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:
        n, last = n // 10, n % 10
        if last != digit:
            kept = last
            digits = last
    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 more than 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 other than DIGIT, for some 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 21:  
            kept = 21
        digits = 21
    return 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 except for some 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 = _______________________  # 21
    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."""

    kept, digits = 0, 0
    while n > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = kept + last
        digits = kept
    return digits

>>> 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

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
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 = 10 * 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 > 0:
    n, last = n // 10, n % 10
    if last != digit:
        kept = 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.'''
    kept, digits = 0, 0

    while n > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = kept + last*10
        digits = digits

    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."

    kept, digits = 0, 0
    while n > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = kept + last*10
        digits = digits * 10 + last
    return digits
```

Try it out:

```python
>>> 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.
    """
    kept, digits = 0, 0
    while n > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = kept + last*10
            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 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
```

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 other than DIGIT, 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 = 10*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 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 not equal to DIGIT, 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/10 + last

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 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/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

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
    digits = digits + 1

return 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
Implementing a Function

def remove(n, digit):
    """Return all digits of non-negative N except those equal to 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 round(kept * 10 ** (digits-1))