Function Examples
Announcements
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
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 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>
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.

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

print(5)
print(square(2))
```

<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>

4
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)

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>
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>
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.

```
from operator import add, mul

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

print(print(5))
None

print(5)
5

print(print(5))

    None

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

print(5)  # evaluated to None
print(5)  # evaluated to None

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

4
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')
    def g():
        return arg
    return g
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)

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

<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>

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.

```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>
</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)

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

def g():
    return arg
```

<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><code>print(5)</code></td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td><code>print(print(5))</code></td>
<td>None</td>
<td>5</td>
</tr>
<tr>
<td><code>delay(delay)()(6)()</code></td>
<td>None</td>
<td>5 None</td>
</tr>
</tbody>
</table>

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.

```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>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 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)()(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)

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

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

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)

delay = lambda arg:
    print('delayed')
    return arg

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

<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</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)

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

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

<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.

```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
```

<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>delay delayed</td>
<td>delayed</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

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

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.

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

A function that takes any argument and returns a function that returns that 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>6</td>
<td>delayed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delayed 6</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)

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

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))() | 6 | delayed delayed 6
print(delay(print)()(4)) | | 
```

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

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

```
print(print(5))  # None
print(print(print(5)))  # None
print(delay(print)(6))()  # delayed
print(delay(print)(4))()  # delayed
```

<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 None</td>
</tr>
<tr>
<td>print(print(5))</td>
<td>None</td>
<td>5 None</td>
</tr>
<tr>
<td>delay(delay)()()</td>
<td>6</td>
<td>delayed 6</td>
</tr>
<tr>
<td>print(delay(print)())(4)</td>
<td>5</td>
<td>delayed</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)

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

def g():
    return arg

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

<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 6</td>
</tr>
<tr>
<td>print(delay(print)()(4))</td>
<td>None</td>
<td>delayed 4 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.

```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 arg
```

<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>6</td>
<td>delayed</td>
</tr>
<tr>
<td>print(delay(print)()()4))</td>
<td>None</td>
<td>delayed</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)
```

This expression | Evaluates to | Interactive Output
--- | --- | ---
What Would Python Print?

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

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

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

def plunder(arggg):
    return arggg
    return plunder

5
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)
```

<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>5</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')
    return plunder(arggg):
    return plunder
```

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

Interactive Output

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

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

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

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

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

This expression

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

Evaluates to

```
5
```

Interactive Output

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.
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>5</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
```

<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></td>
<td>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.

<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>

A function that always returns the identity function:

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

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

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

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

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

A function that always returns the identity function
```

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

**This expression**

```
func square(x)
```

**Evaluates to**

```
Matey
```

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

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

```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></td>
<td>Matey</td>
</tr>
<tr>
<td><strong>func square(x)</strong></td>
<td></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

```

<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>16</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
```

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><code>func square(x)</code></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.

```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 17</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.

```
add(pirate(3)(square)(4), 1)  # 17
matey
```
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

add(pirate(3)(square)(4), 1)
func square(x)
16
pirate(pirate(pirate))(5)(7)
```

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
```

<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></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>pirate(pirate(pirate))(5)(7)</td>
<td></td>
<td>Identity function</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
```

```
A function that always returns the identity function
```

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

```
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><code>add(pirate(3)(square)(4), 1)</code></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><code>pirate(pirate(pirate))(5)(7)</code></td>
<td>Identity function</td>
<td>Matey</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
```

<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>func square(x)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></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 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.
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
```

This expression

<table>
<thead>
<tr>
<th>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>func square(x)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td><code>pirate(pirate(pirate))(5)(7)</code></td>
<td></td>
<td>Matey Matey</td>
</tr>
<tr>
<td><code>Identity function</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</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

A function that always returns the identity function

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</td>
</tr>
<tr>
<td><code>pirate(pirate(pirate)) (5)(7)</code></td>
<td>Error</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.

<table>
<thead>
<tr>
<th>This expression</th>
<th>Evalutes 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>Error</td>
<td>Matey Error</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.
```python
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

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

```
Return Value
```

```
Return Value
```

```
Return Value
```
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
    return horse

def mask(horse):
    return horse(mask)

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
    return horse(mask)

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
    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)

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

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

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)
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

```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 _______________________________

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

Read the description
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 ____________________________

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

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.
    """
    kept, digits = 0, 0
    while ________________________________:
        n, last = n // 10, n % 10
        if _______________________________
            kept = ___________________
            digits = ___________________
    return _______________________________

>>> remove(231, 3)
21
>>> remove(243132, 2)
4313
"""
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.
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

```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

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

    return kept, digits

>>> remove(231, 3)
21
>>> remove(243132, 2)
4313
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
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 = _____________________
        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
    digits = digits

return kept
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 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 + 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

```python
def remove(n, digit):
    """Return all digits of non-negative N greater than or 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 = 10 * kept + last
    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 > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = kept + last*10
        digits = kept
    return kept
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 = _____________________
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 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 = 10*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

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 \n    that are not DIGIT, for some \n    non-negative DIGIT less than 10.\n    ""

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

    kept, digits = 0, 0

    while n > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = 10*kept + last*10**digits
            digits += 1

    return kept
```

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)
        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 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

```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

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

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

kept, digits = 0, 0
while n > 0:
    n, last = n // 10, n % 10
    if last != digit:
        kept = kept/10 + last
    digits = ________________________
return _________________________
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*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 > 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.""
    kept, digits = 0, 0
    while n > 0:
        n, last = n // 10, n % 10
        if last != digit:
            kept = kept/10 + last
            digits = digits + 1
    return kept * 10 ** (digits-1)
```

- 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, 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
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

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

(Demo)

```python
@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?