What Happened with def square(square)?

def square(square):
    return mul(square, square)
from operator import mul
square(4)
What Happened with def square(square)?

def square(square):
    return mul(square, square)
from operator import mul
square(4)
What Happened with `def square(square)`?

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

from operator import mul

square(4)
```
What Happened with def square(square)?

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

from operator import mul

square(4)
```
What Happened with def square(square)?

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

from operator import mul

square(4)
What Happened with \texttt{def square(square)}?

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

```
from operator import mul
square(4)
```
What Happened with def square(square)?

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

from operator import mul

square(4)
What Happened with def square(square)?

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

from operator import mul

square(4)
What Happened with def square(square)?

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

from operator import mul

square(4)
```

Environments & values
Expressions
What Happened with def square(square)?

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

from operator import mul

square(4)
What Happened with def square(square)?

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

- `def square(square):`
  - return `mul(square, square)`
- `from operator import mul`
- `square(4)`

**Environments & values**

**Expressions**
What Happened with def square(square)?

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

```
square(4)
```

```
from operator import mul
```

```
square(4)
```

```
square: 4
```

```
square: square
```

```
mul: mul
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```

```
square: square
```
What Happened with def square(square)?

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

```
from operator import mul
```

```
square(4)
```

```
mul:

mul(a,b):

square:

square: 4

square(square):

return mul(square, square)
```

```
Environments & values
Expressions
```

Thursday, September 1, 2011
What Happened with def square(square)?

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

from operator import mul

square(4)
What Happened with def square(square)?

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

from operator import mul

square(4)
What Happened with def square(square)?

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

from operator import mul

square(4)

16
What Happened with def square(square)?

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

```
square(4)
```

```
mul(a,b):
    return mul(a, b)
square:
    return mul(square, square)
```

```
from operator import mul
```

```
Don't actually name functions and formal parameters the same thing!
```

Expressions

Environments & values

Thursday, September 1, 2011
What Happened with def square(square)?

```python
from operator import mul

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

square(4)
```

Environment diagrams will be on holiday until Wednesday.

Don't actually name functions and formal parameters the same thing!
A statement
is executed by the interpreter
to perform an action
Statements

A statement is executed by the interpreter to perform an action

Compound statements:

<header>:
  <statement>
  <statement>
  <statement>
  ...
</header>

<separating header>:
  <statement>
  <statement>
  <statement>
  ...
  ...

Thursday, September 1, 2011
A statement is executed by the interpreter to perform an action

Compound statements:

<header>:
  <statement>
  <statement>
  ...
  <separating header>:
    <statement>
    <statement>
    ...

    ...

    ...
A statement is executed by the interpret to perform an action

**Compound statements:**

```
<header>:
    <statement>
    <statement>
    ...
<separating header>:
    <statement>
    <statement>
    ...
    ...
```
A statement is executed by the interpreter to perform an action

Compound statements:

<header>:
  <statement>
  <statement>
  ...<suite>
<separating header>:
  <statement>
  <statement>
  <statement>
  ...
  ...

Thursday, September 1, 2011
Compound Statements

Compound statements:

<header>:
  <statement>
  <statement>
  ...
  Suite
  ...
<separating header>:
  <statement>
  <statement>
  <statement>
  ...
  ...

Thursday, September 1, 2011
Compound Statements

Compound statements:

<header>:
  <statement>
  <statement>
  ...

<separating header>:
  <statement>
  <statement>
  ...
  ...

A suite is a sequence of statements
Compound Statements

Compound statements:

\[
\begin{align*}
&\text{<header>:} \\
&\hspace{1cm} \text{<statement>} \\
&\hspace{1cm} \text{<statement>} \\
&\ldots \\
&\text{<separating header>:} \\
&\hspace{1cm} \text{<statement>} \\
&\hspace{1cm} \text{<statement>} \\
&\ldots \\
&\ldots
\end{align*}
\]

A suite is a sequence of statements

To “execute” a suite means to execute its sequence of statements, in order
Compound Statements

Compound statements:

<header>:
    <statement>
    <statement>
    ...
<separating header>:
    <statement>
    <statement>
    ...
...

A suite is a sequence of statements

To “execute” a suite means to execute its sequence of statements, in order

Execution Rule for a sequence of statements:

• Execute the first

• Unless directed otherwise, execute the rest
The Fibonacci Sequence
The Fibonacci Sequence

0, 1, 1, 2, 3, 5, 8, 13, ...
The Fibonacci Sequence

0, 1, 1, 2, 3, 5, 8, 13, ...

```python
def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2              # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
```
The Fibonacci Sequence

```python
def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2  # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
```

... predicates: $0, 1, 1, 2, 3, 5, 8, 13, ...$
The Fibonacci Sequence

def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2  # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
The Fibonacci Sequence

def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2              # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr

...
The Fibonacci Sequence

```python
def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2  # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
```

0, 1, 1, 2, 3, 5, 8, 13, ...

...
The Fibonacci Sequence

def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2  # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
The Fibonacci Sequence

```
... pred: curr:
```

```python
def fib(n):
    '''Compute the nth Fibonacci number, for n >= 2.'''
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2              # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
```
def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2  # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
Higher-Order Functions Introduction

(Demo)
Pig Introduction

(Demo)