Names, Assignment, and User-Defined Functions

Discussion Question 1

What is the value of the final expression in this sequence?

```python
>>> f = min
>>> f = max
>>> g, h = min, max
>>> max = g
>>> max(f(2, g(h(1, 5), 3)), 4)
```

Environment Diagrams

<table>
<thead>
<tr>
<th>Environment diagrams visualize the interpreter's process.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Environment Diagram" /></td>
</tr>
</tbody>
</table>

Assignment Statements

Execution rule for assignment statements:

1. Evaluate all expressions to the right of = from left to right.
2. Bind all names to the left of = to those resulting values in the current frame.
Discussion Question 1 Solution

```
func min(...) 3
f(2, g(h(1, 5), 3))
g(h(1, 5), 3)

func max(...) 5
h(1, 5)
func min(...) 3
```

Defining Functions

Assignment is a simple means of abstraction: binds names to values.

Function definition is a more powerful means of abstraction: binds names to expressions.

```
>>> def myfunc(a, b):  # Function definition is a powerful means of abstraction.
...     return a + b

# Calling the function
result = myfunc(3, 4)
```

Calling User-Defined Functions

```
from operator import mul

square = lambda x: x * x

def area(width, height):
    return width * height

# Calling the function
area(5, 7)
```

Looking Up Names In Environments

```
# Every expression is evaluated in the context of an environment.

# So far, the current environment is either:
#   - The global frame alone, or
#   - A local frame, followed by the global frame.

# Most important two things I'll say all day:
#   - An environment is a sequence of frames.
#   - A name evaluates to the value bound to that name in the earliest frame of the current
  #     environment in which that name is found.

# E.g., to look up a name in the body of the square function:
#   - Look for that name in the local frame.
#   - If not found, look for it in the global frame.
#   (Built-in names like "max" are in the global frame too, but we don't draw them in environment diagrams.)
```