Announcements
Multiple Environments
Life Cycle of a User-Defined Function

Def statement:
- Name: `square(x):`
- Body (return statement): `return mul(x, x)`

What happens?
- A new function is created!
  - Name bound to that function in the current frame.

Call expression:
- Operator: `square(2+2)`
  - Function (value of operator) called on arguments (values of operands).
  - Argument: `4` (value of `2+2`).

Calling/Applying:
- Argument: `4`
- Signature: `square(x)`: 4
  - A new frame is created!
  - Parameters bound to arguments.
  - Body is executed in that new environment.
  - Return value: `16`.
Multiple Environments in One Diagram!

```python
1 from operator import mul
2 def square(x):
3     return mul(x, x)
4     square(square(3))
```

Interactive Diagram
Multiple Environments in One Diagram!

```python
1 from operator import mul
2 def square(x):
3     return mul(x, x)
4 square(square(3))
```

Interactive Diagram
Multiple Environments in One Diagram!

An environment is a sequence of frames.

- The global frame alone
- A local, then the global frame

Interactive Diagram
Names Have No Meaning Without Environments

Every expression is evaluated in the context of an environment.

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

An environment is a sequence of frames.

- The global frame alone
- A local, then the global frame

Interactive Diagram
Names Have Different Meanings in Different Environments

A call expression and the body of the function being called are evaluated in different environments.

```
1 from operator import mul
2 def square(square):
3     return mul(square, square)
4 square(4)
```

Every expression is evaluated in the context of an environment.

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

Interactive Diagram
Miscellaneous Python Features

Division
Multiple Return Values
Source Files
Doctests
Default Arguments

(Demo)
Conditional Statements
A statement is executed by the interpreter to perform an action

**Compound statements:**

1. **Statement**
2. **Clause**
3. **Suite**

- The first header determines a statement’s type
- The header of a clause “controls” the suite that follows
- def statements are compound statements
**Compound Statements**

**Compound statements:**

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 statement
- Unless directed otherwise, execute the rest
Conditional Statements

(Demo)

```python
def absolute_value(x):
    """Return the absolute value of x."""
    if x < 0:
        return -x
    elif x == 0:
        return 0
    else:
        return x
```

Execution Rule for Conditional Statements:

1. Evaluate the header's expression.
2. If it is a true value, execute the suite & skip the remaining clauses.

Syntax Tips:

1. Always starts with "if" clause.
2. Zero or more "elif" clauses.
3. Zero or one "else" clause, always at the end.
def absolute_value(x):
    """Return the absolute value of x."""
    if x < 0:
        return -x
    elif x == 0:
        return 0
    else:
        return x

George Boole
def absolute_value(x):
    """Return the absolute value of x."""
    if x < 0:
        return -x
    elif x == 0:
        return 0
    else:
        return x

George Boole

False values in Python: False, 0, '', None (more to come)

True values in Python: Anything else (True)

Read Section 1.5.4!

Reading: http://composingprograms.com/pages/15-control.html#conditional-statements
Iteration
While Statements

(Demo)

1. Evaluate the header’s expression.
2. If it is a true value, execute the (whole) suite, then return to step 1.

Execution Rule for While Statements:

1. Evaluate the header’s expression.
2. If it is a true value, execute the (whole) suite, then return to step 1.