61A Lecture 3

Wednesday, August 29
Life Cycle of a User-Defined Function

Def statement: `square(x):`
- Name
- Formal parameter: `x`
- Body (return statement): `return mul(x, x)`
- Return expression

What happens?
- Function created
- Name bound

Call expression: `square(2+2)`
- Operator: `square`
- Function: `func square(x)`
- Operand: `2+2`
- Argument: `4`

Calling/Applying:
- Argument: `4`
- Signature
- Body executed: `return mul(4, 4)`
- Return value: `16`

- New frame!
- Params bound
- Body executed
Multiple Environments in One Diagram!

(Demo)

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

Example: [http://goo.gl/668fU](http://goo.gl/668fU)
Multiple Environments in One Diagram!

An environment is a sequence of frames.

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

Example: [http://goo.gl/668fU](http://goo.gl/668fU)
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.

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

Example: [http://goo.gl/668fU](http://goo.gl/668fU)
Formal Parameters

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

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

Formal parameters have local scope

(Demo)

Example: http://goo.gl/OapJa
Python Feature Demonstration

Operators

Multiple Return Values

Docstrings

Doctests

Default Arguments

Statements
Statements

A statement is executed by the interpreter to perform an action

Compound statements:

- 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:

\[
\text{<header>:
  \text{<statement>}
  \text{<statement>}
  \ldots
}\]

\[
\text{<separating header>:
  \text{<statement>}
  \text{<statement>}
  \ldots
}\]

... 

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
Local Assignment

Example:

```python
def percent_difference(x, y):
    difference = abs(x-y)
    return 100 * difference / x
diff = percent_difference(40, 50)
```

**Execution rule for assignment statements:**

1. Evaluate all expressions right of `=`, from left to right.
2. Bind the names on the left the resulting values in the first frame of the current environment.

Example: [http://goo.gl/wcF71](http://goo.gl/wcF71)
Conditional Statements

```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:

Each clause is considered in order.

1. Evaluate the header's expression.

2. If it is a true value, execute the suite & skip the remaining clauses.
def absolute_value(x):
    """Return the absolute value of x."""
    if x > 0:
        return x
    elif x == 0:
        return 0
    else:
        return -x
Boolean Contexts

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

Two boolean contexts

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

True values in Python: Anything else (True)

Read Section 1.5.4!
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.

Example: http://goo.gl/07y0D