Functions
Announcements
Expressions
Types of expressions

An expression describes a computation and evaluates to a value

\[ 18 + 69 \]
\[ \frac{6}{23} \]
\[ \sin \pi \]
\[ \log_2 1024 \]
\[ 2^{100} \]
\[ f(x) \]
\[ 7 \mod 2 \]
\[ \left| -1869 \right| \]
\[ \sum_{i=1}^{100} i \]
\[ \sqrt{3493161} \]
\[ \lim_{x \to \infty} \frac{1}{x} \]
\[ \binom{69}{18} \]
All expressions can use function call notation

(Demo)
Anatomy of a Call Expression

Evaluation procedure for call expressions:

1. Evaluate the operator and then the operand subexpressions

2. Apply the function that is the value of the operator to the arguments that are the values of the operands

Operators and operands are also expressions

So they evaluate to values
Evaluating Nested Expressions

\[\text{mul}(\text{add}(4, \text{mul}(4, 6)), \text{add}(3, 5))\]
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of subexpression

1st argument to mul

Value of the whole expression

mul(add(4, mul(4, 6)), add(3, 5))

mul

add(4, mul(4, 6))

add

4

24

mul(4, 6)

mul

4

6

add

4

28

mul

add

3

5

add

8

add

3

5

224
Names, Assignment, and User-Defined Functions

(Demo)
Types of Expressions

**Primitive expressions:**

2
Number or Numeral

add
Name

'hello'
String

**Call expressions:**

\[ \text{max}(\text{min}(\text{pow}(3, 5), -4), \text{min}(1, -2)) \]

An operand can also be a call expression

\[ \text{max}(2, 3) \]
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)
```

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

???
Environment Diagrams
Environment Diagrams

Environment diagrams visualize the interpreter’s process.

Code (left):
Statements and expressions
Arrows indicate evaluation order

Frames (right):
Each name is bound to a value
Within a frame, a name cannot be repeated

1. `from math import pi`
2. `tau = 2 * pi`

Global frame
Name pi Value 3.1416

Just executed
Import statement

Next to execute
Assignment statement

(Demo)
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.

http://pythontutor.com/composingprograms.html#code=a%20%3D%201%0Ab%20%3D%202%0Ab,%20a%20%3D%20a%20%2B%20b,%20b&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D
Discussion Question 1 Solution

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

(Demo)
Defining Functions
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

Function **signature** indicates how many arguments a function takes

```python
>>> def <name>(<formal parameters>):
    return <return expression>
```

Function **body** defines the computation performed when the function is applied

**Execution procedure for def statements:**

1. Create a function with signature `<name>(<formal parameters>)`
2. Set the body of that function to be everything indented after the first line
3. Bind `<name>` to that function in the current frame
Calling User-Defined Functions

Procedure for calling/applying user-defined functions (version 1):
1. Add a local frame, forming a new environment
2. Bind the function's formal parameters to its arguments in that frame
3. Execute the body of the function in that new environment

```python
from operator import mul
def square(x):
    return mul(x, x)
square(-2)
```
Calling User-Defined Functions

Procedure for calling/applying user-defined functions (version 1):

1. Add a local frame, forming a new environment
2. Bind the function's formal parameters to its arguments in that frame
3. Execute the body of the function in that new environment

from operator import mul

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

square(-2)
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 some 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.)
  
  (Demo)