Announcements

Data Representations

Data Representations

Functions with Shared Local State

Linked Lists (Sneak Preview)

Pairs Implemented as Functions

An Inefficient Dictionary Implementation

Dispatch Dictionaries
Dispatch Dictionaries

Enumerating different messages in a conditional statement isn’t very convenient:
- Equality tests are repetitive
- We can’t add new messages without re-writing the dispatch function

A dispatch dictionary has messages as keys and functions (or data objects) as values. Dictionaries handle the message look-up logic; we can concentrate on implementing behavior. 

```python
def box_dispatch(contents):
    def dispatch(m):
        if m == 'contents':
            return contents
        if m == 'put':
            def put(value):
                nonlocal contents
                contents = value
            return put
    return dispatch

def box_dict(contents):
    def put(value):
        d['contents'] = value
        d = {'contents': contents, 'put': put}
    return d
```

Constraint Networks

Solving for Variables in an Equation

```
\begin{align*}
    a + b &= c \\
    a &= c - b \\
    b &= c - a \\
\end{align*}
```

Algebraic equations are declarative: They describe a relation among different quantities. Python functions are procedural: They describe how to compute a result from a set of input arguments.

Constraint programming:
- We define the relationship between quantities
- We provide values for the “known” quantities
- The system computes values for the “unknown” quantities

**Challenge:** We want a general means of combination.

A Constraint Network for Temperature Conversion

Combination idea: All intermediate quantities have values too.

```
\begin{align*}
    u &= v \\
    a + b + c &= 5 \times (f - 32) \\
\end{align*}
```

Both sides of the equation are equal; they must be the same quantity.