Data Abstraction

• Compound values combine other values together
  • A date: a year, a month, and a day
  • A geographic position: latitude and longitude
• Data abstraction lets us manipulate compound values as units
• Isolate two parts of any program that uses data:
  • How data are represented (as parts)
  • How data are manipulated (as units)
• Data abstraction: A methodology by which functions enforce an abstraction barrier between representation and use

Rational Numbers

<table>
<thead>
<tr>
<th>numerator</th>
<th>denominator</th>
</tr>
</thead>
</table>

Exact representation of fractions

A pair of integers

As soon as division occurs, the exact representation may be lost! (Demo)

Assume we can compose and decompose rational numbers:

Constructor

\[ \text{rational}(n, d) \text{ returns a rational number} \]

Selectors

\[ \text{numer}(x) \text{ returns the numerator of} \ x \]

\[ \text{denom}(x) \text{ returns the denominator of} \ x \]

Rational Number Arithmetic

\[ \begin{align*}
\frac{3}{2} \times \frac{3}{5} &= \frac{9}{10} \\
\frac{3}{2} + \frac{3}{5} &= \frac{21}{10}
\end{align*} \]

Rational Number Arithmetic Implementation

```python
def mul_rational(x, y):
    return rational(numer(x) * numer(y),
                   denom(x) * denom(y))

def add_rational(x, y):
    nx, dx = numer(x), denom(x)
    ny, dy = numer(y), denom(y)
    return rational(nx * dy + ny * dx, dx * dy)

def print_rational(x):
    print(numer(x), '/', denom(x))

def rationals_are_equal(x, y):
    return numer(x) * denom(y) == numer(y) * denom(x)
```

These functions implement an abstract representation for rational numbers
Representing Pairs Using Lists

```python
>>> pair = [1, 2]  # A list literal: Comma-separated expressions in brackets
>>> x, y = pair  # "Unpacking" a list
>>> x
1
>>> y
2
>>> pair[0]  # Element selection using the selection operator
1
>>> pair[1]
2
>>> from operator import getitem  # Element selection function
>>> getitem(pair, 0)
1
>>> getitem(pair, 1)
2
```

More lists next lecture

Representing Rational Numbers

```python
def rational(n, d):
    # Construct a rational number that represents N/D.
    return [n, d]
def numer(x):
    # Return the numerator of rational number X.
    return x[0]
def denom(x):
    # Return the denominator of rational number X.
    return x[1]
```

```python
g = gcd(n, d)  # Greatest common divisor
return [n//g, d//g]
```

(Demo)

Reducing to Lowest Terms

Example:

\[
\frac{3}{2} + \frac{5}{3} = \frac{5}{2} \quad \frac{2}{5} + \frac{1}{10} = \frac{1}{2}
\]

\[
\frac{15}{6} + \frac{1}{3} = \frac{5}{2} \quad \frac{25}{15} + \frac{1}{25} = \frac{1}{2}
\]

Abstraction Barriers

<table>
<thead>
<tr>
<th>Parts of the program that...</th>
<th>Treat rationals as...</th>
<th>Using...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use rational numbers to perform computation</td>
<td>Whole data values</td>
<td>add_rational, mul_rational, rationals_are_equal, print_rational</td>
</tr>
<tr>
<td>Create rationals or implement rational operations</td>
<td>Numerators and denominators</td>
<td>rational, numer, denom</td>
</tr>
<tr>
<td>Implement selectors and constructor for rationals</td>
<td>Two-element lists</td>
<td>List literals and element selection</td>
</tr>
</tbody>
</table>

Implementation of lists

What is Data?

- We need to guarantee that constructor and selector functions work together to specify the right behavior.
- Behavior condition: If we construct rational number \( x \) from numerator \( n \) and denominator \( d \), then \( \text{numerator}(x) / \text{denominator}(x) \) must equal \( n/d \).
- Data abstraction uses selectors and constructors to define behavior.
- If behavior conditions are met, then the representation is valid.

You can recognize an abstract data representation by its behavior.

(Demo)
```python
def rational(n, d):
    def select(name):
        if name == 'n':
            return n
        elif name == 'd':
            return d
    return select

def numer(x):
    return x('n')

def denom(x):
    return x('d')
```

This function represents a rational number.

Rationals Implemented as Functions

Constructor is a higher-order function

Selector calls `x`

Interactive Diagram