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

- 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:
  - rational(n, d) returns a rational number x

- Selectors:
  - numer(x) returns the numerator of x
  - denom(x) returns the denominator of x

Rational Number Arithmetic

\[
\begin{align*}
\text{Example} & \quad \frac{3}{2} + \frac{3}{5} = \frac{21}{10} \\
\text{General form} & \quad \frac{nx}{dx} + \frac{ny}{dy} = \frac{nx*dy + ny*dx}{dx*dy}
\end{align*}
\]

Rational Number Arithmetic Implementation

\[
\begin{align*}
def \text{add_rational}(x, y): & \\
& \text{num, dx} = \text{numer}(x), \text{denom}(x) \\
& \text{ny, dy} = \text{numer}(y), \text{denom}(y) \\
& \text{return rational}(\text{num} + \text{ny} * \text{dx}, \text{dx} + \text{dy})
\end{align*}
\]

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

>>> 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

def rational(n, d):
    """Construct a rational number that represents N/D."""
    return [n, d]

Representing Rational Numbers

def divide_rational(x, y):
    return [x[0] * y[1], x[1] * y[0]]

Reducing to Lowest Terms

Example:

\[
\frac{3}{2} \times \frac{5}{3} = \frac{3}{2} \times \frac{1}{3} = \frac{5}{2}
\]

from math import gcd

def rational(n, d):
    """Construct a rational that represents n/d in lowest terms."""
    g = gcd(n, d)
    return [n // g, d // g]

Abstraction Barriers

Parts of the program that... Treat rationals as... Using...

Use rational numbers to perform computation whole data values add_rational, mul_rational, rational_are_equal, print_rational
Create rationals or implement rational operations numerators and denominators rational, num, denom
Implement selectors and constructor for rationals two-element lists list literals and element selection

Implementation of lists

Violating Abstraction Barriers

add_rational([[1, 2], [1, 4]])

def divide_rational(x, y):
    return [[x[0] * y[1], x[1] * y[0]]

Data Representations

What are 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 numerator/denominator must equal x/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
Rationals Implemented as Functions

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

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

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

This function represents a rational number.

Constructor is a higher-order function.

Selector calls `x`.

Dictionaries

Dictionaries are unordered collections of key-value pairs.

Dictionary keys do have two restrictions:

- A key of a dictionary cannot be a list or a dictionary (or any mutable type).
- Two keys cannot be equal; there can be at most one value for a given key.

This first restriction is tied to Python’s underlying implementation of dictionaries.

The second restriction is part of the dictionary abstraction.

If you want to associate multiple values with a key, store them all in a sequence value.

```python
x = rational(3, 8)
numer(x)
```