1.1. Unpacking a Sequence into Separate Variables

Problem

You have an N-element tuple or sequence that you would like to unpack into a collection

of N variables.

Solution

Any sequence (or iterable) can be unpacked into variables using a simple assignment

operation. The only requirement is that the number of variables and structure match

the sequence. For example:

**>>>** p = (4, 5)

**>>>** x, y = p

**>>>** x

4

**>>>** y

5

>>>

**>>>** data = [ 'ACME', 50, 91.1, (2012, 12, 21) ]

**>>>** name, shares, price, date = data

**>>>** name

'ACME'

**>>>** date

(2012, 12, 21)

**>>>** name, shares, price, (year, mon, day) = data

**>>>** name

'ACME'

**>>>** year

2012

**>>>** mon

12

**>>>** day

21

>>>

If there is a mismatch in the number of elements, you’ll get an error. For example:

**>>>** p = (4, 5)

**>>>** x, y, z = p

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: need more than 2 values to unpack

>>>

Discussion

Unpacking actually works with any object that happens to be iterable, not just tuples or

lists. This includes strings, files, iterators, and generators. For example:

**>>>** s = 'Hello'

**>>>** a, b, c, d, e = s

**>>>** a

'H'

**>>>** b

'e'

**>>>** e

'o'

>>>

When unpacking, you may sometimes want to discard certain values. Python has no

special syntax for this, but you can often just pick a throwaway variable name for it. For

example:

**>>>** data = [ 'ACME', 50, 91.1, (2012, 12, 21) ]

**>>>** \_, shares, price, \_ = data

**>>>** shares

50

**>>>** price

91.1

>>>

However, make sure that the variable name you pick isn’t being used for something else

already