ACM/CS 114
Parallel algorithms for scientific applications

Michael A. G. Aïvázis

California Institute of Technology

Spring 2012

Strings

- string literals

Literal	*Description*
`'Hello'`	single quotes
`"Hello",`	double quotes
`"It's embedded"`	mix and match
`'''Hello''',"""Hello"""`	triple quotes – extend over multiple lines

- strings are immutable ordered sequences of unicode characters
  - no `char` type: `'a'` is a string with one character
  - native support for all code pages
- common operations: given the strings s_i:

Expression	*Description*
`s1+s2, s*4`	concatenation, repetition
`s[3], s[3:4]`	indexing, slicing
`len(s)`	length
`'H' in s`	membership
`for x in s`	iteration

String coercions

- all objects are representable as strings
- coercion can be triggered explicitly
  - using the `str` constructor
  - using the `repr` built-in function
- strings have a powerful formatting method:

```
1     'temperature={:+13.4f}, pressure={:13.4f}'.format(T, P)
```

- more details after we go over the rules for function calls

Tuples

- tuples are immutable ordered inhomogeneous sequences of objects

Literal	*Description*
`()`	the empty tuple
`(1,)`	a tuple with one item
`(1,2,3,4)`	a longer tuple
`(1,'Hello', 'world')`	tuple elements don't have to be the same type
`(1,2,('Hello', 'world'),4)`	tuples nest arbitrarily deeply

- common operations:

Expression	*Description*
`t1+t2, t*4`	concatenation, repetition
`t[3], t[3:4]`	indexing, slicing
`len(t)`	length
`x in t`	membership
`for x in t`	iteration

Lists

- lists are mutable ordered inhomogeneous sequences of objects

Literal	*Description*
`[]`	the empty list
`[1]`	a list with one item
`[1,2,3,4]`	a longer list
`[1,'Hello', 'world']`	list elements don't have to be the same type
`[1,2,['Hello', 'world'],4]`	lists nest arbitrarily deeply

- common operations:

Expression	*Description*
`l1+l2, l*4`	concatenation, repetition
`len(l), x in l, for x in l`	length, membership, iteration
`l[3], l[3:4]`	indexing, slicing
`del l[3], l[3:7]=[]`	shrink
`l[3:7]=[1,2]`	slice assignment
`l.append(1)`	add to the end of the list

Sets

- sets are mutable unordered inhomogeneous containers of objects

Literal	*Description*
`set()`	the empty set
`{1})`	a set with one item
`{1,2,3,4}`	a longer set
`{1,'Hello', 'world'}`	set elements don't have to be the same type

- common operations:

Expression	*Description*
`s1|s2`	union
`s1&s2`	intersection
`s1-s2`, `s1^s2`	difference, symmetric difference
`len(s)`, `x in s`, `for x in s`	length, membership, iteration
`s.add(x)`	add
`s.discard(x)`	remove if present
`s.remove(x)`	remove; raise `KeyError` if not present

Dictionaries

- dictionaries are mutable inhomogeneous associative maps

Literal	*Description*
`{}`	the empty dictionary
`{'first':'Guido', 'last':'van Rossum'}`,	a dictionary with two items

- dictionary keys must be *hashable*; dictionary values may be of any type
- common operations

Expression	*Description*
`d['first']='Guido'`	make an association
`d['first']`	value retrieval raise `KeyError` if key not present
`d.get('first'),d.get('first', default='')`	value retrieval
`len(d)`	the number of keys
`x in d`	key presence
`d.keys(),d.values(),d.items()`	views of the dictionary contents