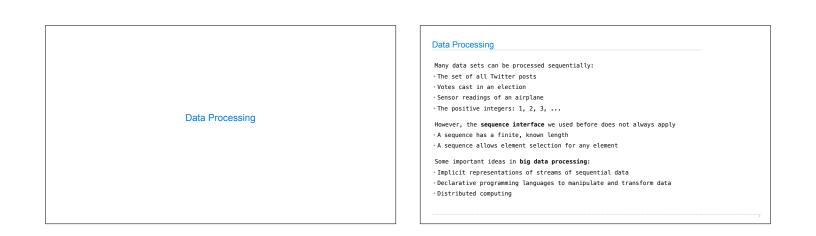
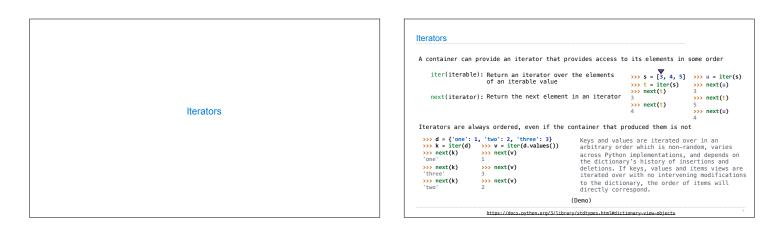


Announcements





For Statements	The For Statement for <name> in <expression>: <uite>  1.Evaluate the header <expression>, which must evaluate to an iterable object 2.For each element in that sequence, in order: A.Bind <name> to that element in the first frame of the current environment B.Execute the <suite>  When executing a for statement, iter returns an iterator and mext provides each item: &gt;&gt;&gt; counts = [1, 2, 3] &gt;&gt;&gt; for item in counts: 1 2 3 while True: 1 2 3 4 1 2 3 5 1 1 2 3 5 1 1 2 3 1 2 3 1 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 2 3 2 3 2 3 2 3 2 3 2 3 3 2 3 3 3 3 3 3 3 3 3 3 4 1 4 3 4 1 4 1 1 1 1 2 3 3 4 1 1 1 1 2 3 3 3 4 1 1 3 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 3 2 3 3 3 4 1 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4</suite></name></expression></uite></expression></name>
----------------	--

## Processing Iterators

A StopIteration exception is raised whenever next is called on an empty iterator

>>> contains('strength', 'stent')
True
>>> contains('strength', 'rest')
False
>>> contains('strength', 'tenth')
True

def contains(a, b):
 ai = iter(a)
 for x in b:
 try:
 while next(ai) != x:
 pass # do nothing
 except StopIteration:
 return False
 return True



Built-in Functions for Iteration	
Many built-in Python sequence operations n	return iterators that compute results lazily
<pre>map(func, iterable):</pre>	Iterate over func(x) for x in iterable
filter(func, iterable):	Iterate over x in iterable if func(x)
<pre>zip(first_iter, second_iter):</pre>	Iterate over co-indexed (x, y) pairs
reversed(sequence):	Iterate over x in a sequence in reverse order
To view the contents of an iterator, place	e the resulting elements into a container
list(iterable):	Create a list containing all x in iterable
<pre>tuple(iterable):</pre>	Create a tuple containing all x in iterable
sorted(iterable):	Create a sorted list containing x in iterable
	(Demo)

