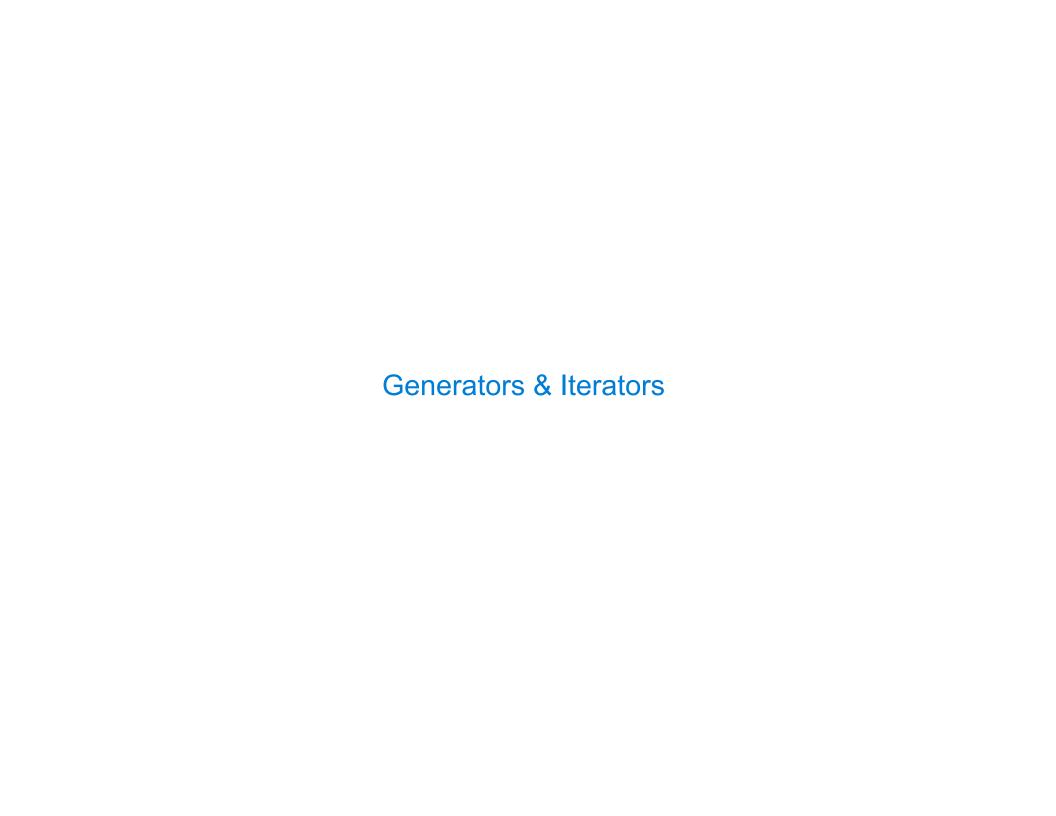


Generators and Generator Functions

```
>>> def plus_minus(x):
...     yield x
...     yield -x
>>> t = plus_minus(3)
>>> next(t)
3
>>> next(t)
-3
>>> t
<generator object plus_minus ...>
```

A generator function is a function that yields values instead of returning them
A normal function returns once; a generator function can yield multiple times
A generator is an iterator created automatically by calling a generator function
When a generator function is called, it returns a generator that iterates over its yields

(Demo)



Generator Functions can Yield from Iterables

```
A yield from statement yields all values from an iterator or iterable (Python 3.3)
                              >>> list(a_then_b([3, 4], [5, 6]))
                              [3, 4, 5, 6]
                          def a_then_b(a, b): def a_then_b(a, b):
                              for x in a:
                                                      yield from a
                                  yield x
                                                      yield from b
                              for x in b:
                                  yield x
                                   >>> list(countdown(5))
                                   [5, 4, 3, 2, 1]
                               def countdown(k):
                                   if k > 0:
                                       yield k
                                       yield from countdown(k-1)
```

(Demo)

0

Example: Partitions

Yielding Partitions

A partition of a positive integer n, using parts up to size m, is a way in which n can be expressed as the sum of positive integer parts up to m in increasing order.

partitions(6, 4)

```
2 + 4 = 6
                                 def count_partitions(n, m):
                                     if n == 0:
1 + 1 + 4 = 6
                                         return 1
3 + 3 = 6
                                     elif n < 0:
                                         return 0
1 + 2 + 3 = 6
                                     elif m == 0:
1 + 1 + 1 + 3 = 6
                                         return 0
                                     else:
2 + 2 + 2 = 6
                                          with m = count partitions(n-m, m)
1 + 1 + 2 + 2 = 6
                                          without m = count partitions(n, m-1)
1 + 1 + 1 + 1 + 2 = 6
                                          return with m + without m
1 + 1 + 1 + 1 + 1 + 1 = 6
                                (Demo)
```