# Python Iterables

- Iterable object
  - Can be used with for loops

```
for x in iterable:
    # do something with x
```

  - Has the __iter__() method defined which returns an iterator
    - You can get an iterator for an iterable with iter(iterable)

- Iterators
  - Objects that have the `__next__()` method defined
  - Calling `next(x)` on iterator `x` either
    - Returns the next item in the sequence
    - Raises a `StopIteration` exception if there are no more items
  - `next(x)` calls the object's `__next__()` method

- Iterable classes
  - It is common to make a class both an iterable and an iterator
    - Define `__iter__()` to return self
    - Define `__next__()` accordingly

- Generators
  - A function that uses yield statements
  - Calling the function returns a generator object, but does not yet execute the body of the function
  - Generator objects are iterators

- When getting the next item
    - The code in the function is run until a yield statement is reached, and then the yielded value is the next item
    - After yielding, the function is suspended (or paused)
        - Local variables are remembered
        - The next time an item is needed, execution resumes where it left off

- When the generator function returns, the iterator will raise StopIteration