Iterators

- Basic container types, such as list, tuple, and set, qualify as iterable types, which allows them to be used as an iterable object in a for loop.

for *element* **in** *iterable*:

- An iterator is an object that manages an iteration through a series of values. If variable, **i**, identifies an iterator object, then each call to the built-in function, **next(i)**, produces a subsequent element from the underlying series, with a **StopIteration** exception raised to indicate that there are no further elements.
- An iterable is an object, **obj**, that produces an iterator via the syntax **iter(obj)**.

Generators

- The most convenient technique for creating iterators in Python is through the use of generators.
- A generator is implemented with a syntax that is very similar to a function, but instead of returning values, a yield statement is executed to indicate each element of the series.
- For example, a generator for the factors of n:

```
def factors(n):                    # generator that computes factors
  for k in range(1,n+1):
    if n % k == 0:                 # divides evenly, thus k is a factor
      yield k                      # yield this factor as next result
```

Comprehension Syntax

- A very common programming task is to produce one series of values based upon the processing of another series.
- Often, this task can be accomplished quite simply in Python using what is known as a comprehension syntax.

```
[ expression for value in iterable if condition ]
```

- This is the same as

```
result = [ ]
for value in iterable:
  if condition:
    result.append(expression)
```

Packing

- If a series of comma-separated expressions are given in a larger context, they will be treated as a single tuple, even if no enclosing parentheses are provided.
- For example, consider the assignment

$$\text{data} = 2, 4, 6, 8$$

- This results in identifier, data, being assigned to the tuple (2, 4, 6, 8). This behavior is called **automatic packing** of a tuple.

Unpacking

- As a dual to the packing behavior, Python can automatically unpack a sequence, allowing one to assign a series of individual identifiers to the elements of sequence.
- As an example, we can write

```
a, b, c, d = range(7, 11)
```

- This has the effect of assigning a=7, b=8, c=9, and d=10.