Iterables and Iterators

Amish Shah

Iteration

```
cities = ['Los Angeles', 'San Francisco', 'Monterey', 'San Diego']
count = len(cities)
i = 0
while i < count:
    city = cities[i]
    print(city)
    i += 1
```

```
cities = ['Los Angeles', 'San Francisco', 'Monterey', 'San Diego']
for i in range(len(cities)):
    city = cities[i]
    print(city)
```

The right way

```
for city in cities:
    print(city)
```

What is a Iterable?

- An **iterable** object is an object that implements __iter__, which is expected to return an **iterator** object.
- Examples - list, dictionary, file, etc.

```
cities = ['Los Angeles', 'San Francisco', 'Monterey', 'San Diego']
dir(cities)
```

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 ....,
 '__iter__',
 ....,.
 'reverse',
 'sort']
```

What is an iterator?

- An **iterator** is an object that implements __next__, which is expected to return the next element of the iterable object that returned it, and raise a StopIteration exception when no more elements are available.
- In the simplest case the iterable will implement __next__ itself and return self in __iter__.

```
cities = ['Los Angeles', 'San
Francisco', 'Monterey', 'San Diego']
it = iter(cities)
dir(it)
```

```
['__class__',
 '__delattr__',
 .....,
 '__iter__',
.....,
 '__next__',
 ....,
 '__subclasshook__']
```

Generalized iterator loop

```
for name in iterable:
    statements
```

- Iterator produces a stream of values
- Assign name to stream
- Execute statements for each element

Lists are iterable

```
for name in ['Sacramento', 'Phoenix', 'Salem', 'Olympia']:
  print(name)
```

```
Sacramento
Phoenix
Salem
Olympia
```

Strings are iterable

```
for ch in 'Sacramento':
    print(ch)
```

```
S
a
c
r
a
m
e
n
t
o
```

Dictionaries are iterable

```
states = {'California': 'Sacramento', 'Oregon':'Salem',
'Arizona':'Phoenix'}
for key in states:
    print(key)
```

```
California
Oregon
Arizona
```

Iterating dictionary key/values

```
for key in states.keys():
    print(key)
for value in states.values():
    print(value)
for key, value in states.items():
    print(key, value)
```

Tuples are iterable

```
states = ('California', 'Oregon','Arizona')
for state in states:
    print(state)
```

```
California
Oregon
Arizona
```

Files are iterable

```
with open('/Users/amishshah/constitution.txt') as file:
    for line in file:
        print(repr(line))
```

```
'We the People of the United States, in Order to form a more perfect Union,
establish Justice, insure domestic Tranquility, provide for the common defence,
promote the general Welfare, and secure the Blessings of Liberty to ourselves and
our Posterity, do ordain and establish this Constitution for the United States of
America.\n'
'Article. I.\n'
'Section. 1.\n'
```

Directories

```
from os import scandir
for file in scandir('.'):
    print(file)
```

Pattern matching

```
import re
p = re.compile(r'\d+')
for match in p.finditer('12 drummers drumming, 11 ... 10 ...'):
    print(match)
```

Itertools

```
from itertools import count, cycle, repeat
for i in count(10):
    print(i)   # 10 11 12 13 14 15 ....
for i in cycle('ABCDEF'):
    print(i)   # A B C D E F A B C D E F ...
for i in repeat(10, 3):
    print(i)    # 10 10 10
```

Itertools - 2

```
from itertools import accumulate, chain
for i in accumulate([1, 2, 3, 4, 5]):
    print(i) # 1 3 6 10 15
for i in chain([1, 2, 3, 4, 5], [10, 11, 12, 13, 14]):
    print(i) # 1 2 3 4 5 10 11 12 13 14
```

Other uses of iterations

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8]
for product in [number*number for number in numbers]:
    print(product) # 1 4 9 16 25 36 49 64
print(sum(numbers)) #36
print(min(numbers)) # 1
print(max(numbers))# 8
```

Using enumerations

```
cities = ['Los Angeles', 'San Francisco', 'Monterey', 'San
Diego']
for i, city in enumerate(cities):
    print(i, city)
```

```
0 Los Angeles
1 San Francisco
2 Monterey
3 San Diego
```

Enumerating files

```
with open('/Users/amishshah/constitution.txt') as file:
    for lineno, line in enumerate(file):
        print(lineno, repr(line))
```

Iterating a pair of streams

```
states = ['California', 'Oregon', 'Arizona']
capitals = ['Sacramento', 'Salem', 'Phoenix']
for state, capital in zip(states, capitals):
    print(state, capital)
```

```
California Sacramento
Oregon Salem
Arizona Phoenix
```

Customizing an iteration

```
sentence = "The quick brown fox jumps over the lazy dog"
for word in sentence.split():
    if len(word) % 2  == 0:
        print(len(word), word)
```

```
4 over
4 lazy
```

Using a generator

```
def get_words_with_even_chs (sentence):
    for word in sentence.split():
        if len(word) % 2 == 0:
            yield word

words = get_words_with_even_chs(sentence)
for word in words:
    print(len(word), word)
```

Creating an iterator

```
class MyRange:
    def __init__ (self, start, end):
        self.value = start
        self.end = end

    def __iter__(self):
        return self

    def __next__(self):
        if self.value >= self.end:
            raise StopIteration
        value = self.value
        self.value += 1
        return value
```

That’s it folks!

- Github link for the presentation and code
  - https://github.com/amishpshah/python_talks
- LinkedIn
  - https://www.linkedin.com/in/amishshah
- Email
  - ashahengineer@gmail.com