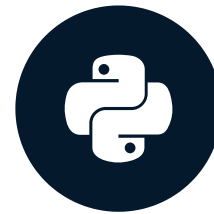


Introduction to iterators

PYTHON TOOLBOX



Hugo Bowne-Anderson
Data Scientist at DataCamp

Iterating with a for loop

- We can iterate over a list using a for loop

```
employees = ['Nick', 'Lore', 'Hugo']  
for employee in employees:  
    print(employee)
```

```
Nick  
Lore  
Hugo
```

Iterating with a for loop

- We can iterate over a string using a for loop

```
for letter in 'DataCamp':  
    print(letter)
```

```
D  
a  
t  
a  
C  
a  
m  
p
```

Iterating with a for loop

- We can iterate over a range object using a for loop

```
for i in range(4):  
    print(i)
```

```
0
```

```
1
```

```
2
```

```
3
```

Iterators vs. iterables

- Iterable
 - Examples: lists, strings, dictionaries, file connections
 - An object with an associated `iter()` method
 - Applying `iter()` to an iterable creates an iterator
- Iterator
 - Produces next value with `next()`

Iterating over iterables: next()

```
word = 'Da'  
it = iter(word)  
next(it)
```

```
'D'
```

```
next(it)
```

```
'a'
```

```
next(it)
```

```
StopIteration                Traceback (most recent call last)  
<ipython-input-11-2cdb14c0d4d6> in <module>()  
-> 1 next(it)  
StopIteration:
```

Iterating at once with *

```
word = 'Data'  
it = iter(word)  
print(*it)
```

```
D a t a
```

```
print(*it)
```

- No more values to go through!

Iterating over dictionaries

```
pythonistas = {'hugo': 'borne-anderson', 'francis': 'castro'}  
for key, value in pythonistas.items():  
    print(key, value)
```

```
francis castro  
hugo borne-anderson
```


Iterating over file connections

```
file = open('file.txt')  
it = iter(file)  
print(next(it))
```

```
This is the first line.
```

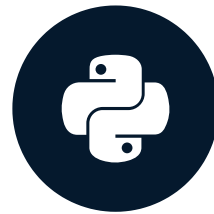
```
print(next(it))
```

```
This is the second line.
```

Let's practice!
PYTHON TOOLBOX

Playing with iterators

PYTHON TOOLBOX



Hugo Bowne-Anderson
Data Scientist at DataCamp

Using enumerate()

```
avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']  
e = enumerate(avengers)  
print(type(e))
```

```
<class 'enumerate'>
```

```
e_list = list(e)  
print(e_list)
```

```
[(0, 'hawkeye'), (1, 'iron man'), (2, 'thor'), (3, 'quicksilver')]
```

enumerate() and unpack

```
avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']  
for index, value in enumerate(avengers):  
    print(index, value)
```

```
0 hawkeye  
1 iron man  
2 thor  
3 quicksilver
```

```
for index, value in enumerate(avengers, start=10):  
    print(index, value)
```

```
10 hawkeye  
11 iron man  
12 thor  
13 quicksilver
```

Using zip()

```
avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']  
names = ['barton', 'stark', 'odinson', 'maximoff']  
z = zip(avengers, names)  
print(type(z))
```

```
<class 'zip'>
```

```
z_list = list(z)  
print(z_list)
```

```
[('hawkeye', 'barton'), ('iron man', 'stark'),  
(('thor', 'odinson'), ('quicksilver', 'maximoff'))]
```

zip() and unpack

```
avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']  
names = ['barton', 'stark', 'odinson', 'maximoff']  
for z1, z2 in zip(avengers, names):  
    print(z1, z2)
```

```
hawkeye barton  
iron man stark  
thor odinson  
quicksilver maximoff
```

Print zip with *

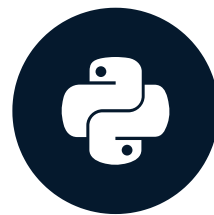
```
avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']  
names = ['barton', 'stark', 'odinson', 'maximoff']  
z = zip(avengers, names)  
print(*z)
```

```
('hawkeye', 'barton') ('iron man', 'stark')  
('thor', 'odinson') ('quicksilver', 'maximoff')
```


Let's practice!
PYTHON TOOLBOX

Using iterators to load large files into memory

PYTHON TOOLBOX



Hugo Bowne-Anderson
Data Scientist at DataCamp

Loading data in chunks

- There can be too much data to hold in memory
- Solution: load data in chunks!
- `pandas` function: `read_csv()`
 - Specify the chunk: `chunksize`

Iterating over data

```
import pandas as pd
result = []
for chunk in pd.read_csv('data.csv', chunksize=1000):
    result.append(sum(chunk['x']))
total = sum(result)
print(total)
```

4252532

Iterating over data

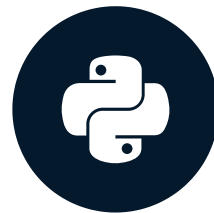
```
import pandas as pd
total = 0
for chunk in pd.read_csv('data.csv', chunksize=1000):
    total += sum(chunk['x'])
print(total)
```

4252532

Let's practice!
PYTHON TOOLBOX

Congratulations!

PYTHON TOOLBOX



Hugo Bowne-Anderson
Data Scientist at DataCamp

What's next?

- List comprehensions and generators
- List comprehensions:
 - Create lists from other lists, DataFrame columns, etc.
 - Single line of code
 - More efficient than using a for loop

Let's practice!
PYTHON TOOLBOX