PYTHON DATA SCIENCE TOOLBOX II

# Python Data Science Toolbox II

# You've learned:

- Writing custom functions
- Using custom functions in data science

# You'll learn:

- List comprehensions
  - Wrangle data to create other lists
- Iterators
  - You've encountered these before!
  - Rapidly iterate data science protocols and procedures over sets of objects

PYTHON DATA SCIENCE TOOLBOX II

# See you in the course!

PYTHON DATA SCIENCE TOOLBOX II

# Iterators in Pythonland

# Iterating with a for loop

- We can iterate over a list using a for loop

```
In [1]: employees = ['Nick', 'Lore', 'Hugo']

In [2]: for employee in employees:
   ...:     print(employee)
Nick
Lore
Hugo
```

# Iterating with a for loop

- We can iterate over a string using a for loop

```
In [1]: 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

```
In [1]: 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()

```
In [1]: word = 'Da'

In [2]: it = iter(word)

In [3]: next(it)
Out[3]: 'D'

In [4]: next(it)
Out[4]: 'a'

In [5]: next(it)
---------------------------------------------------------------
StopIteration                 Traceback (most recent call last)
<ipython-input-11-2cdb14c0d4d6> in <module>()
----> 1 next(it)
StopIteration:
```

# Iterating at once with *

```
In [1]: word = 'Data'

In [2]: it = iter(word)

In [3]: print(*it)
D a t a

In [4]: print(*it)
```

No more values to go through!

# Iterating over dictionaries

```
In [1]: pythonistas = {'hugo': 'bowne-anderson', 'francis':
'castro'}

In [2]: for key, value in pythonistas.items():
   ...:     print(key, value)
francis castro
hugo bowne-anderson
```

# Iterating over file connections

```
In [1]: file = open('file.txt')

In [2]: it = iter(file)

In [3]: print(next(it))
This is the first line.

In [4]: print(next(it))
This is the second line.
```

PYTHON DATA SCIENCE TOOLBOX II

# Let's practice!

PYTHON DATA SCIENCE TOOLBOX II

# Playing with iterators

# Using enumerate()

```
In [1]: avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']

In [2]: e = enumerate(avengers)

In [3]: print(type(e))
<class 'enumerate'>

In [4]: e_list = list(e)

In [5]: print(e_list)
[(0, 'hawkeye'), (1, 'iron man'), (2, 'thor'), (3, 'quicksilver')]
```

# enumerate() and unpack

```
In [1]: avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']

In [2]: for index, value in enumerate(avengers):
  ....:     print(index, value)
0 hawkeye
1 iron man
2 thor
3 quicksilver

In [3]: for index, value in enumerate(avengers, start=10):
  ....:     print(index, value)
10 hakweye
11 iron man
12 thor
13 quicksilver
```

# Using zip()

```
In [1]: avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']

In [2]: names = ['barton', 'stark', 'odinson', 'maximoff']

In [3]: z = zip(avengers, names)

In [4]: print(type(z))
<class 'zip'>

In [5]: z_list = list(z)

In [6]: print(z_list)
[('hawkeye', 'barton'), ('iron man', 'stark'), ('thor',
'odinson'), ('quicksilver', 'maximoff')]
```

# zip() and unpack

```
In [1]: avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']

In [2]: names = ['barton', 'stark', 'odinson', 'maximoff']

In [3]: for z1, z2 in zip(avengers, names):
  ....:     print(z1, z2)
hawkeye barton
iron man stark
thor odinson
quicksilver maximoff
```

# Print zip with *

```
In [1]: avengers = ['hawkeye', 'iron man', 'thor', 'quicksilver']

In [2]: names = ['barton', 'stark', 'odinson', 'maximoff']

In [3]: z = zip(avengers, names)

In [4]: print(*z)
('hawkeye', 'barton') ('iron man', 'stark') ('thor', 'odinson')
('quicksilver', 'maximoff')
```

PYTHON DATA SCIENCE TOOLBOX II

# Let's practice!

PYTHON DATA SCIENCE TOOLBOX II

# Using iterators for big data

# 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

```
In [1]: import pandas as pd

In [2]: result = []

In [3]: for chunk in pd.read_csv('data.csv', chunksize=1000):
   ...:     result.append(sum(chunk['x']))

In [4]: total = sum(result)

In [5]: print(total)
4252532
```

# Iterating over data

```
In [1]: import pandas as pd

In [2]: total = 0

In [3]: for chunk in pd.read_csv('data.csv', chunksize=1000):
   ...:     total += sum(chunk['x'])

In [4]: print(total)
4252532
```

PYTHON DATA SCIENCE TOOLBOX II

# Let's practice!

PYTHON DATA SCIENCE TOOLBOX II

# Congratulations!

# 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

PYTHON DATA SCIENCE TOOLBOX II

# See you in the next chapter!