

Programming for Data Analytics

Week8: Introduction to Pandas

Dr. Haithem Afli

Haithem.afli@cit.ie

[@AfliHaithem](https://twitter.com/AfliHaithem)

2018/2019





Introduction to Pandas Series and Dataframe data structures.

- Reading data into a Dataframe
- Accessing Data from a Dataframe
- Merging and Grouping Data

Pandas

- NumPy is a great tool for dealing with **numeric matrices** and vectors in Python
 - For more complex data, such as tables it is limited.
- Fortunately, when dealing with complex data we can use the **Python Data Analysis Library** (a.k.a. pandas).
- Pandas is an open source library providing high-performance, easy-to-use data structures for the Python programming language.
 - Used primarily for data manipulation and analysis.
- Resources
 - <http://pandas.pydata.org/pandas-docs/version/0.13.1/pandas.pdf>

Data Structures in Pandas

- Pandas introduces two new data structures to Python –
 - **Series**
 - **DataFrame**
- Both of which are built on top of NumPy (which means it's very fast).
- **A Series** is a one-dimensional object similar to an array, list, or column in a table.
- Pandas will assign a **labelled index** to each item in the Series.
 - By default, each item will receive an index label from 0 to N, where N is the length of the Series minus one.
- **S = Series(data, index = index)**
 - The data can be many different things such as a NumPy arrays, list of scalar values, dictionary

Series - Examples

```
import numpy as np
import pandas as pd

s1 = pd.Series(np.random.randn(5))
s2 = pd.Series(np.random.randn(5), index=['a','b','c','d','e'])
# number of indices must match number of data points

print (s1)
print (s2)
```

```
import pandas as pd

# Dictionary with annual car robberies in each Irish city
d = {'Dublin': 245, 'Cork': 150, 'Limerick': 125, 'Galway': 360,
'Belfast': 300}

# if you pass a dictionary to a series, the keys becomes the
indexes of the Series
cities = pd.Series(d)

print (cities)
```

Series

- You can use the **index to select specific items** from the Series.
 - The first print will print the entire series
 - The second will print the item associated with index 'b' (note you can access one item at time using this method)
 - The third uses **double square brackets** and prints a subset of the original series (**note it returns a independent Series object**)

```
s1 = pd.Series([1, 2, 3, 4, 5], index=['a','b','c','d','e'])  
  
print (s1)  
  
print (s1['b'])  
  
print (s1[['a', 'b']])
```

Series

- Another useful feature of a series is using boolean conditions
 - **irishCities <200** returns a Series of True/False values, which we then pass to our Series cities, returning the corresponding True items.

```
# Dictionary with annual car robberies in each Irish city  
d = {'Dublin': 245, 'Cork': 150, 'Limerick': 125, 'Galway':  
360, 'Belfast': 300}
```

```
irishCities = pd.Series(d)
```

```
print (irishCities[ irishCities <200 ])
```


```
print (type(irishCities[irishCities <200]))
```

As with NumPy, relational operators return a **separate copy** of the data. The original series and the one returned by the relational operator don't refer to the same copy of the same data.

Series

- It is also very easy to change a value within a series.

```
d = {'Dublin': 245, 'Cork': 150, 'Limerick': 125, 'Galway': 360, 'Belfast':  
300}  
irishCities = pd.Series(d)  
  
print (irishCities)  
  
irishCities["Cork"] = 180  
irishCities["Kilkenny"] = 120  
  
print (irishCities)
```

A red arrow points from the text box below to the line of code in the previous block that assigns the value 120 to the 'Kilkenny' key in the 'irishCities' series.

Similar to the syntax we
use for adding a key value
pair to a dictionary.

- What does the code below achieve?

```
d = {'Dublin': 245, 'Cork': 150, 'Limerick': 125, 'Galway': 360, 'Belfast': 300}
irishCities = pd.Series(d)

print (irishCities)

irishCities[irishCities<160] = 100

print (irishCities)
```

This code will go through the Series setting any value that is currently less than 160 to a value of 100.

When you use Boolean selection coupled with assignment it selects the entries in the existing Series object to be changed.

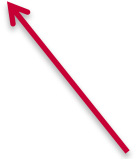
Series

- Normal NumPy mathematical operations can be completed on Series objects as well.

```
d = {'Dublin': 245, 'Cork': 150, 'Limerick': 125, 'Galway': 360, 'Belfast': 300}
irishCities = pd.Series(d)

print (irishCities*100)

print (np.square(irishCities))
```



Notice in this example we still use NumPys square method but rather than passing it a NumPy array we pass it a Series instead

Series – len and unique function

- As with all data structures we have seen so far we can use the ***len()*** function to obtain the number of values stored in a Series (this also works for a dataframe, which return the number of rows)
- Another useful function to use with a Series object is the ***unique*** function, which returns all the unique data items in a specific series object (it is returned as a NumPy array).

```
import pandas as pd
```

```
seriesA = pd.Series(['A', 'C', 'B', 'B', 'A'])
```

```
['A' 'C' 'B']
```

```
print (pd.unique(seriesA))
```

Example

- Create a Pandas Series variable to store the data depicted in the table below (we will use the names as indices and the grades as the values).
- Write code that will return a Series containing all those that failed the exam.
- Next write code that will increase any grade less than 40 by 5%

| Name | Grade |
|--------|-------|
| Jim | 78 |
| Elaine | 23 |
| Ted | 65 |
| Frank | 88 |
| Sarah | 80 |
| Tim | 33 |

Example

```
import pandas as pd
```

```
Elaine  23
Tim     33
dtype: int64
```

```
Elaine  28
Frank   88
Jim     78
Sarah   80
Ted     65
Tim     38
dtype: int64
```

It is possible to turn this Series into a one-column DataFrame with the `to_frame` method.

This method will use the Series name as the new column name:

```
>>> director.to_frame()
```

Data Frame

- A DataFrame is a data structure comprised of **rows and columns** of data.
 - It is similar to a spreadsheet or a database table.
 - You can also think of a DataFrame as a **collection of Series objects** that share an index
- To create a DataFrame out of common Python data structures, we can pass a dictionary of lists to the DataFrame constructor.
- We can also easily create a dataframe by passing it a 2D NumPy array.
- The syntax for creating a data frame is as follows:
 - ***DataFrame(data, columns=listOfColumns)***
- Using the columns parameter allows us to tell the constructor how we'd like the columns ordered.

Discussion



Thank you

Haithem.afli@cit.ie

[@AfliHaithem](#)