Introduction to pandas: Takeaways

Syntax

PANDAS DATAFRAME BASICS

- Reading a file into a dataframe:

```
f500 = pd.read_csv('f500.csv',index_col=0)
```

- Returning a dataframe's data types:

```
col_types = f500.dtypes
```

- Returning the dimensions of a dataframe:

```
dims = f500.shape
```

SELECTING VALUES FROM A DATAFRAME

- Selecting a single column:

```
f500["rank"]
```

- Selecting multiple columns:

```
f500[["country", "rank"]]
```

- Selecting the first n rows:

```
first_five = f500.head(5)
```

- Selecting rows from a dataframe by label:

```
drink_companies = f500.loc[["Anheuser-Busch InBev", "Coca-Cola", "Heineken Holding"]]

big_movers = f500.loc[["Aviva", "HP", "JD.com", "BHP Billiton"], ["rank","previous_rank"]]

middle_companies = f500.loc["Tata Motors":"Nationwide", "rank":"country"]
```

DATA EXPLORATION METHODS

- Describing a Series object:

```
revs = f500["revenues"]

summary_stats = revs.describe()
```

- Unique Value Counts for a Column:

```
country_freqs = f500['country'].value_counts()

top5_countries = country_freqs.head()
```

ASSIGNMENT WITH PANDAS

- Replacing a specific column with a new Series object:

```
f500["revenues_b"] = f500["revenues"] / 1000
```

- Replacing a specific value in a dataframe:

```
f500.loc["Dow Chemical","ceo"] = "Jim Fitterling"
```

BOOLEAN INDEXING IN PANDAS

- Filtering a dataframe down on a specific value in a column:

```
kr_bool = f500["country"] == "South Korea"

top_5_kr = f500[kr_bool].head()
```

- Updating values using Boolean filtering:

```
f500.loc[f500["previous_rank"] == 0, "previous_rank"] = np.nan

prev_rank_after = f500["previous_rank"].value_counts(dropna=False).head()
```

Concepts

- NumPy provides fundamental structures and tools that makes working with data easier, but there are several things that limit its usefulness as a single tool when working with data:
    - The lack of support for column names forces us to frame the questions we want to answer as multi-dimensional array operations.
    - Support for only one data type per ndarray makes it more difficult to work with data that contains both numeric and string data.
    - There are lots of low level methods, however there are many common analysis patterns that don't have pre-built methods.

- The **pandas** library provides solutions to all of these pain points and more. Pandas is not so much a replacement for NumPy as an extension of NumPy. The underlying code for pandas uses the NumPy library extensively. The main objects in pandas are **Series** and **Dataframes**. Series is equivalent to a 1D Ndarray while a dataframe is equivalent to a 2D Ndarray.
- Different label selection methods:

Select by Label	Explicit Syntax	Shorthand Convention	Other Shorthand
Single column from dataframe	df.loc[:,"col1"]	df["col1"]	df.col1
List of columns from dataframe	df.loc[:,["col1","col7"]]	df[["col1","col7"]]	
Slice of columns from dataframe	df.loc[:,"col1":"col4"]		
Single row from dataframe	df.loc["row4"]		
List of rows from dataframe	df.loc[["row1", "row8"]]		
Slice of rows from dataframe	df.loc["row3":"row5"]	df["row3":"row5"]	
Single item from series	s.loc["item8"]	s["item8"]	s.item8
List of items from series	s.loc[["item1","item7"]]	s[["item1","item7"]]	
Slice of items from series	s.loc["item2":"item4"]	s["item2":"item4"]	

Resources

- Dataframe.loc[]
- Indexing and Selecting Data