Calculating With Pandas: Takeaways

Syntax

- Access the value of a series for a given label:

```
ages = people['Age']
age_dexter = ages['Dexter']
```

- Access the value of a series by index:

```
ages = people['Age']
age_1 = ages[1]
```

- Get the values counts of column:

```
people['Gender'].value_counts()
```

- Get all rows where a given columns a specific values:

```
females = people[people['Gender'] == 'F]
```

- Combining boolean masks:

```
people[(people['Weight'] <= 75) & (people['Height'] <= 1.7)]
```

- Combining masks with columns selection:

```
people.loc[people['Height'] <= 1.7, 'Weight']
people.loc[people['Height'] <= 1.7, ['Age', 'Weight']]
people.loc[people['Height'] <= 1.7, 'Age':'Height']
```

- Creating and adding a parial column:

```
emails = pd.Series()
emails['Rita'] = 'rita@gmail.com'
emails['Bob'] = 'bob@hotmail.com'
people['Email'] = emails
```

Concepts

- Pandas uses `pandas.Series` objects to represent 1-dimensional data. This data structure functions as an enhanced dictionary. We can access its values using the row labels as keys, but it also benefits from the same speedup as ndarrays.
- Most of the features that we learned for NumPy translate almost literally to pandas.
- We can add a partial column to a pandas dataframe. Pandas will associate the values to the rows with matching labels. Values associated with labels that don't exist in the dataframe are discarded. The value associated with missing labels is set to `NaN`.

Resources

- Pandas series documentation