# Processing Dataframes in Chunks: Takeaways



## Syntax

- Specifying the number of rows we want each chunk of a dataframe to contain:

```
chunk_iter = pd.read_csv("data.csv", chunksize = 10000)
```

- Printing each chunk:

```
for chunk in chunk_iter:
    print(chunk)
```

- Combining pandas objects:

```
series_list = [pd.Series([1,2]), pd.Series([2,3])]
```

- Creating a GroupBy object:

```
s4 = s3.groupby(s3.index)
```

- Observing the groups in a GroupBy object:

```
for key, item in s4:
    print(key.get_group(key))
```

- Timing chunking and processing:

```
%%timeit
lifespans = []
chunk_iter = pd.read_csv("moma.csv", chunksize=250, dtype={"ConstituentBeginDate": "float",
"ConstituentEndDate": "float"},  usecols=['ConstituentBeginDate', 'ConstituentEndDate'])
for chunk in chunk_iter:
    lifespans.append(chunk['ConstituentEndDate'] - chunk['ConstituentBeginDate'])
    lifespans_dist = pd.concat(lifespans)
```

## Concepts

- We can use chunking to load in and process dataframes in chunks when working with data sets that don't fit into memory.
- Breaking a task down, processing the different parts separately, and combining them later on is an important workflow in batch processing.
- We can cut down the overall running time by only loading in the columns we're examining.

## Resources

- Batch Processing
- Documentation for the Series.groupby method