The `pandas` Package

> Adapted from a guide originally contributed by Mike Zhu (@mz888)!

> NOTE: this document references the "jeter_stats.csv" and "jeter_stats.xlsx" files, which are available for download from the data directory.

The `pandas` package provides capabilities for working with structured data, including spreadsheet-like objects called "DataFrames".

Reference:

- Pandas Website
- Pandas Docs
- Pandas Source
- `DataFrame` - like a spreadsheet
- Input and Output
- `head()` and `tail()`
- `iloc[]`
- `read_csv()`
- `iterrows()`
- `groupby()`

Installation

First install the package using Pip, if necessary:

```
pip install pandas
```

Usage

Data Frames

The Pandas `DataFrame` datatype represents a table of data, like a spreadsheet.

Creating Data Frames

We're able to transform different types of data structures (e.g. a list of lists, a dictionary of lists, etc.) into a Pandas data frame.

When using a list to create a new data frame, each entry in the list represents another row in the table:

```
import pandas as pd

my_list = [
  [1, "a"],
  [2, "b"],
  [3, "c"]
]

df = pd.DataFrame(my_list)

df # columns will be numeric by default
#>    0  1
#> 0  1  a
#> 1  2  b
#> 2  3  c

df.columns = ["number", "letter"] # possible to override column names

df
#>    number letter
#> 0       1      a
#> 1       2      b
#> 2       3      c
```

When using a dictionary to create a new data frame, each key in the dictionary represents a column with its own values:

```
import pandas as pd

my_dict = {
    "number": [1,2,3],
    "letter": ["a", "b", "c"]
}

df = pd.DataFrame(my_dict)

df
#>    number letter
#> 0       1      a
#> 1       2      b
#> 2       3      c
```

It's also possible to process a spreadsheet or CSV file into a data frame:

```
import pandas as pd

stats = pd.read_csv("/path/to/jeter_stats.csv")
# ... OR ...
stats = pd.read_excel("/path/to/jeter_stats.xlsx")

stats
#>     year  games  at_bats  runs  hits  walks
#> 0   1995     15       48     5    12      3
#> 1   1996    157      582   104   183     48
#> 2   1997    159      654   116   190     74
#> 3   1998    149      626   127   203     57
#> 4   1999    158      627   134   219     91
#> 5   2000    148      593   119   201     68
#> 6   2001    150      614   110   191     56
#> 7   2002    157      644   124   191     73
#> 8   2003    119      482    87   156     43
#> 9   2004    154      643   111   188     46
#> 10  2005    159      654   122   202     77
#> 11  2006    154      623   118   214     69
#> 12  2007    156      639   102   206     56
#> 13  2008    150      596    88   179     52
#> 14  2009    153      634   107   212     72
#> 15  2010    157      663   111   179     63
#> 16  2011    131      546    84   162     46
#> 17  2012    159      683    99   216     45
#> 18  2013     17       63     8    12      8
#> 19  2014    145      581    47   149     35
```

Using Data Frames

Row Operations

Inspect the first and last few rows, respectively:

```
stats.head(3)
#>    year  games  at_bats  runs  hits  walks
#> 0  1995     15       48     5    12      3
#> 1  1996    157      582   104   183     48
#> 2  1997    159      654   116   190     74

stats.tail(3)
#>     year  games  at_bats  runs  hits  walks
#> 17  2012    159      683    99   216     45
#> 18  2013     17       63     8    12      8
#> 19  2014    145      581    47   149     35
```

Count rows:

```
stats.count()
```

Iterate through rows:

```
for index, row in stats.iterrows():
    print(row["year"])
```

Reference a specific row by its index (e.g. 0):

```
stats.iloc[0]
```

Convert any row to a dictionary:

```
stats.iloc[0].to_dict()
```

Column Operations

Reference a specific column:

```
stats["games"]
#> 0       15
#> 1      157
#> 2      159
#> 3      149
#> 4      158
#> ...
```

Perform some column aggregations:

```
stats["games"].sum() #> 2747

stats["games"].min() #> 15

stats["games"].max() #> 159
```

```
stats["games"].mean() #> 137.35

stats["games"].median() #> 153.5
```

Filter rows matching some given condition:

```
stats[stats["games"] > 150]

#>      year  games  at_bats  runs  hits  walks
#> 1    1996    157      582   104   183     48
#> 2    1997    159      654   116   190     74
#> 4    1999    158      627   134   219     91
#> 7    2002    157      644   124   191     73
#> 9    2004    154      643   111   188     46
#> 10   2005    159      654   122   202     77
#> 11   2006    154      623   118   214     69
#> 12   2007    156      639   102   206     56
#> 14   2009    153      634   107   212     72
#> 15   2010    157      663   111   179     63
#> 17   2012    159      683    99   216     45
```

Calculate new ad-hoc columns like "batting average" and "on-base percentage":

```
stats["average"] = stats["hits"] / stats["at_bats"]

stats["obp"] = (stats["hits"] + stats["walks"]) / stats["at_bats"]

stats
#>      year  games  at_bats  runs  hits  walks   average       obp
#> 1    1996    157      582   104   183     48  0.314433  0.396907
#> 2    1997    159      654   116   190     74  0.290520  0.403670
#> 4    1999    158      627   134   219     91  0.349282  0.494418
#> 7    2002    157      644   124   191     73  0.296584  0.409938
#> 9    2004    154      643   111   188     46  0.292379  0.363919
#> 10   2005    159      654   122   202     77  0.308869  0.426606
#> 11   2006    154      623   118   214     69  0.343499  0.454254
#> 12   2007    156      639   102   206     56  0.322379  0.410016
#> 14   2009    153      634   107   212     72  0.334385  0.447950
#> 15   2010    157      663   111   179     63  0.269985  0.365008
#> 17   2012    159      683    99   216     45  0.316252  0.382138
```

Exporting Data Frames

Convert a data frame to list of dictionaries, each representing a row in the data frame:

```
stats.to_dict("records") # "records" is a specific parameter of the to_dict()
function, not a characteristic of the underlying data
```

Convert a data frame to list of lists, each representing a row in the data frame:

```
stats.values.tolist()
```

Save a data frame back to a spreadsheet or CSV file:

```
stats.to_csv("/path/to/jeter_stats_v2.csv")
# ... OR ...
stats.to_excel("/path/to/jeter_stats_v2.xlsx")
```