**Data Wrangling with pandas Cheat Sheet**

**Syntax – Creating DataFrames**

```
Invert DataFrame:

df = pd.DataFrame(
    {'a': [4, 5, 6],
     'b': [7, 8, 9],
     'c': [10, 11, 12]},
    index = [4, 5, 10])
```

**Reshaping Data – Change the layout of a data set**

```
Gather columns into rows:

df.melt(df)
```

```
Spread rows into columns:

df.pivot(columns='var', values='val')
```

```
Append rows of DataFrames:

df.concat([df1, df2])
```

```
Append columns of DataFrames:

df.concat([df1, df2], axis=1)
```

**Subset Observations (Rows)**

```
Extract rows that meet logical criteria:

df[df.Length > 7]
```

```
Randomly select fraction of rows:

df.sample(frac=0.5)
```

```
Select rows by position:

df.iloc[10:20]
```

```
Select and order top n entries:

df.nlargest(n, 'value')
```

```
Select and order bottom n entries:

df.nsmallest(n, 'value')
```

**Subset Variables (Columns)**

```
Select multiple columns with specific names:

df["width", "length", "species"]
```

```
Select single column with specific name:

df['width'] or df.width
```

```
Select columns whose name matches regular expression regex:

df.filter(regex='regex')
```

**Logic in Python (and pandas)**

```
< Less than

>= Greater than

equals

Not equal to

Group membership

is NaN

is not NaN

is not NaN

Logical and, or, or xor, any, all
```

**Method Chaining**

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
         .rename(columns={'variable': 'var',
                          'value': 'val'}))
```

**Data Wrangling with pandas Cheat Sheet**

**Tidy Data – A foundation for wrangling in pandas**

Tidy data complements pandas’s vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.
## Summarize Data

- `df['w'].value_counts()`: Count number of rows with each unique value of variable
- `len(df)`: Number of rows in DataFrame.
- `df['w'].unique()`: Number of distinct values in a column.
- `df.describe()`: Basic descriptive statistics for each column (or GroupBy)

Pandas provides a large set of summary functions that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

- `sum()`: Sum values of each object.
- `count()`: Count non-NA/null values of each object.
- `median()`: Median value of each object.
- `quantile([0.25, 0.75])`: Quantiles of each object.
- `std()`: Standard deviation of each object.
- `var()`: Variance of each object.
- `min()`: Minimum value in each object.
- `max()`: Maximum value in each object.
- `mean()`: Mean value of each object.
- `unique()`: Number of rows with each unique value of variable

## Handling Missing Data

- `df.dropna()`: Drop rows with any column having NA/null data.
- `df.fillna(value)`: Replace all NA/null data with value.

## Make New Columns

- `df.assign(Area=lambda df: df.Length*df.Height)`: Add single column.
- `pd.qcut(df.col, n, labels=False)`: Bin column into n buckets.

Pandas provides a large set of vector functions that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

- `min(axis=1)`: Element-wise min. (Series).
- `max(axis=1)`: Element-wise max. (Series).
- `cumsum()`: Cumulative sum. (Vector)
- `cummax()`: Cumulative max. (Vector)
- `cummin()`: Cumulative min. (Vector)
- `cumprod()`: Cumulative product. (Vector)

## Group Data

- `df.groupby(by=“col”)`: Return a GroupBy object, grouped by values in column named "col".
- `df.groupby(level=“ind”)`: Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

- `size()`: Size of each group.
- `agg(function)`: Aggregate group using function.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

- `shift(1)`: Copy with values shifted by 1.
- `rank(method='dense')`: Ranks with no gaps.
- `rank(method='min')`: Ranks. Ties get min rank.
- `rank(pct=True)`: Ranks rescaled to interval [0, 1].
- `rank(method='first')`: Ranks. Ties go to first value.

- `shift(-1)`: Copy with values lagged by 1.
- `cumsum()`: Cumulative sum. (Vector)
- `cummax()`: Cumulative max. (Vector)
- `cummin()`: Cumulative min. (Vector)
- `cumprod()`: Cumulative product. (Vector)

## Windows

- `df.expanding()`: Return an Expanding object allowing summary functions to be applied cumulatively.
- `df.rolling(n)`: Return a Rolling object allowing summary functions to be applied to windows of length n.

## Plotting

- `df.plot.hist()`: Histogram for each column
- `df.plot.scatter(x='w', y='h')`: Scatter chart using pairs of points

## Combine Data Sets

- `adfpd.merge(bdf, how='inner', on='x1')`: Join matching rows from bdf to adf.
- `ydfpd.merge(zdf, how='outer', on='x1')`: Join data. Retain all rows in both sets.

## Set-like Operations

- `pd.merge(ydf, zdf, how='outer')`: Rows that appear in either or both ydf and zdf (Union).
- `pd.merge(ydf, zdf, how='left_only')`: Rows that appear in both ydf and zdf (Intersection).

---