Pandas Cheatsheet

Dataframe

Load dataset

```python
pd.read_csv(csv_file)
pd.read_excel(excel_file)
```

Set index

```python
df.set_index(keys=col_name, inplace=True)
```

Get basic information about the dataframe

```python
df.head()
df.describe()
df.info()
df.columns
```

Get size of dataset

```python
df.shape
```

Drop rows

Drop the rows where at least one of the elements is missing

```python
df.dropna()
```

Drop the rows where all of the elements are missing

```python
df.dropna(how='all')
```

Drop the rows where columns in subset are missing

```python
df.dropna(subset=[col_name])
```

Pandas Cheatsheet

Drop rows

Drop the rows at index_number

```python
df.drop(index_number)
```

Drop the rows that do not meet condition

```python
df.drop(df[<condition>].index)
ex: df.drop(df[df['High Income'] == False].index)
```

Delete a column

Delete a column

```python
del df[col_name]
```

Replace NaN values

Replace any NaN values with new_value in the Dataframe

```python
df.fillna(new_value, inplace=True)
```

Replace any NaN values with new_value in col_name

```python
df[col_name].fillna(new_value, inplace=True)
```

Sort dataset

Sort dataset by col_name

```python
df.sort_values(by=col_name)
```

Sort dataset by col_name in a descending order

```python
df.sort_values(by=col_name, ascending=False)
```

Pandas Cheatsheet

Unique values

Unique values in col_name

```python
df[[col_name]].drop_duplicates()
df[[col_name_1, col_name_2]].drop_duplicates()
```

Number of unique values in col_name

```python
df[col_name].nunique()
```

Select columns

Select specific columns using index location

```python
df.iloc[rows_index, columns_index]
```

Select specific columns by column names

```python
df[[col_name]]
```

Filter dataset

Filter dataset based on criteria

```python
df.loc[criteria]
ex: df.loc[df['Age'] > 30]
```

Filter dataset based on multiple criteria

```python
df.loc[(condition1) & (condition2)]
df.loc[(condition1) | (condition2)]
```

Pandas Cheatsheet

Filter dataset

Filter dataset where col_name is in list_of_col_values

```python
df.loc[df[col_name].isin([list_of_col_values])]
ex: df.loc[df['State'].isin(['California','Texas','Florida'])]
```

Python

Filter dataset by index value

```python
df[df.index == 'SFO']
```

Python

Create new columns

String manipulation

```python
df[new_col_name] = df[col_name_1] + ' ' + df[col_name_2]
```

Python

Numerical column

```python
df[new_col_name] = df[col_name_1] + df[col_name_2]
```

Python

Aggregate functions

Basic aggregate functions

```python
df[col_name].sum()
df[col_name].mean()
df[col_name].min()
df[col_name].max()
df[col_name].count()
```

Python

Group by - you may put in multiple columns in groupby

- Group by col_name
- Apply aggregate functions

```python
df.groupby([col_name]).mean()
df.groupby([col_name_1, col_name_2, col_name_3]).mean()
```

Python

Pandas Cheatsheet

Group by - you may put in multiple columns in groupby

- Group by col_name_1
- Apply aggregate functions
- Sort by col_name_2

```python
df.groupby([col_name_1]).mean().sort_values(by=col_name_2)
```

- Group by col_name_1
- Apply aggregate functions
- Display col_name_2

```python
df.groupby([col_name_1])[col_name_2].mean()
```

- Group by col_name_1
- Apply aggregate functions
- Display col_name_2
- Sort by the result of aggregate functions

```python
df.groupby([col_name_1])[col_name_2].mean().sort_values(ascending=False)
```

- Group by col_name
- Apply multiple aggregate functions in agg()
- Display col_name_2

```python
df.groupby([col_name_1])[col_name_2].agg(['sum','count'])
```

Join dataframes

Merge

```python
DataFrame.merge(right, how='inner', on=None, left_on=None, right_on=None,
left_index=False, right_index=False, sort=False, suffixes='_x', '_y', copy=True,
indicator=False, validate=None)
```

Merge example

```python
pd.merge(df1, df2, how='right', on=['First Name', 'Last Name'])
```

Pandas Cheatsheet

Join dataframes

Join

```
DataFrame.join(other, on=None, how='left', lsuffix='', rsuffix='', sort=False)
```
Python

Join example

```
df1.join(df2, lsuffix='_1', rsuffix='_2')
```
Python

Series

Create a series

Scalar

```
scalar_series = pd.Series(5, index =[0, 1, 2, 3, 4, 5])
```
Python

List

```
array = np.array(['a','b','c','d'])
pd.Series(array)
```
Python

List with indices

```
array = np.array(['a','b','c','d'])
pd.Series(array,index=[1,2,3,4])
```
Python

```
names = np.array(['Daisy', 'Matt', 'Kelly', 'Mike'])
cities=["Atlanta", "San Francisco", "New York", "Seattle"]

pd.Series(names, index=cities)
```
Python

Pandas Cheatsheet

Series

Create a series

Dict

```python
aDict = {'Apple':3, 'Banana':5, 'Cherry': 2, 'Mango': 13, 'Peach': 10}
pd.Series(aDict)
```

Sort a Series

By index

```python
s = pd.Series(aDict)
s.sort_index()
```

By value

```javascript
s = pd.Series(aDict)
s.sort_values()
```

Descending order

```python
s.sort_index(ascending=False)
s.sort_values(ascending=False)
```

Access an element

By index number

```python
array = np.array(['a','b','c','d','e','f','g','h'])
s = pd.Series(array)

s[0]
s[:3]
```

Pandas Cheatsheet

Access an element

By index label

```python
aDict = {'Apple':3, 'Banana':5, 'Cherry': 2, 'Peach': 10}
s = pd.Series(aDict)

s['Apple']
s['Peach']
```

Binary operations

Add

```python
s1.add(s2)
s1 + s2
```

Subtract

```python
s1.sub(s2)
s1 - s2
```

Multiply

```python
s1.mul(s2)
s1 * s2
```