DataFrame Cheat Sheet

e.g

df	C1	C2	C3
R1	10	A	800
R2	20	B	900

Creating a DataFrame

- pd.DataFrame(<input>)
- **Using Arrays**
  <input> = [[10, 'A', 800], [20, 'B', 900]]
- **Using Dictionary**
  <input> = {"c1": [10, 20], "c2": ['A','B']}
- **Using Series**
  s1 = Series([10, 20]), s2 = Series(['A','B'])...
  dict = { 'C1' : s1, 'C2' : s2, 'C3' : s3}
  pd.DataFrame(dict, columns = <list>)

- Importing data - Creating from files : *fname*
  - csv : *df = pd.read_csv(fname)*
  - tsv : *df = pd.read_table(fname) # delimited text*
  - xls : *df = pd.read_excel(fname)*
  - Json : *df = pd.read_json(fname / json_string)*
  - html: *df = pd.read_html(url)*
- Exporting data - Writing to a file:
  - Use corresponding *to_<<format>>* func.
  - e.g *to_csv(), to_excel(), to_json()....*

- Creating from Database : *dbname*
  - *conn = sl.connect(<<string point to database>>)*
  - *pd.read_sql(query, conn)*
- Writing to a Database:
  - *pd.to_sql(table_name, conn)*

Viewing / Inspecting Data

- *df.head(n)* # retrieve first n rows
- *df.tail(n)* # retrieve last n rows
- *df.sample(frac)* # retrieve fraction of rows (0.5)
- *len(df)* # number of rows in dataframe
- *df.shape* # retrieve number of rows, cols
- *df.info()* # details like index, datatypes.
- *df.describe()* # retrieve summary statistics

- Retrieve Unique Values and counts
  df. value_counts(dropna=False)
  df[<collist>]. value_counts(dropna=False)
  df.[col]nunique() # num of distinct values in col

- Retrieve column / column values
- *df[<col>]* # retrieve col values as Series
- *df[[<collist>]]* # retrieve col values – dataframe
- *df.colname*

- Retrieve Row Values included sorted / top-n.
- *df.loc[n]* # Select rows by Index
- *df.iloc[n]* # Select rows by Position
  - e.g. *df.iloc[1,:] # first row all columns*
- *df[cond] # condition based filtering (bool values)*
- *df.sort_values(<<col>>, ascending = False)*
- *df.rank(method = <<options>>)*
- *df.nlargest(n)* # Select and order top n entries
  df.nsmallest(n) # Select and order bottom n

Data Summaries

- Functions to operate on cols, groupby…
  - ***sum()***
  - ***count()***
  - ***min(), max()***
  - ***mean(), median()***
  - ***var()*** # Variance
  - ***std()*** # Standard deviation
  - ***quantile[0.25, 0.5]***
  - ***.apply(func)***
  - ***corr()*** # correlation between cols
  - *cumsum()* # cumulative sum
  - *cumprod()* # cumulative sum

- Grouping Data with groupby
  - *df .groupby(by = <<collist>>)*
    # returns *groupby object* values in col(s)
  - *.size ()* # size of each group
  - *.agg (fn)* # aggregate group using fn / function

- Exporting data - Writing to a file:
  - Use corresponding *to_<<format>>* func.
  - e.g *to_csv(), to_excel(), to_json()....*

Data Manipulation / DQ

- Rename columns
  df.columns = [column name list]
  df.rename(columns = {'old':'new'})

- Check for Nulls and handle
  - *df.isnull()* # returns boolean array
  - *df.notnull()*
  - *df.dropna()* # drop all rows with nulls
  - *df.dropna(axis =1)* # drop all cols w/ nulls
  - *df.fillna(x)* # replace null values with x
  - *df.fillna(np.mean)*
  - s.replace(a, b) # replace values with b.
  - *df.shift(-1)* # copy with values lagged

- Row / Indexes
  - *df. set_index('column name')* # change ind
  - *df.reset_index()* # new column of index

- Combining data / adds
  - *df1.append(df2)* # rows of df1 add to df2
  - *pd.concat(df1, df2)* # cols of df1 to df2

# Joins cols of df1 with df2 matching on col. How specifies type of sql-join
  - *df1.join(df2, on = "col", how = "<options>")*