**PANDAS CHEAT SHEET**

by Simon Schmid. Work in progrss, without any guarantees. Spotted a mistake? Mail [here](mailto:simon@netwings.ch).

**BASICS**

**Install**

**pip install pandas**  *- only do this once*

**Libraries to import**

**import pandas as pd**  *- basic package*

**import numpy as np**  *- need this for NaN*

**import datetime**  *- to deal with time*

**Help**

**?**  *- get help on a method ( e.g. type pd.head? )*

**DATA IN AND OUT**

**Constructors**

**pd.Series(list)**  *- construct a Series from a list* ([reference](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html))

* list: ['value1', 'value2']

**s = pd.Series(dict)**  *- construct a Series from a dict* ([reference](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html))

* dict: ({"key1": value1, "key2": value2})

**df = pd.DataFrame(listoftdicts)**  *- Create a DF from a list of dictionaries* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html))

* listofdicts: [{'field1': value1, 'field2': value2}, {'field1': value3, 'field2': value4}]

**df = pd.DataFrame(dictoflists)**  *- Create a DF from a dictionary of lists* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html))

* dictsoflists: {'field1': [value1, value3], 'field2': [value2, value4}

**Data in**

**df = pd.read\_csv("file.csv")**  *- Create a DF from a CSV file* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html))

* nrows=59  *- to only get a number of rows*
* na\_values=["string1", "string2", ...]  *- to specify values to ignore*
* dtype=str  *- treat everything as strings*
* sep="\t"  *- for tab-delimited files*
* error\_bad\_lines=False  *- Ignore errors (i.e. too many fields on a line)*
* header=None  *- use if csv has no header-row*
* names=["id", "cat"]  *- specify column names to be used*

**df = pd.read\_excel("file.xlsx")**  *- Create a DF from an XLS file* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_excel.html))

* sheetname="name"  *- the name of the sheet inside excel*
* skiprows=n  *- skip the first n rows*
* names=list  *- use these column names*

**pd.read\_sql(query, conn)**  *- execute a SQL-query on a given connection* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_sql.html))

* index\_col="column"  *- the column to be used as index*

**Data out**

**df.to\_csv("file.csv")**  *- save a DF into the specified csv* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_csv.html))

* index=False  *- don't save the index column*

**df.to\_dict()**  *- save dataframe as dictionary* ([reference](https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.DataFrame.to_dict.html))

* orient="records"/"list"/...  *- way of constructing the dictionary*

**df.to\_json()**  *- save as json-string (almost like to\_dict)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_json.html))

* orient="records"/"index"  *- orientation of the json*

**s.to\_frame()**  *- converts a series into a dataframe* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.to_frame.html))

**DATA TYPES**

**NaN**

**NaN**  *- Placeholder for missing data*

**np.nan**  *- code for nan (need to import numpy as np)*

**SELECTING STUFF**

**Select whole table**

**df**  *- represents whole table*

**Select columns**

**df.field1**  *- fetch only one column*

**df["field1"]**  *- alternative notation*

**df[["field1", "field2"]]**  *- fetch several columns*

**df[condition]**  *- only fetch rows where condition is true*

* condition = df['field' == value]  *- an example*

**Select rows**

**df.head(n)**  *- only first n rows* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.head.html))

**df.tail(n)**  *- only last n rows* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.tail.html))

**df.loc[index]**  *- get row at particular index* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html))

**df.iloc[integer]**  *- treat index as if it was a range of integers* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.iloc.html))

**Group data fields**

**df.groupby("field1")**  *- to initialize grouped output - need field & aggregate function* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html))

* use like: df.groupby("field1")["field2"].function()

**df.groupby(["field1", "field2"])**  *- groupby on two levels*

**DESCRIBE AND SUMMARIZE**

**Properties of the dataframes**

**df.index**  *- A list of row-indices (usually numbers)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Index.html))

**df.columns**  *- A list of column names* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.columns.html))

**df.dtypes**  *- A list of column datatypes* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dtypes.html))

**df.shape**  *- A tuple specifying (rows, columns)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.shape.html))

**df.values**  *- A matrix of the table without headers and row indices* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.values.html))

**Aggregate Statistics**

**.max()**  *- maxiumum* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.max.html))

* axis=1  *- use all these functions not column-wise but row-wise*

**.min()**  *- minimum* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.min.html))

**.mean()**  *- mean* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.mean.html))

**.std()**  *- standard deviation* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.std.html))

**.sum()**  *- sum* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sum.html))

**.count()**  *- count* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.count.html))

**.size()**  *- useful for double-groupbys, similar to value\_counts()* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.size.html))

**MODIFY DATAFRAMES**

**df.copy()**  *- copy the dataframe (instead of just referencing it)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.copy.html))

**Modify the index**

**df.set\_index("field")**  *- change the index-column to "field1"* ([reference](https://pandas.pydata.org/pandas-docs/version/0.21/generated/pandas.DataFrame.set_index.html))

* inplace=True  *- make the changes on the object, not a copy*

**df.rename\_axis("Name")**  *- Rename the index column. Use 'None' to delete the index name* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.rename_axis.html))

* inplace=True  *- make the changes on the object, not a copy*

**Modify columns**

**df.insert(pos, "field1", values)**  *- insert new column "field1" at pos with values* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.insert.html))

**df.pop("field1")**  *- delete column "field1"* ([reference](http://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.DataFrame.pop.html))

**df.assign(newfield = df["field1"] ... )**  *- assign values to new column (original remains)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.assign.html))

**df.rename(columns=dict)**  *- rename columns, using {'old1': 'new1', 'old2': 'new2'} as dict* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.rename.html))

* inplace=True  *- make the changes on the object, not a copy*

**Modify rows**

**df.append(series/dataframe)**  *- adds the row, returns new object* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.append.html))

**df.drop(df[condition].index)**  *- delete rows from table based on condition* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop.html))

* inplace=True  *- make the changes on the object, not a copy*

**Modify data structure**

**df.pivot()**  *- transform long data into wide data* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html))

* index='field1'  *- the column to be used as the new index column*
* columns='field2'  *- the column which has the values that will make up the new columns*
* values='field3'  *- the colunn containing the values*

**df.melt()**  *- transform wide data into long data* ([reference](https://pandas.pydata.org/pandas-docs/version/0.21/generated/pandas.melt.html))

**df.unstack()**  *- transform groupby-subrows into columns (useful to chart stacked bars)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.unstack.html))

**df.transpose()**  *- switches row and columns over the whole dataset* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.transpose.html))

**df.T**  *- shorthand for dr.transpose()*

**Sort data globally**

**df.sort\_index()**  *- sort not by values but by index* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_index.html))

**df.sort\_values("field1")**  *- sort values* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_values.html))

* ascending=False  *- in descending order*
* na\_position="first"/"last"  *- position of NaN values*

**COMBINE DATAFRAMES**

**df.merge(df2)**  *- merge dataframe with other dataframe* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.merge.html))

* on="field"  *- fieldname(s) to match (if they have same name)*
* left\_on="df1-field"  *- fieldname to match on left side*
* right\_on="df2-field"  *- fieldname to match on right side*
* left\_index=True  *- whether to use the index as the left-side match field*
* right\_index=True  *- whether to use the index as the left-side match field*
* how="inner/left/right/outer"  *- just like in SQL*

**df.join(df2)**  *- join a dataframe (with identical number of rows) to another, horizontally* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html))

**pd.concat([df1,df2])**  *- adds all the dataframes in the list, vertically* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.concat.html))

* axis=1  *- add horizontally, not vertically*
* ignore\_index=True  *- construct new index, don't use existing one*

**MODIFY DATA GLOBALLY**

**Deal with NaNs**

**pd.isnull()**  *- Built-in function to test for null on any value* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.isnull.html))

**pd.notnull()**  *- same but opposite* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.notnull.html))

**df.dropna()**  *- get rid of NaNs, optional: in a subset* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html))

* subset="field1"  *- only apply on subset*
* inplace=true  *- make the changes on the object, not a copy*
* how="all"  *- only drop rows where all fields are NaN*

**df.fillna(value)**  *- replace NaN's with other value* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.fillna.html))

* inplace=true  *- make the changes on the object, not a copy*

**Deal with duplicates**

**df.drop\_duplicates()**  *- gets rid of duplicate vlalues* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop_duplicates.html))

* subset="field"  *- only consider certain fields (or list of fields)*
* keep="first/last/False"  *- which of the values to keep*
* inplace=True  *- make the changes on the object, not a copy*

**df.duplicated()**  *- The duplicates (inverse of drop\_dupclicated())* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.duplicated.html))

* subset="field"  *- only consider certain fields (or list of fields)*
* keep="first/last/False"  *- which of the values to keep*

**Various**

**df.round({'field1': n, 'field2': m})**  *- round the numbers in particular columns* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.round.html))

**df.dot(df2)**  *- dot product of two dataframes* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dot.html))

**df.update(df2)**  *- Update values in df with no-Nan values from df2* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.update.html))

**DEAL WITH INDIVIDUAL DATA FIELDS (I.E. SERIES)**

Many of these functions can be used on whole dataframes as well.

**Filter fields**

**df["field1"].isnull()**  *- returns true if null* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.isnull.html))

**df["field1"].notnull()**  *- opposite* ([reference](https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.DataFrame.notnull.html))

**df["field1"].isin(["str1", "str2"])**  *- returns true if field1 equals a value in the list* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.isin.html))

**~df["field1"].isin(["str1", "str2"])**  *- returns true if field1 doesn't equal a value in the list*

**df["field1"].str.contains("str")**  *- returns true if field1 contains the string str* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.str.contains.html))

* na=False  *- Avoid Error for NaN values*
* regex=True/False  *- by default, regex can be included*
* case=True/False  *- case sensitive or not, default True*

**Aggregate summaries over fields**

**df["field1"].describe()**  *- displays max, min, mean, etc* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.describe.html))

**df["field1"].max()**  *- calculate max, etc. also: mean, min, ...* ([reference](https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.Series.max.html))

**df["field"].value\_counts()**  *- frequency of each value, in tabular form* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.value_counts.html))

* normalize=True  *- in percentages*
* dropna=False  *- include NaN's*
* ascending=True  *- Sort in inverse order*

**df["field"].unique()**  *- get a list of unique (distinct) values* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.unique.html))

**pd.get\_dummies(df["field"])**  *- Based on unique values in a field, create a set of dummy columns* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html))

* prefix="prefix"  *- Prefix to use before using unique values as column headers*
* drop\_first=True

**Mathematical modifications**

**df.rolling(n, on="column")**  *- returns the rolling average of "column" as a DF* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.rolling.html))

* min\_periods=n  *- set number of periods to average over*

**df['field1"].pct\_change()**  *- calculates %-change between period t and t+1 (on series or df)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.pct_change.html))

**df['field1"].agg(['func1', 'func2'])**  *- applies aggregate function like 'mean' etc. to column and spits out a dataframe* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.agg.html))

**Data modifications**

**df["field1"].astype(int)**  *- convert to a type (int, str, float)* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.astype.html))

* errors="ignore"  *- ignore errors*

**df["field1"].replace("str1", "str2")**  *- replace (like excel, whole cell) (use str.replace for parts inside the string)*([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.replace.html))

* dictionary  *- uses key-value pairs in the dictionary to replace multiple values*
* regex=True  *- use regex to find instances of str1*

**df["field1"].str.extract(regex)**  *- extracts a regex from the field* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.str.extract.html))

* expand=True  *- force return of a dataframe instead of a series*
* .dropna()  *- to drop the NA values*

**Assign field values dynamically**

**df.loc[cond, "field"] = "value"**  *- sets "field" = "value" in all rows where "cond" is True* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html))

**df.apply(function)**  *- applies some function to the dataframe* ([reference](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.apply.html))

* axis = 1  *- tells the function to treat the df in ROWS. default: COLUMNS*

**DEAL WITH TIME**

**Data Conversion**

**pd.to\_datetime(df.column)**  *- Turn a string into a datetime. without args, leaves format unchanged* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.to_datetime.html))

* format="format"  *- specify the format (e.g. "%Y-%m-%d")* ([formats](http://strftime.org/))

**.strftime('format')**  *- Turn a datetime into a formatted string*

**df['field1'].apply(lambda (t): t.strftime('format'))** *- to transform a datetime col into a string* ([reference](https://docs.python.org/3/library/time.html#time.strftime))

**Extract datetime info**

**df['field1'].year**  *- get the year from a date-formatted column* ([reference](https://docs.python.org/2/library/datetime.html#datetime-objects))

**df['field1'].month**  *- ---the month*

**df['field1'].day**  *- ---the day of the month*

**df['field1'].dayofweek**  *- ---the day of the week*

**Timedelta**

**td = datetime.timedelta()**  *- get a representation for a time interval* ([reference](https://docs.python.org/2/library/datetime.html#timedelta-objects))

* days=n  *- specify number of days*

**td.days**  *- return number of days in a timedelta object*

**td.years**  *- return number of days in a timedelta object*

**Filter date columns**

**df['YYYY']**  *- select rows from that year*

**df['YYYY':'YYYY']**  *- select range of years*

**Mathematical modifications**

**df.resample('rule')**  *- aggregate data so some specific time interval* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.resample.html)), ([rules](https://stackoverflow.com/questions/17001389/pandas-resample-documentation/17001474*-%2017001474))

**df.rolling(n)**  *- aggregate with n neighbors. chain with .mean(), .sum() or other* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.rolling.html))

**Other date stuff**

**datetime.today()**  *- Today's date, as a datetime object*

**DISPLAY OPTIONS FOR JUPYTER NOTEBOOKS**

**pd.set\_option("optionName", value)**  *- Change the behavior of displayed content in notebooks* ([reference](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.set_option.html))

* "display.max\_rows"  *- the number of rows of a DataFrame*
* "display.max\_columns"  *- The number of columns of a DataFrame*
* "display.max\_colwidth"  *- The number of chars inside a column*
* "display.float\_format"  *- something like "{:,.0f}".format*
* etc.

**pd.options.display.max\_rows**  *- to display the current settings*