**Python**

Use two asterisks and a question mark like np.\*load\*? To get an overview of all functions that contain that word:

A screenshot of a computer

Description automatically generated

If you have a = [1, 2, 3] and then write b = a, then you are not *copying* a to b, but *creating a second reference* to [1, 2, 3]. Like here:

A screenshot of a computer

Description automatically generated A diagram of a chemical formula

Description automatically generated

Use isinstance(a, int) to check whether an object is of a specific object type:  
A screenshot of a computer

Description automatically generated

Use iter() to check whether an object is iterable (e.g. you can perform a loop on it):

A screenshot of a computer code

Description automatically generated

**Importing parts of a module**

A screenshot of a computer program

Description automatically generated

**Binary operations**

A screenshot of a computer

Description automatically generated

**Note that == is not the same as is**

Take a = 2, b = a, and c = list(a). Both a == b and a == c will return True, because all of the objects are equal to value 2. But only a is b will return True, because b refers to a and is not a *separate copy* like c. The list() function always creates a separate copy (a new list).

**Python scalar types (data types)**

A screenshot of a computer

Description automatically generated None, str, bytes, float, bool, int

**Working with dates**

Use from datetime to import types like datetime, date or time.

Use datetime() to create a date with time:

A screenshot of a computer

Description automatically generated

Use date() to return date and time() to return time:

A screenshot of a phone

Description automatically generated

The strftime method formats a datetime as a string:

A white rectangular box with red and blue text

Description automatically generated A screenshot of a computer code

Description automatically generated

**Replace parts of dates with 0, e.g. minutes and/or seconds:**

A screenshot of a computer

Description automatically generated

**for loops with continue and break**

Use continue to continue the iteration (skip the value)

A screenshot of a computer code

Description automatically generated A screenshot of a computer program

Description automatically generated

To stop the iteration use break:

A screenshot of a computer program

Description automatically generated

The break keyword only terminates the innermost for loop; any outer for loops will continue to run:

A screenshot of a computer

Description automatically generated

A while loop:

A screenshot of a computer

Description automatically generated

Use pass in a loop in blocks where no action is to be taken:

A screenshot of a computer code

Description automatically generated

**Built-In Data Structures, Functions, and Files**

Tuple, list, and dictionary are some of the most frequently used sequence types.

**Tuples**

If you have a tuple like values = 1, 2, 3, 4, 5 and want to assign only the first two values a name and other values are not important you can make use of \*rest or \*\_. Name after asterisk can be rest or \_ or anything else.

A screenshot of a computer

Description automatically generated

Count the number of occurrences in a tuple using a.count():

A screenshot of a computer

Description automatically generated

**Lists**

The list() function can be used to materialize an iterator or a generator:

A screenshot of a computer

Description automatically generated

**List concatenation** can be done using + or .extend(). Using .extend() is preferable due to computational efficiency.

A screenshot of a computer

Description automatically generated

**Slicing in Python example:**

A diagram of numbers and letters

Description automatically generated

A screenshot of a computer

Description automatically generated

Use [::2] to select every second value or [::-1] to inverse a list:

A screenshot of a computer

Description automatically generated

**Dictionary**

Use .pop to delete a value in a dictionary and return it as a variable:

A screenshot of a computer code

Description automatically generated

Use functions .keys(), .values(), and .items() to iterate (or select) over keys, values, and both keys & values in a dictionary:

A screenshot of a computer

Description automatically generated

Update a dictionary using .update():

A screenshot of a computer

Description automatically generated

Iterating over a dictionary (create a dictionary from two lists):

A white background with black text

Description automatically generated

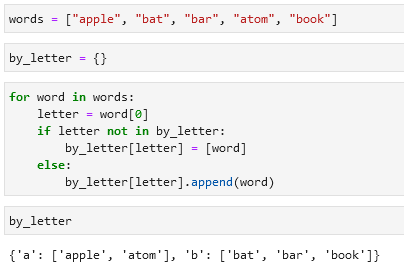
Create a dictionary from a dict():

A screenshot of a computer program

Description automatically generated

**To sort a list of values by e.g. its first letter:**

Option 1: Option 2 (preferred):

 A screenshot of a computer program

Description automatically generated

Or option 3 (preferred):

A screenshot of a computer

Description automatically generated

**Valid keys and values in dictionaries**

* Values can be any Python object.
* Keys have to be immutable objects like scalar types (int, float, string) or tuples (all the objects in the tuple need to be immutable).

To check whether a value can be used as a key use hash() (check for hashability):

A screenshot of a computer

Description automatically generated

**Set**

A set is an unordered collection of unique elements.

Two ways to create a set:

A screenshot of a computer

Description automatically generated

Set operations:

A screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generated

**Sequence functions**

**Enumerate**

A close up of words

Description automatically generated

**zip**

zip “pairs” up the elements of a number of lists, tuples, or other sequences to create a list of tuples:

A screenshot of a computer code

Description automatically generated

The number of elements it produces is determined by the shortest sequence:

A screenshot of a computer

Description automatically generated

A common use of zip() is simultaneously iterating over multiple sequences:

A screenshot of a computer code

Description automatically generated

**reversed**

Iterates over the elements of a sequence in reverse order:

A screenshot of a computer

Description automatically generated

**List, Set, and Dictionary Comprehensions**



=

A white background with black text

Description automatically generated

**Example of a list comprehension:**

A screenshot of a computer code

Description automatically generated

**Dictionary comprehension:**

A close-up of a text

Description automatically generated

**Set comprehension:**



A screenshot of a computer

Description automatically generated

Alternative to set comprehension is the map() function:

A person standing in front of a white rectangular object

Description automatically generated A screenshot of a computer code

Description automatically generated

**Dictionary comprehension:**

A screenshot of a computer

Description automatically generated

**Nested list comprehension:**

Example 1:

A screenshot of a computer code

Description automatically generated

Example 2:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Functions**

Example of a function:

A screenshot of a computer code

Description automatically generated

A screenshot of a phone

Description automatically generated

**Cleaning text in a list:**

re, .strip(), .sub(), .title()

A screenshot of a computer program

Description automatically generated

**Lambda functions**

A screenshot of a computer

Description automatically generated

**Generators**

Generator example:

A screenshot of a computer program

Description automatically generatedyield

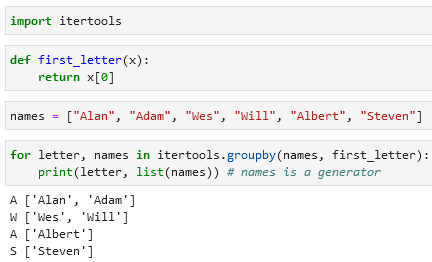
**Generator expressions**

A screenshot of a computer program

Description automatically generated

**intertools module**

Example with a intertools.groupby() function:



Some useful functions from the itertools module:

A screenshot of a computer

Description automatically generated

chain(\*iterables), combinations(iterable, k), permutations(iterable, k), groupby(iterable[,keyfunc]), product(\*iterables, repeat = 1)

**Files and the operating system**

Open a file:

A close-up of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

Two possibilities to close a file:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

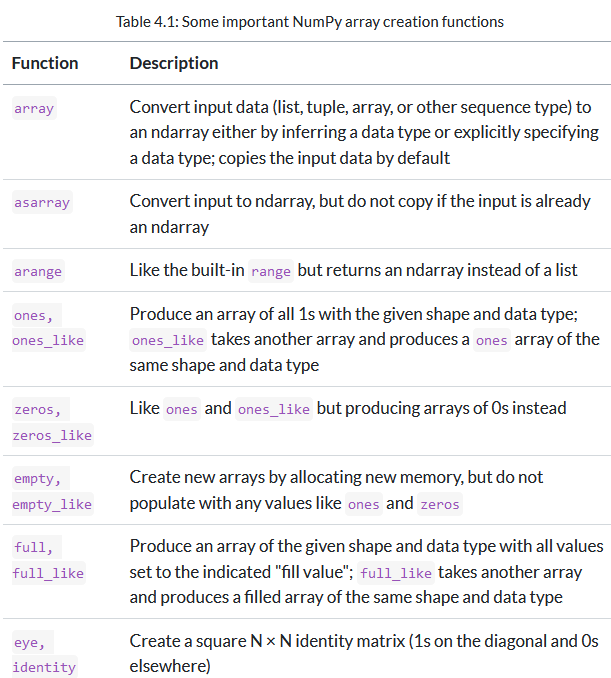
Description automatically generated

A screenshot of a computer program

Description automatically generated

**Numpy**

Functions for creating an NumPy array:



np.array, np.asarray, np.arange, np.ones, np.ones\_like, np.zeros, np.zeros\_like, np.empty, np.empty\_like, np.full, np.full\_like, np.eye, np.identity

If not specified, the data type will be float64 (in many cases).

**NumPy slicing**

A screenshot of a puzzle

Description automatically generated

**Boolean indexing**

Imagine every value in the *names* array is a representative of the *data* array. So value in *names* with *index 0* is a value in *data* with *index 0* and so on. If we want to select all of the values in *data* that correspond to *“Bob”* in *names* you can do it this way:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Note that both arrays should be of the same lengths (e.g. in this example both arrays have 7 values).

If you want to select everything but except a specific value you can use != or ~():

A screenshot of a computer program

Description automatically generated

To select AND / OR use & for AND and | for OR:

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Fancy indexing**

To select specific rows in a particular order use a list:

A screenshot of a computer

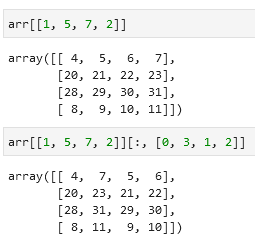
Description automatically generated

Using two lists for index will return values that are found using these “coordinates”. In the example below 4 is returned after coordinate 1, 0 and 23 is returned after coordinate 5 and 3 etc.

A screenshot of a computer code

Description automatically generated

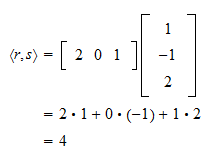
Another example:



**Transposing arrays**

Calculating inner matrix product using np.dot():

A screenshot of a computer program

Description automatically generated 

@ is an alternative to np.dot():

A screenshot of a computer

Description automatically generated

Also an option df.swapaxes(0, 1):

A screenshot of a computer code

Description automatically generated

**Pseudorandom number generation**

Use the np.random.standard\_normal() function and module (random) to generate pseudorandom numbers:

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a white and purple box

Description automatically generated

**Universal functions**

Use np.modf(df) to return the fraction and the number separately:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Conditional logic as array operations**

You can make a loop to select values from *xarr* when a value is True in *cond* and select values from *yarr* when it is False. A more efficient alternative is the np.where(condition, df1, df2) function:

A screenshot of a computer program

Description automatically generated

Or you can replace all positive values with a certain number (in the example below with 2) and all negative values with a different number (below -2):

A screenshot of a computer program

Description automatically generated

**Basic array statistical methods**

A screenshot of a table

Description automatically generated

sum, mean, std, var, min, max, argmin, argmax, cumsum, cumprod

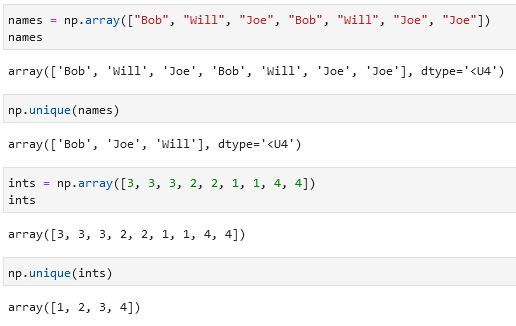
Examples like cumsum():

A screenshot of a computer program

Description automatically generated

**Unique and other set logic**

Remove duplicates and sort them using np.unique(df):



Python alternative (which is more inefficient):

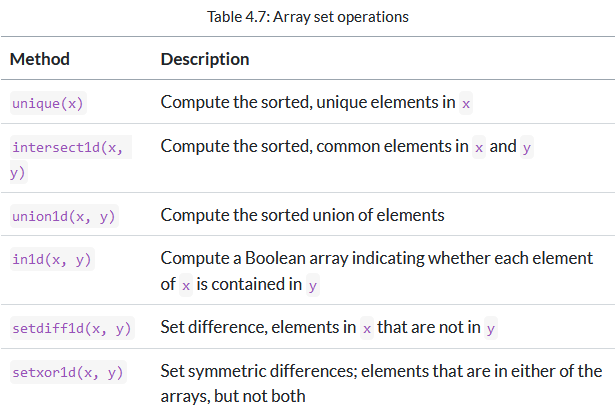
A close-up of a computer screen

Description automatically generated

Check whether some values are in an array using np.in1d():

A screenshot of a computer

Description automatically generated



Unique, intersect1d, union1d, in1d, setdiff1d, setxor1d

**Saving and loading files in NumPy**

Use np.save(), np.load(), np.savez(), np.savez\_compressed() to save and load files.

A screenshot of a computer

Description automatically generated

Same with comments:

A screenshot of a computer program

Description automatically generated

**Linear algebra functions**

A screenshot of a computer

Description automatically generated

Diag, dot, trace, det, eig, inv, pinv, qr, svd, solve, lstsq

**Pandas**

Transform a DataFrame into a NumPy array using df.to\_numpy():

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Some index methods and properties**

A screenshot of a computer

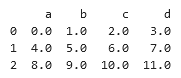
Description automatically generated

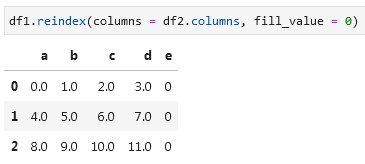
**The .reindex() function will perform a join of values present in your Series based on the current and the new indexes:**

A screenshot of a computer program

Description automatically generated

Example with fill\_value:

df1:  A screenshot of a table

Description automatically generated 

**Create a DataFrame using np.arange() and .reshape():**

A screenshot of a computer code

Description automatically generated

**The .reindex() function arguments:**

A screenshot of a computer

Description automatically generated

**If you want to change the position of index/columns of a DataFrame you can use df.loc:**

A screenshot of a computer

Description automatically generated

**Difference between df.loc and df.iloc:** df.iloc selects on the index *(0, 1, 2, 3 etc.)* while df.loc selects numerical values in the index like *1, 3, 2* will look for rows with these indexes:

A screenshot of a computer

Description automatically generated A screenshot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated

**Indexing with DataFrame**

A screenshot of a computer program

Description automatically generated

**Adding two DataFrames**

Use df1.add(df2, fill\_value = 0) to add one DataFrame to the second:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Flexible arithmetic methods between two DataFrames:**

A screenshot of a math method

Description automatically generated

**Function application and mapping**

Apply format to a DataFrame using frame.applymap():

A screenshot of a table

Description automatically generated

A screenshot of a computer

Description automatically generated

**Sorting and ranking**

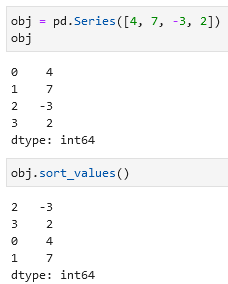
To sort an index use df.sort\_index():

A screenshot of a computer code

Description automatically generated A screenshot of a computer

Description automatically generated

To sort a Series by its values use obj.sort\_values():



To sort a DataFrame pass a column name(s) which should be used to sort:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Tie-breaking methods with rank:**

A screenshot of a computer

Description automatically generated

**Summarizing and computing descriptive statistics**

Options for reduction methods:

A screenshot of a computer

Description automatically generated

Return the index of the maximum/minimum value using df.indxmax() or df.indxmin():

A screenshot of a computer

Description automatically generated

Cumulative sum by rows using df.cumsum():

A screenshot of a math test

Description automatically generated

A screenshot of a math test

Description automatically generated

**A full list of summary statistics and related methods:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Unique values, value counts, and membership methods**

A screenshot of a computer

Description automatically generated

**Data loading, storage, and file formats**

Text and binary data loading functions in pandas:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Import a csv using pd.read\_csv() with different settings:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Some pd.read\_csv() function arguments:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Reading Excel files**

Use pd.read\_excel() or pd.ExcelFile() to open (read) Excel files:



**Reading SQL**

A screenshot of a computer

Description automatically generated

**Data cleaning and preparation**

**Handling missing data**

dropna, fillna, isna, notna

Examples with df.fillna():

A screenshot of a calculator

Description automatically generated A screenshot of a calculator

Description automatically generated A screenshot of a computer

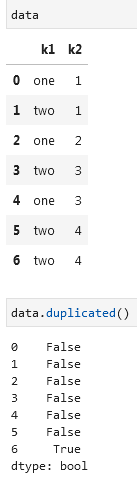
Description automatically generated

Function arguments of df.fillna():

A screenshot of a computer

Description automatically generated

Removing duplicates using df.duplicated() and df.drop\_duplicates():

A screenshot of a math book

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Perform a LEFT JOIN on a column (add a column, merge two tables) using .map():**

A screenshot of a menu

Description automatically generated A screenshot of a computer

Description automatically generated

**Replace values using df.replace():**

A screenshot of a math test

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Renaming axis indexes**

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Binning, grouping, categorization**

You can bin using pd.cut(df, bins) and then get a GROUP BY using df.value\_counts():

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

pd.qcut(df, bin).value\_counts():

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Working with outliers**

If you want to get values that are more than a certain positive number and less than a certain negative number you can use .abs() > 3:

A screenshot of a computer

Description automatically generated

**To select all rows having a value exceeding 3 or –3, you can use the any method:**

A screenshot of a table

Description automatically generated A screenshot of a table with numbers

Description automatically generated

A screenshot of a computer

Description automatically generated

Replace all values above 3 and below -3 with a 3/-3:

A screenshot of a computer

Description automatically generated

Example of np.sign() (it returns 1 for positive values, -1 for negative, and 0 for 0):

A screenshot of a computer

Description automatically generated

**To randomly reorder columns/rows in a DataFrame use np.random.permutation():**

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Permutation of columns:

A screenshot of a computer

Description automatically generated A screenshot of a computer

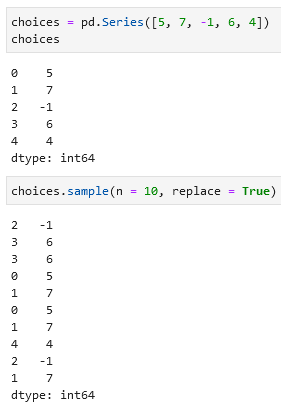
Description automatically generated

Select a certain number of random rows using df.sample(n = ):

A screenshot of a computer

Description automatically generated

If you want to select more random rows than there are in a Series/DataFrame, you can add replace = True, it will allow returning duplicate rows:



**Computing indicator/dummy variables**

You have a column named *City* with 3 variables: *London, Paris, Berlin.* You can unpivot this column into columns like *London?, Paris?, Berlin?* where in each column a 0 will mean *No* and 1 will mean *Yes.* To do so use pd.get\_dummies(df[“column\_name”]):

A screenshot of a computer

Description automatically generated

Merge these column back with the rest of the table:

A screenshot of a computer code

Description automatically generated

Unpivoting using str.get\_dummies():

A screen shot of a movie

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a movie

Description automatically generated

Combining pd.get\_dummies() and pd.cut():

A screenshot of a computer

Description automatically generated

**String manipulation (text manipulation/edition)**

A screenshot of a computer

Description automatically generated

A black text on a white background

Description automatically generated

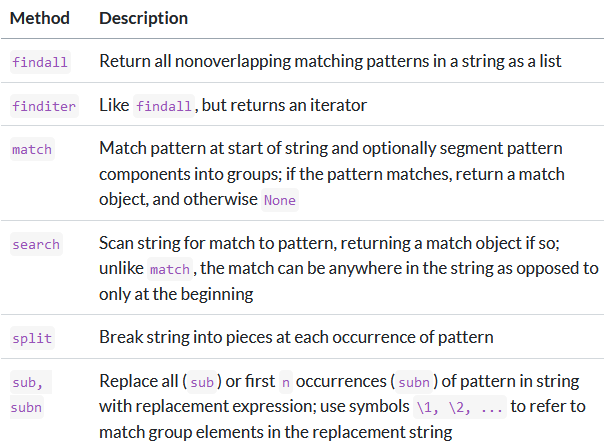
Some examples with .split(), .index(), .find(), .count(), .replace():

A screenshot of a computer code

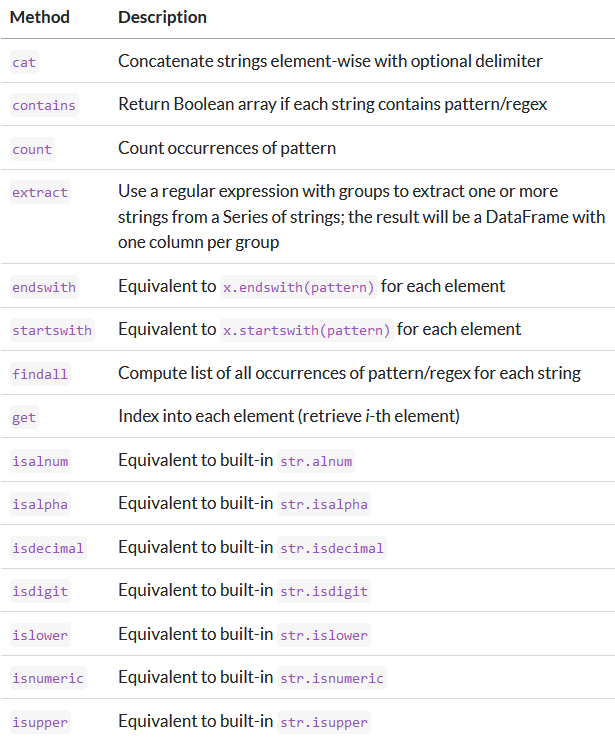
Description automatically generated A screenshot of a computer

Description automatically generated

**Regular expressions (regexs) methods**



**String functions in pandas**



A screenshot of a computer program

Description automatically generated

**Categorical methods (save memory and computation time)**

A screenshot of a computer

Description automatically generated

**Data wrangling: JOIN, combine, and reshape**

A screenshot of a computer

Description automatically generated

To index this Series:

A screenshot of a computer

Description automatically generated

You can also unpivot a table using df.unstack() or pivot a table using df.stack():

A screenshot of a data

Description automatically generated A screenshot of a computer

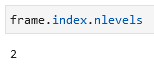
Description automatically generated

Hierarchical indexing for DataFrames:

A screenshot of a computer

Description automatically generated

Check how many levels does an index have with df.index.nlevels:



Creating a MultiIndex:

A screenshot of a computer code

Description automatically generated

To rearrange an index use df.swaplevel(index1, index2):

A screenshot of a computer

Description automatically generated

To sort an index use df.sort\_index() (with df.sort\_index() or df.sort\_index(level = 0) performance is much better):

A screenshot of a computer

Description automatically generated

Take a column/two columns and create an index out of them using df.set\_index(column1, column2). Columns will be removed from the DataFrame, but you can keep them by adding drop = False:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

To move the hierarchical index levels into the columns use df.reset\_index():

A screenshot of a number

Description automatically generated

**Combining and merging datasets**

Merge two DataFrames using df.merge(df1, df2, on = “column\_name”):

A screenshot of a cell phone

Description automatically generated A screenshot of a computer code

Description automatically generated

Perform an inner-join:

A screenshot of a computer

Description automatically generated

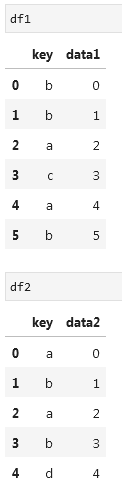
Perform a full-join / outer join:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Example of a left-join:

 A screenshot of a computer

Description automatically generated

Types of joins in the how = argument:

A screenshot of a computer

Description automatically generated

Join on two columns:

A screenshot of a computer

Description automatically generated

When you're joining columns on columns, the indexes on the passed DataFrame objects are discarded. If you need to preserve the index values, you can use reset\_index to append the index to the columns.

A screenshot of a computer

Description automatically generated

df.merge() function arguments:

A screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generated

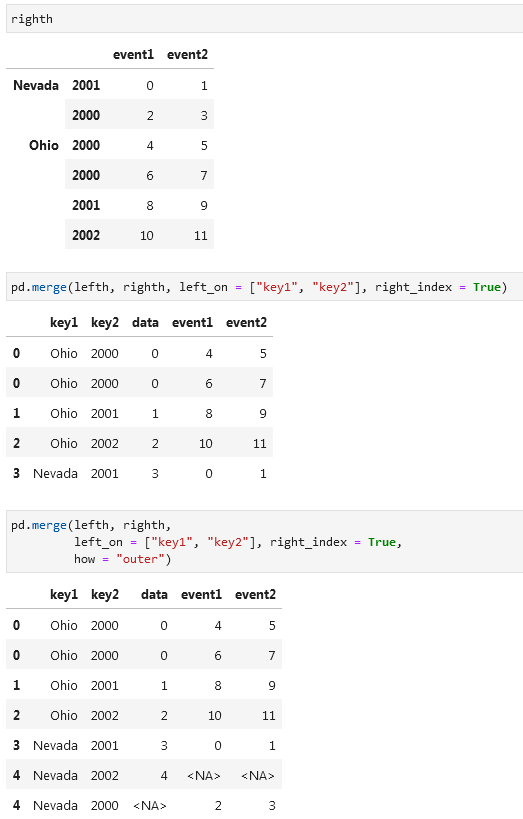
To merge on index make use of right\_index = True or left\_index = True:

A screenshot of a computer

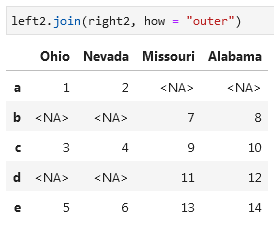
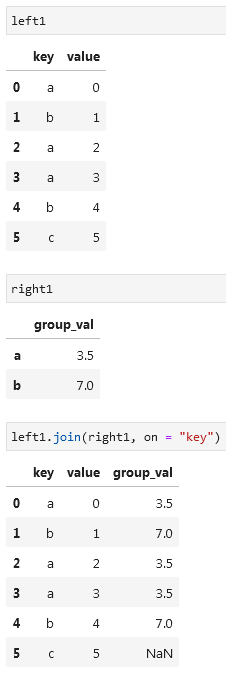
Description automatically generated

A screenshot of a computer

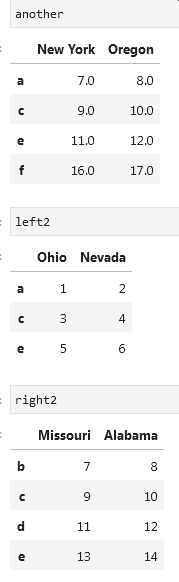
Description automatically generated



You can also use df1.join(df2, how =) to perform a join:

Merge several DataFrames:

 A screenshot of a computer

Description automatically generated

**Appending tables (concatenation)**

You can append tables using np.concatenate:

A screenshot of a computer program

Description automatically generated

Use pd.concat([df1, df2, df3]) to append tables:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Create hierarchical index using keys:

A screenshot of a computer

Description automatically generated

If you combine tables along the axis = 1 (horizontally and not vertically) keys will be the names of the columns:

A screenshot of a computer program

Description automatically generated

Creating a hierarchical index in a DataFrame:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

If you want to ignore the index in both tables you can add ignore\_index = True:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

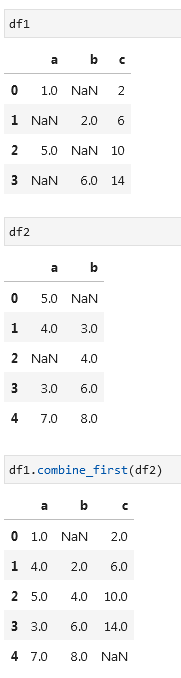
pd.concat function arguments:

A screenshot of a computer program

Description automatically generated

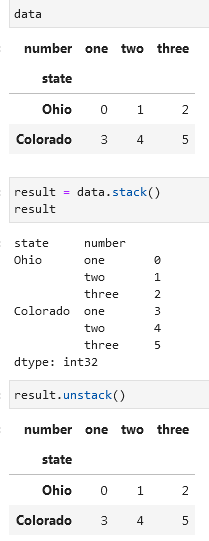
If you have two tables and want to select data from the first table and only select values from the second table when the values in the first table are blank, then you can use np.where(pd.(isna(table1), table2, table1). Also see df1.combine\_first(df2):

A screenshot of a computer

Description automatically generated 

**Reshaping and pivoting**

Use df.stack() or df.unstack() to pivot/unpivot. If you want to select a specific index

 A screenshot of a computer

Description automatically generated

Add blank values with stacking and unstacking:

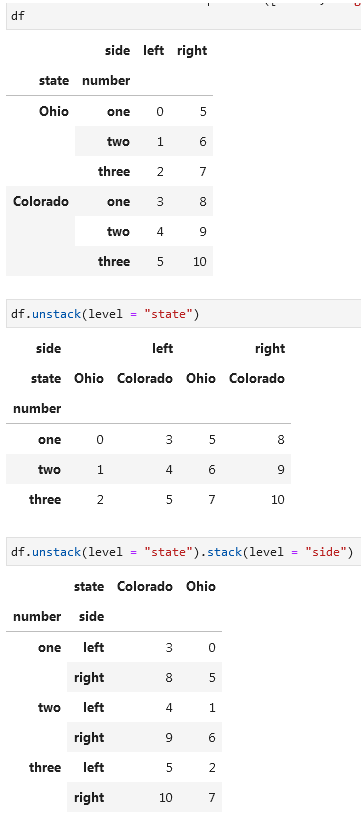
A screenshot of a computer code

Description automatically generated

A screenshot of a computer

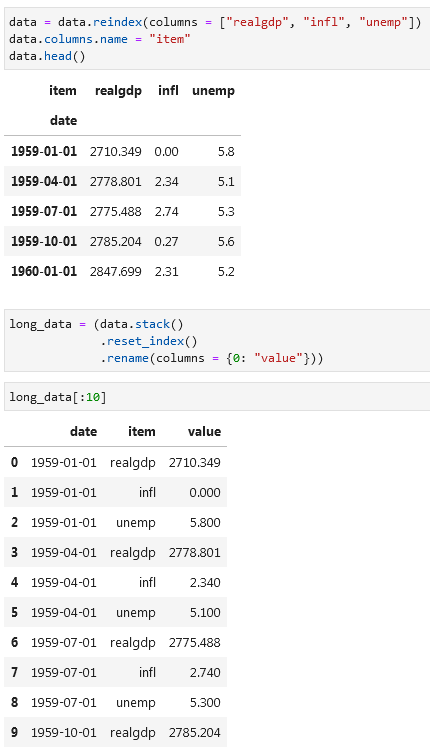
Description automatically generated

Indicate which level (index) to use for stacking/unstacking with level =:



Converting columns with year and quarter information into a day-month-year index (pop() means select column and drop it from the DataFrame) using pd.PeriodIndex() and df.to\_timestamp(“D”):





A screenshot of a computer

Description automatically generated

To unpivot other columns you can use pd.melt(df, id\_vars = “column\_that\_will\_not\_be\_unpivoted”):

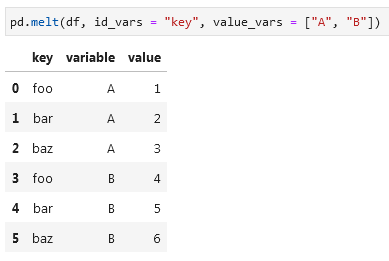
A screenshot of a computer code

Description automatically generated

To return the table back to its original form:

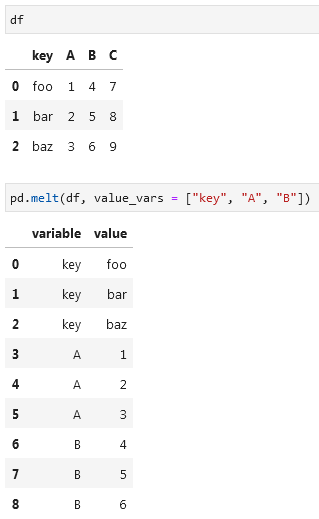


You can filter columns that you want to use in the unpivoting process using value\_vars. Here column C was dropped:



You can also drop your group identifier:

A screenshot of a computer

Description automatically generated 

**Plotting and visualization**

matplotlib.pyplot.subplots options:

A screenshot of a computer

Description automatically generated

**Annotations and drawing on a subplot**

A screenshot of a computer code

Description automatically generated

A graph showing the price of a stock market

Description automatically generated

**Saving plots to file**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Matplotlib configuration (font, size)**

For more extensive customization and to see a list of all the options, matplotlib comes with a configuration file matplotlibrc in the matplotlib/mpl-data directory. If you customize this file and place it in your home directory titled .matplotlibrc, it will be loaded each time you use matplotlib.

A screenshot of a computer

Description automatically generated

**Plotting with pandas**

A screenshot of a computer program

Description automatically generated

A screenshot of a table

Description automatically generated

If you want to get a pivot table between two variables you can use pd.crosstab():

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**To normalize a table (so that values in a column are equal to 100% / 1):**

A screenshot of a computer

Description automatically generated

**Data aggregation and group operations**

Hadley Wickham coined the term *split-apply-combine* for describing group operations.

Using the example below you group your data first according to its indexes, then you sum the values inside these groups, and finally you combine the results of these groups.

A diagram of a number

Description automatically generated

GROUP BY examples with df[“col\_with\_values”].groupby(df[“group\_on\_col”]):

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

To group on two columns:

A screenshot of a computer

Description automatically generated

You can also unpivot this result:

A screenshot of a computer

Description automatically generated

You can also use external indexes to perform a GROUP BY, like in the example below:

A screenshot of a computer

Description automatically generated

A useful GroupBy method is .size() (like a pivot that returns the number of values in each group/category):

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Alternative to .size() can be .count():

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Any missing values are excluded from the result by default. This can be disabled using the dropna = False feature:

A screenshot of a computer program

Description automatically generated

**Selecting a column/columns**

A screenshot of a computer

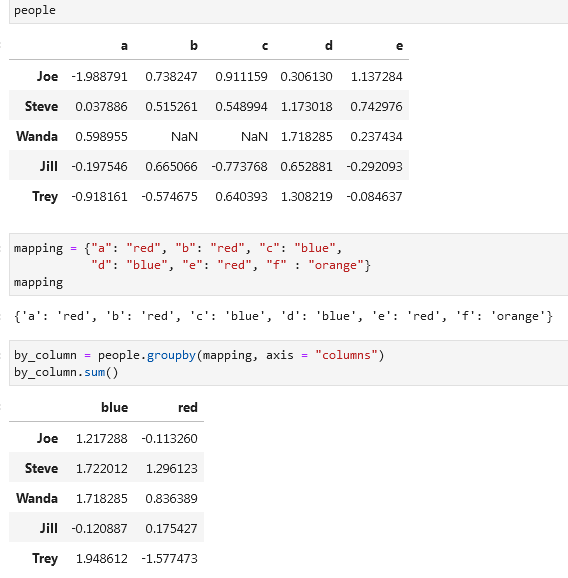
Description automatically generated

A screenshot of a computer

Description automatically generated

**Grouping with dictionaries and series**

You can pass an index column that will be used for grouping as a separate dictionary/series:

 A screenshot of a computer

Description automatically generated

**Grouping with functions**

A screenshot of a computer

Description automatically generated

**Group by index levels**

A screenshot of a computer

Description automatically generated

**Data aggregation**

Optimized groupby methods:

A screenshot of a computer

Description automatically generated

Returning two smallest values from a groupby using .nsmallest(2):

A screenshot of a computer

Description automatically generated

To use your own aggregations write .agg(function):

A screenshot of a computer code

Description automatically generated

Methods like .describe() can also be used on grouped objects:

A screenshot of a computer

Description automatically generated

**Aggregating on multiple columns:**

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

If you want to return custom names make use of tuples with column names first and functions second:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Example with applying 3 functions on two selected columns:

A screenshot of a computer

Description automatically generated

Another example:

A screenshot of a computer

Description automatically generated

**Applying different functions on different columns is possible using dictionary. Two examples:**

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

To return aggregated data without row indexes use as\_index = False (better) or .reset\_index() on the result:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

You can return the top X rows per each category using your custom function and a groupby(“column”).apply(function):  
A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Examples with .describe():

A screenshot of a computer

Description automatically generated

If you don’t want to create an index-column with the grouped value you can type group\_keys = False:

A screenshot of a table

Description automatically generated A screenshot of a computer

Description automatically generated

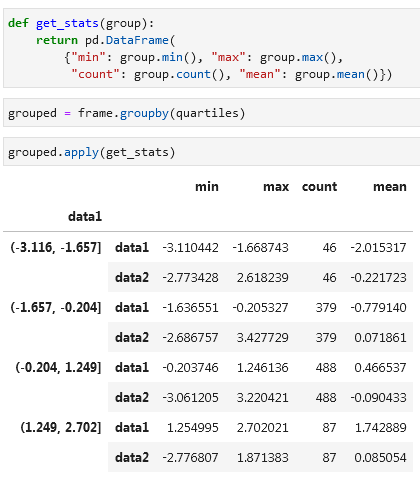
**Quantile and bucket analysis**

Use pd.cut(df, number\_of\_buckets) to put your values into buckets:

A screenshot of a computer

Description automatically generated

These buckets can be now used for further analysis:



Alternatively using .agg() instead of .apply(function):

A screenshot of a computer

Description automatically generated

Using the pd.qcut() (equal-size buckets based on sample quantiles; pass labels = False to obtain just the quartile indices instead of intervals) formula instead of pd.cut() (equal-length buckets):

A screenshot of a computer

Description automatically generated

**Filling Missing Values with Group-Specific Values**

You can use s.fillna(s.mean()) to fill N/A values with a mean of a Series:

A screenshot of a computer

Description automatically generated

Another example with filling group-specific values:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Alternatively you can use a dictionary:

A screenshot of a computer program

Description automatically generated

**Random Sampling and Permutation**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Group Weighted Average and Correlation**

A screenshot of a computer

Description automatically generated

Next to .describe() you can use df.info() to get an overview of a DataFrame:

A screenshot of a computer

Description automatically generated

**Use .pct\_change() to calculate change to the previous value:**

A screenshot of a computer code

Description automatically generated A screenshot of a computer

Description automatically generated

**Calculate growth on a year basis:**

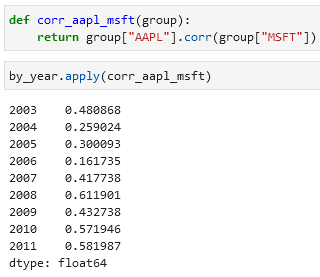
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Calculate intercolumn correlations:**



**Group Transforms and “Unwrapped” GroupBys**

You can use .transform() to replace values with group averages:

A screenshot of a cell phone

Description automatically generated A screenshot of a computer program

Description automatically generated

Alternatively, you can just pass a built-in aggregation function:

A screenshot of a test

Description automatically generated

Using built-in aggregation functions is better in terms of computing.

Other examples like multiplying by 2, ranking:

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated

Example of normalization using .transform, .apply and .transform using built-in functions:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Pivot tables**

To create a pivot table use df.pivot\_table(index = , values = ). By default the pivot table returns the values as mean.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

You can add totals using margins = True:

A screenshot of a computer

Description automatically generated

You can use count (will exclude null values) or len (will not exclude null values) to count the number of values in a DataFrame. Use fill\_value = 0 to fill the null values with e.g. 0.

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

pivot\_table options:

A screenshot of a computer

Description automatically generated

**Cross-tabulations or crosstab**

Use pd.crosstab() to create a pivot:

A screenshot of a computer

Description automatically generated

Example of a crosstab vs a pivot\_table:

A screenshot of a table

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Working with time series**

Calculating the delta (difference between two dates) using datetime.now():

A screenshot of a computer

Description automatically generated

Calculating the delta using timedelta() (e.g. adding 12 days to a date):

A screenshot of a computer

Description automatically generated

Types in the datetime module:

A screenshot of a computer

Description automatically generated

Converting between string and datetime using .strftime():

A screenshot of a computer

Description automatically generated

datetime format specification (ISO C89 compatible):

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

You can use pd.to\_datetime() to parse many different kinds of date representation:

A screenshot of a phone

Description automatically generated

Locale-specific date formatting:

A screenshot of a computer

Description automatically generated

Check if an index is unique using df.index.is\_unique:

A screenshot of a computer

Description automatically generated

Indexing on this table will return a value or an index & value:

A screenshot of a computer

Description automatically generated

If you have nonunique timestamps and you want to aggregate data on this index add level = 0:

A screenshot of a computer program

Description automatically generated

**Generating date ranges**

Use pd.date\_range(date1, date2) to generate a date range between two dates:

A screenshot of a computer code

Description automatically generated

Add the number of periods to generate a date range starting from a certain date or ending with a certain date:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

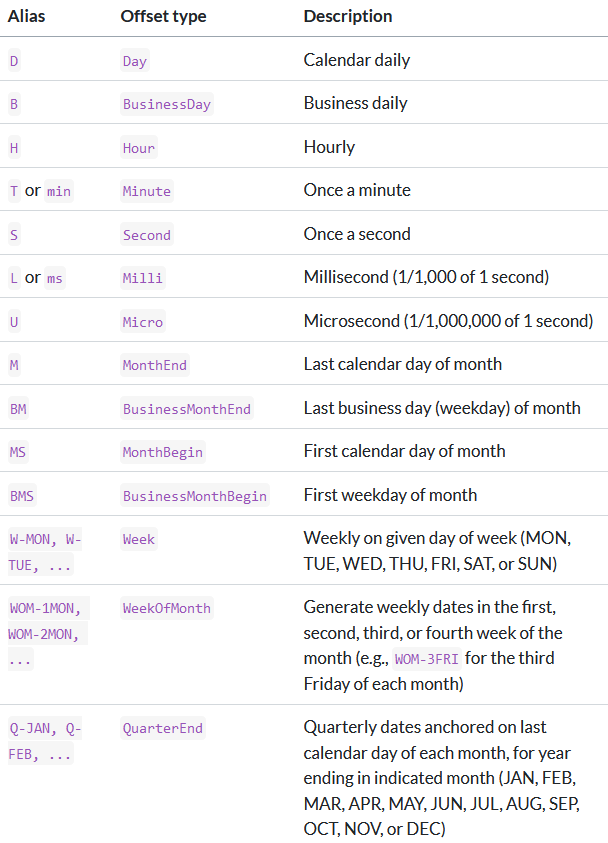
Description automatically generated

To get the last business day of each month add freq = "BM":

A screenshot of a computer code

Description automatically generated

Base time series frequencies (not comprehensive):



A screenshot of a calendar

Description automatically generated

If you want to drop time in your input date you can add normalize = True:

A screenshot of a computer

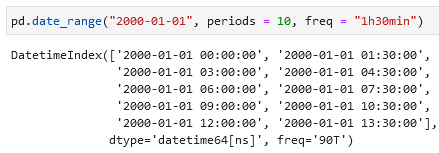
Description automatically generated

If you want to get an overview of dates that have a 4-hours difference between them add freq = "4H":

A screenshot of a computer code

Description automatically generated

Or every 1,5 hours:



If you want to get every third Friday of a month:

A screenshot of a computer

Description automatically generated

How to create a Series using dates as index:

A screenshot of a computer code

Description automatically generated

You can shift values forward or backward using .shift() (note that missing data is created at the beginning/end of the table):

A screenshot of a computer

Description automatically generated

An alternative to .pct\_change() is df / df.shift(1) - 1 to calculate percentage growth to the previous period:

A screenshot of a computer

Description automatically generated A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated

If you move your data forward/backward you can also move your index in the same direction (taking into account it is correct to do so). Just add the right frequency in the freq = field:

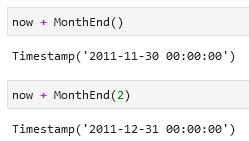
A screenshot of a computer code

Description automatically generated A screenshot of a computer program

Description automatically generated

Offsetting dates is also possible with Day() and MonthEnd():

A screenshot of a computer

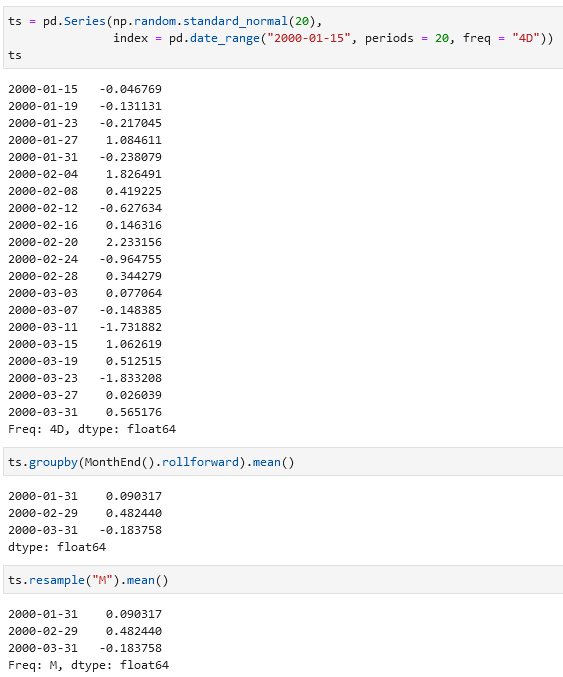
Description automatically generated 

Example of .rollforward() and .rollback():

A screenshot of a computer

Description automatically generated

Getting a monthly average using df.resample(“M”).mean():



**Working with time zones**

Normally tables are created without time zones, but you can add them using tz = . Use df.index.tz to check for a timezone.

A screenshot of a computer

Description automatically generated

To localize your data use .tz\_localize():

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

After localizing your data to a particular time zone, it can be converted to another time zone with .tz\_convert():

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

Also applicable to .index.tz\_localize():

A screenshot of a computer

Description automatically generated

**Periods and period arithmetic**

Period arithmetic:

A screenshot of a computer

Description automatically generated

Frequency conversion using .asfreq():

A screenshot of a computer

Description automatically generated

A diagram of a calendar

Description automatically generated

A screenshot of a computer

Description automatically generated

Quarterly information:

A screenshot of a computer program

Description automatically generated A calendar with months and dates

Description automatically generated

**Converting timestamps to periods and back**

Use .to\_period() to convert timestamps to periods:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Creating an index from your year and quarter columns**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Resampling and frequency conversion**

Use .resample() to change the occurrence of dates like e.g. from daily to monthly:

A screenshot of a computer

Description automatically generated

.resample() arguments:

A screenshot of a computer

Description automatically generated

**Downsampling**

A screenshot of a computer

Description automatically generated

Difference between closed = ”left” and closed = “right”:

A close-up of a number

Description automatically generated

You can also offset e.g. 1 second from your index to make it more clear which interval the timestamp refers to:

A screenshot of a computer code

Description automatically generated

**Open, high, low, close resampling is achieved using .ohlc():**

A screenshot of a computer

Description automatically generated

**Upsampling**

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Resampling with periods**

You can use .resample() to change the period of your DataFrame index:

A screenshot of a computer

Description automatically generated

If you are upsampling (e.g. going from yearly to quarterly) then you can specify features like .ffill():

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Grouped time resampling**

You can also use pd.Grouper() to resample a DataFrame:

A screenshot of a computer

Description automatically generated

**Moving window functions**

To get a rolling moving average or an exponentially weighted moving average use .rolling() or .ewm():

A screenshot of a computer

Description automatically generated A graph with lines and text

Description automatically generated

**Modelling libraries in Python**

Usually the first modelling step is turning a DataFrame into a NumPy array. Use df.to\_numpy() for it:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

To convert a NumPy array back to a DataFrame use pd.DataFrame():

A screenshot of a computer

Description automatically generated

Creating categorial columns and replacing them with Boolean values:

A screenshot of a computer

Description automatically generated

**Patsy library**

Check *Chapter* *12.2 Creating model descriptions with Patsy*.

**statsmodels library**

statsmodels is a Python library for fitting many kinds of statistical models, performing statistical tests, and data exploration and visualization. statsmodels contains more "classical" frequentist statistical methods, while Bayesian methods and machine learning models are found in other libraries.

Some kinds of models found in statsmodels include:

* Linear models, generalized linear models, and robust linear models
* Linear mixed effects models
* Analysis of variance (ANOVA) methods
* Time series processes and state space models
* Generalized method of moments

**Scikit-learn library**

Check *Chapter* *12.4 Introduction to scikit-learn*.

**Data analysis examples**

Unpivoting values in a column is possible using df.explode(“column\_name”):

A screen shot of a movie

Description automatically generated

Open a json file using json.load():

A screenshot of a computer

Description automatically generated

**Appendix A. Advanced NumPy**

A screenshot of a computer program

Description automatically generated

A screenshot of a table

Description automatically generated

A screenshot of a computer code

Description automatically generated

If you want to flatten an array you can use .ravel() (does not produce a copy) or .flatten() (produces a copy):

A screenshot of a computer

Description automatically generated

While using .ravel() you can specify C or F. If you specify F the flattening order will be different:

A screenshot of a computer

Description automatically generated

The key difference between C and FORTRAN order is the way in which the dimensions are walked:

* **C/row major order.** Traverse higher dimensions first (e.g., axis 1 before advancing on axis 0).
* **FORTRAN/column major order.** Traverse higher dimensions last (e.g., axis 0 before advancing on axis 1).

Concatenating arrays is possible using np.concatenate() and np.vstack() or np.hstack():

A screenshot of a computer program

Description automatically generated

Splitting arrays is possible using np.split(). In the example below [1, 3] specifies the indexes at which to split the array into pieces.

A screenshot of a computer program

Description automatically generated

**Array concatenation functions:**

A screenshot of a computer program

Description automatically generated

There are also two functions that can help you concatenate np.r\_ and np.c\_:

A screenshot of a computer code

Description automatically generated

Repeating elements in an array is possible using .repeat(n). The array will be repeated n-number of times.

A screenshot of a computer

Description automatically generated

If you pass an array of integers each element can be repeated a different number of times:

A screenshot of a computer

Description automatically generated

Multidimensional arrays can have their elements repeated along a particular axis:

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

If you want to stack copies of an array along an axis you can use np.tile():

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

If you want to select values based on their indexes and then replace these values with other values you can use .take() and .put():

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Broadcasting**

Broadcasting over axis 0 with a one-dimensional array:

A grid of numbers and a math problem

Description automatically generated with medium confidence

If you want to subtract from columns you need to correct the shape of the mean to (n, 1):

A screenshot of a computer

Description automatically generated A white squares with black numbers

Description automatically generated

Adding a two-dimensional array to a three-dimensional one across axis 0:

A diagram of a battery

Description automatically generated

An alternative to reshaping is using np.newaxis to create a new axis:

A screenshot of a computer

Description automatically generated

Setting array values can be done using [:]:

A screenshot of a computer

Description automatically generated

**Advanced ufunc usage**

Functions .add.reduce() and .add.accumulate() are alternatives to .sum() and .cumsum():

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated

To create a pair-wise cross product between two arrays use np.multiply.outer(array1, array2):

A screenshot of a computer code

Description automatically generated

Example of np.subtract.outer(x, y):

A screenshot of a computer

Description automatically generated

Use np.add.reduceat() to sum values into specified bins (based on indexes). In the example below the first 5 elements are summed into 10, the next 3 into 18 and so on:

A screenshot of a computer

Description automatically generated A screenshot of a computer code

Description automatically generated

Table with ufunc methods:

A screenshot of a computer

Description automatically generated accumulate(), at(), reduce(), reduceat(), outer()

**Nested data types and multidimensional fields**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Partial sorting in NumPy**

If you want to return e.g. 3 highest/lowest elements in an array you can use np.argpartition() and np.partition(). This is an alternative to np.sort(). Difference is (I assume) is speed.

A screenshot of a computer program

Description automatically generated