**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

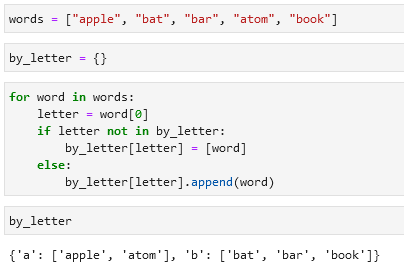
Create a dictionary from a dict():

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

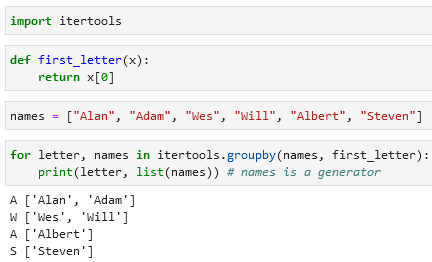
**Generator expressions**

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

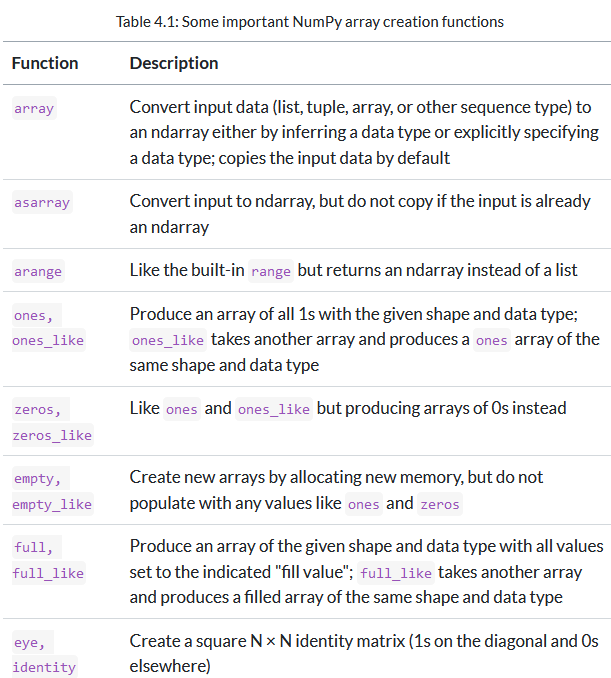
Description automatically generated

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

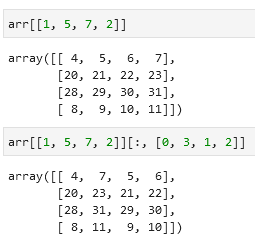
Description automatically generated

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

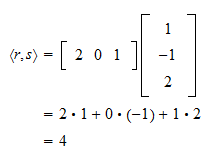
Another example:



**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

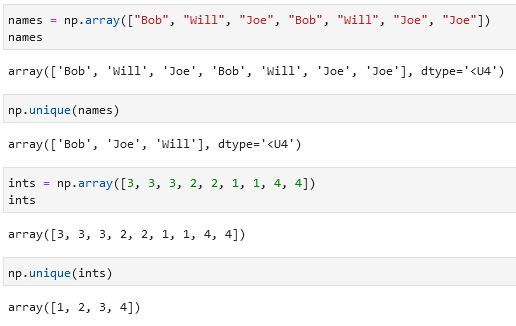
Examples like cumsum():

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):

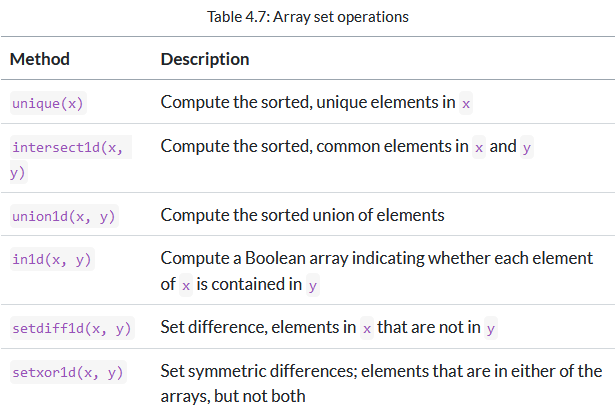
A close-up of a computer screen

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

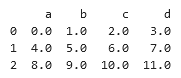
Description automatically generated

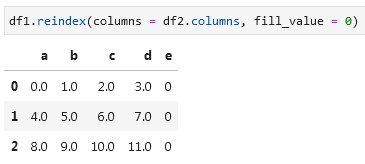
**The .reindex() function will perform a join of values present in your Series based on the current and the new indexes:**

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