

# *Python Handy Tips*

Mahdi Dor Emami

April 28, 2020

# Contents

1	<b>Introduction</b>	1
2	<b>Python General Commands</b>	2
2.1	print command tips	2
2.2	Measuring Time	2
2.3	.join functionality	3
2.4	Lambda Function	3
2.5	Creating a new variable in each iteration of a for loop	3
2.6	Iterator	4
2.7	Python Generator	4
3	<b>Have your files with Glob Library</b>	5
3.1	import glob	5
3.2	glob.glob('*')	5
3.3	glob.glob('*.png')	5
3.4	glob.glob('R*')	5
4	<b>matplotlib Library</b>	5
4.1	import matplotlib.pyplot as plt	5
4.2	plt.plot(...)	5
4.3	plt.show()	5
4.4	plt.xlabel(name)	5
4.5	Subplots	6
5	<b>os Library</b>	7
5.1	import os	7
5.2	os.getcwd()	7
5.3	os.mkdir('name')	7
5.4	os.removedirs('name')	7
5.5	os.chdir(Newdirectory)	7
5.6	os.path.join(Further Address)	7
5.7	os.walk	7
5.8	To Loop over files in a directory	8
6	<b>numpy Library</b>	9
6.1	import numpy as np	9
6.2	np.array(list)	9
6.3	Dimension and Shape	9
6.4	np.array(list).astype('float')	9

6.5	Reshape	9
6.6	np.loadtxt(rawdata,delimiter=",")	9
6.7	np.dstack(TUP)	10
6.8	np.ndarray VS np.array	10
<b>7</b>	<b>pandas Library</b>	<b>11</b>
7.1	import pandas	11
7.2	from pandas import read_csv	11
7.3	pandas.Series(data=Data, index=labels)	11
7.4	pandas.DataFrame(data=Data, index = rowlabels , columns = column labels)	11
7.5	read_csv(filename,names=Names)	12
7.6	DATA.head(#)	12
7.7	Data.dtypes	12
7.8	Data.describe()	12
7.9	Data.groupby('GROUP NAME').size()	12
7.10	Data.pop('GROUP NAME')	12
7.11	Data.loc[index]	12
7.12	Data.isna()	12
7.13	Data.isna().sum()	13
7.14	Data.dropna()	13
7.15	Data.corr(method='pearson')	13
7.16	Plotting Correlation Graph	13
7.17	Pipeline in pandas	14
7.18	Data.skew()	14
7.19	Normalizing	14
7.20	Test,Train Split	14
7.21	One Hot Encoding	15
7.22	data.hist()	15
7.23	data.plot(Options)	15
7.24	DATA.values	16
<b>8</b>	<b>seaborn Library</b>	<b>17</b>
8.1	import seaborn as sns	17
8.2	Correlation Matrix	17
8.3	Scatter Matrix	17
8.4	Scatter Matrix II	17
<b>9</b>	<b>sklearn Library</b>	<b>18</b>
9.1	from sklearn.preprocessing import MinMaxScaler	18
9.2	from sklearn.preprocessing import StandardScaler	18

9.3	from sklearn.preprocessing import Normalizer	18
9.4	from sklearn.preprocessing import Binarizer	18
9.5	SelectKBest(score_func=chi2,k=4)	18
9.6	PCA(n_components=N)	19
9.7	OneHotEncoding	20
9.8	the code	20
9.9	the code	20
<b>10</b>	<b>OpenCV Library</b>	<b>21</b>
10.1	import cv2	21
10.2	Primary Works on Image	21
10.3	Primary Works on Video	22
10.3.1	How to display a video	23
10.3.2	How to save desired frames	23
10.3.3	How to backward-play a video	23
10.3.4	How to export a video file	24
<b>11</b>	<b>Pillow Library</b>	<b>26</b>
11.1	from PIL import Image	26
11.2	Image.open(image)	26
11.3	myimage.show()	26
11.4	myimage.save('filename.anyformat')	26
11.5	Resize(reduce the size only)	26
11.6	Rotate	26
11.7	Convert	26
11.8	Filtering	27
<b>12</b>	<b>Pickle Library</b>	<b>28</b>
12.1	Pickle	28
<b>13</b>	<b>Pathlib Library</b>	<b>29</b>
13.1	Pathlib	29
13.2	Home & CWD	29
13.3	Testing that if this is a real address	29
13.4	Joining paths, getting all subfolders and return back to parent folder	29
13.5	Making a new directory (New folder)	30
13.6	Copying files	30
13.7	Creating an empty file	30
13.8	Rename a file	31
13.9	Remove a file or folder	31
13.10	Prints paths cleaner	31
13.11	path parts	31

13.12	Globbing and finding specific types of files	32
13.13	Counting files by extension	32
13.14	Read & Write a text file	33
13.15	To Loop over files in a directory	33
13.16	Acknowledgements about pathlib tutorial	33
14	<b>Tensorflow</b>	34
14.1	Importing Libraries	34
14.2	General Commands	34
14.3	Tensorflow Datasets	35
14.4	Keras Library	35
14.4.1	Keras Datasets	35
14.4.2	Keras URLs	35
14.4.3	Keras Model	35
14.5	Importing Libraries	35
14.6	Importing Libraries	36
14.7	Importing Libraries	36
15	<b>The End</b>	37
15.1	Importing Libraries	37
15.2	Importing Libraries	37
15.3	Importing Libraries	37
15.4	Importing Libraries	37
15.5	Importing Libraries	37
16	<b>The End</b>	38
16.1	Acknowledgements	38
16.2	Final Words	38

# 1 Introduction

This literature is not a "0 to 100" python learning cookbook or textbook or anything! This is made of only the most useful commands being able to work with python, Written by Mahdi Dor Emami. I hope it contains useful information for you ladies and gentlemen.

**Also**, there are some python files that contain some good examples on specified topic (by the name of the file) with deep explanations inside the files.

## 2 Python General Commands

### 2.1 print command tips

`print()` ⇒ Simply prints... but let's review some tips through examples.

**Example:**

```
>>>print("tell %s(or %d or %f) people" %5)

'tell 5 people'

>>>print("tell {} people" .format(5))

'tell 5 people'

>>>print("tell %s(or %d or %f) + {} people" .format(6)%5)

'tell 5 + 6 people'

>>>print("tell {} + %s(or %d or %f) people" .format(6)%5)

'tell 6 + 5 people'
```

Better than all of the above, and most robust technique :

```
>>>print(f'tell {variable1} + {variable2} (or {variable3}or {variable4}) people')
```

In this way, you can easily add variable name inside the `{}` without needing to write further things like `%` or `.format{}`.

### 2.2 Measuring Time

You have to use library **timeit** to measure time, not **time** library. Because it's way more accurate. Do as below:

**Example:**

```
>>>import timeit
# Imports the library

>>>timeit.timeit('2*3')
# Returns the duration of the statement 2*3 to be done.
```

`timeit.timeit(stmt, setup, timer, number) :`

**stmt:**

This will take the code for which you want to measure the execution time. The default value is "pass".

**setup:**

: This will have setup details that need to be executed before stmt. The default value is "pass."

**timer:**

: This will have the timer value, `timeit()` already has a default value set, and we can ignore it.

**number:**

: The stmt will execute as per the number is given here. The default value is 1000000.

```
>>>timeit.timeit('char in text', setup='text = "sample string"; char = "g"')
# Calculates the time that is necessary for the statement "char in text" that says if char is in text or
```

not. But what is char and text? They are pre-settings that need to be passed in setup.

```
>>>timeit.repeat(stmt,setup,timer,number,repeat)
# This function, repeats measuring the time for "repeat" number. Consider below example:

>>>timeit.repeat('char in text', setup='text = "sample string"; char = "g"',number=1000000,repeat=10)
# This function, measures the time for the operation "char in text" to be done for one million times (if
'char in text' is measured one time, it's a very small number representing its duration). repeat, does it
for 10 times. It will give you a list with 10 time measurements that measured the time for 'char in text'
command running one million times.
```

## 2.3 .join functionality

Just pay attention to the examples! Run them in your Python IDE and see the results.

**Example:**

```
>>>"".join(['Fuck You %d times '% i for i in range(1000)])
# Prints Fuck You 1 until Fuck You 999

>>>"".join(['Fuck You' + ' ' + str(i)+' ' for i in range(1000) ])
# Same as before. Pay attention that there's a space in between the quotes after + sign.

>>>'-''.join(str(m) for m in range(100))
# The result is as below:

'0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-
26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-
51-52-53-54-55-56-57-58-59-60-61-62-63-64-65-66-67-68-69-70-71-72-73-74-75-
76-77-78-79-80-81-82-83-84-85-86-87-88-89-90-91-92-93-94-95-96-97-98-99'
```

## 2.4 Lambda Function

It's a super cool feature, it acts like a function. Just do the examples below to find out. **Example:**

```
>>>x = lambda a : 3*a

>>>print( x(10) )

30
>>>def myfunc(n):

>>>    return lambda a : a * n

>>>mydoubler = myfunc(2)

>>>print(mydoubler(11))

22
```

## 2.5 Creating a new variable in each iteration of a for loop

Sooo cooooooooooooooooooL : **Example:**



```
>>>for x in range(0, 9):

>>>    globals()['var%s' % x] = 'Hello'
```

Another example :

```
>>>for x in range(0, 9):

>>>    globals()['var%s' % x] = 'Hello%s'%i
```

Imagine what you can do with this capability and the pickle library :

```
>>>for i in range(5):

>>>    dat = i

>>>    tmp = 'data_%s'%i

>>>    with open(tmp,'wb') as f:

>>>        pickle.dump(dat,f)

>>>    f.close()
```

## 2.6 Iterator

**x=iter(mylist)** ⇒ make an iteration on mylist. You can use the following commands to be able to see the content of mylist:

**Example:**

```
>>>x = iter(mlist)

>>>next(x)
```

## 2.7 Python Generator

You must define it inside a function... The main item of Generator is 'yield' command that must be inside a function. **Example:**

```
>>>def indexer(text):

>>>    for i,char in enumerate(text):

>>>        yield i,char

>>>res = indexer('Mahdi')

>>>next(res)
```

Just run this function and run next(res) as many times as the length of res. see what will happen...

### 3 Have your files with Glob Library

#### 3.1 `import glob`

Imports the library.

#### 3.2 `glob.glob('*')`

Shows whatever thing is in the Current Working Directory (CWD).

#### 3.3 `glob.glob('*.png')`

Shows all .png files in the CWD.

You can do it with any other extension.

#### 3.4 `glob.glob('R*')`

Shows all the files in the CWD starting with R

### 4 matplotlib Library

#### 4.1 `import matplotlib.pyplot as plt`

`import matplotlib.pyplot as plt` ⇒ Imports the library `matplotlib.pyplot` and names it as `plt`

#### 4.2 `plt.plot(...)`

`plt.plot(...)` ⇒ plots something... you can use "scatter", "stem" etc instead of "plot" in "plt.plot()" command

#### 4.3 `plt.show()`

`plt.show()` ⇒ `plt.plot(variable)` will make it ready to be shown by `plt.show()`, so if you want to see your figures, use it!

#### 4.4 `plt.xlabel(name)`

`plt.xlabel(name)` ⇒ Puts a label on x-axis. You can do the same with `plt.ylabel(name)` for y-axis.

## 4.5 Subplots

`plt.subplot(mnp)`  $\Rightarrow$   $m \times n$  grid plot, which the  $ph$  elements (column-wise) would be plotted now. Remember to put the settings in your IDE (Mine was Spyder) to use "Qt" to show the plots. It is available in Spyder IDE in "Tools (tab)>Preferences>Ipython console>Graphics (tab)> Graphics Backend> Put it on Qt4 or Qt5".It affects new kernels.

**Example:**

```
>>>plt.subplot(131)

>>>plt.plot(Something)

>>>plt.subplot(132)

>>>plt.plot(Something)

>>>plt.subplot(133)

>>>plt.plot(Something)

>>>plt.show()
```

## 5 `os` Library

### 5.1 `import os`

`import os` ⇒ Imports the library

### 5.2 `os.getcwd()`

`os.getcwd()` ⇒ Would show the current working directory

### 5.3 `os.mkdir('name')`

Creates a directory with the name 'name'.

### 5.4 `os.removedirs('name')`

Removes the directory with the name 'name'.

### 5.5 `os.chdir(Newdirectory)`

`os.chdir(Newdirectory)` ⇒ Would change the current directory to the new one. Remember to put two \s instead of one **Example:**

```
>>>os.chdir('C:\\Users\\Mahdi\\Desktop\\Datasets to work on')
```

### 5.6 `os.path.join(Further Address)`

`os.path.join(Further Address)` ⇒ Adds a piece of address to your previous address.

**Example:**

```
>>>Address = os.chdir('C:\\Users\\Mahdi\\Desktop\\Datasets to work on')
```

```
>>>NewAddress = os.path.join(Address,"Cars")
```

```
>>>print(NewAddress)
```

```
'C:\\Users\\Mahdi\\Desktop\\Datasets to work on\\Cars')
```

### 5.7 `os.walk`

You have to use it in the way explained below to be able to get all directories, subdirectories, and all files in a directory.

**Example:**

```
>>>for a,b,c in os.walk(os.getcwd()):

>>>    print(f'a:a')

>>>    print(f'b:b')

>>>    print(f'c:c')
```

Just test it to see how it works.

## 5.8 To Loop over files in a directory

Do as below:

### Example:

```
>>>directory = os.getcwd()
# Comment

>>>for filename in os.listdir(directory):
# os.listdir(directory) lists everything inside the directory

>>>    if filename.endswith(".asm") or filename.endswith(".py"):
# filename.endswith('extension') searches for the files that have the specified extension in the listed items
in directory

>>>        print(os.path.join(directory, filename))
```

YOU CAN DO THIS WITH pathlib LIBRARY TOO. JUST GO TO THE pathlib SECTION.

There is also a good other function that can give the filename and extensions:

```
>>>for file in os.listdir():

>>>    filename,file_extension = os.path.splitext(file)

>>>    print(f'Filename = [filename] && File_extension =[file_extension]')
```

## 6 numpy Library

### 6.1 import numpy as np

**import numpy as np** ⇒ Imports numpy library and makes np as the contraction name for numpy library

### 6.2 np.array(list)

**x=np.array(list)** ⇒ Makes the 'list' a numpy array

### 6.3 Dimension and Shape

**np.dim()** && **np.shape()** ⇒ We can use both of these commands in order to get the dimension or shape of any types of data like tuple, list, dict, etc...

### 6.4 np.array(list).astype('float')

**x=np.array(list).astype('float')** ⇒ Makes the 'list' a numpy array with float type elements. There are some good functions for numpy arrays like:

```
>>>np.max()# Gives the maximum value in the array (or matrix)
```

```
>>>np.min()# Gives the minimum value.
```

```
>>>np.mean()# Calculates the mean.
```

```
>>>np.argmax()# Gives the index referring to the maximum value in the array (or matrix).
```

### 6.5 Reshape

**x.reshape(size,order=)** ⇒ Reshapes the data in specific size with sepecific order. order values are : order='C', order = 'F',order='A'

C means to read / write the elements using C-like index order, with the last axis index changing fastest, back to the first axis index changing slowest. F means to read / write the elements using Fortran-like index order, with the first index changing fastest, and the last index changing slowest. Note that the C and F options take no account of the memory layout of the underlying array, and only refer to the order of indexing. A means to read / write the elements in Fortran-like index order if a is Fortran contiguous in memory, C-like order otherwise.

### 6.6 np.loadtxt(rawdata,delimiter=",")

**x=np.loadtxt(rawdata,delimiter=",")** ⇒ Turns a CSV file which is loaded by **open(filename,'rt')** in the directory, into a numpy array with float values.

## 6.7 `np.dstack(TUP)`

The code concatenates two vectors, matrices, whatever, in 3rd axis (depth). Consider a,b be both vectors of dimension (M,) . `np.dstack((a,b))` returns a (1,M,2) tensor since first, a dimension will be added to both so they become (M,1). Then another dimension is inserted, this time to the head and makes it (1,M,1) for each a,b. And finally, both will be concatenated along the third axis. The result is a (1,M,2) tensor.

There is a difference between `np.dstack(TUP)` and `np.concatenate(TUP,axis=2)`. First, in **`np.concatenate(TUP,axis=2)`** the axis values are in range (0,1,2,...) where 0 is rows, 1 columns, 2 depth ,3 higher dimension,etc. So now, we want to concatenate a,b vectors with dimension (M,) on the third dimension. `a[:,None]` will add another dimension to a , so a becomes (M,1). **`np.concatenate((a[:,None],b[:,None,None]),axis=2)`** will get us a tensor of shape (M,1,2).

**`np.hstack((a,b))`** concatenates a and b on rows so it will give a (2\*M,) dimension matrix.

**`np.vstack((a,b))`** concatenates a and b on columns so it will give a (2,M) dimension matrix. It goes vertically... So, we have two flat vecs `a=[1,2,3]` and `b=[4,5,6]`, It'll go vertically like `[[1,4],[2,5],[3,6]]`.

## 6.8 `np.ndarray VS np.array`

No differences... But, `np.array` is much more high-level, so use `np.array` all the time.

## 7 pandas Library

### 7.1 import pandas

`import pandas` ⇒ Imports pandas library

### 7.2 from pandas import read\_csv

`from pandas import read_csv` ⇒ Imports a sublibrary that is expert to open csv files and turn them into "Pandas Data Frames"

### 7.3 pandas.Series(data=Data, index=labels)

`panda.Series` ⇒ Makes a labeled vector from one dimensional Data.

### 7.4 pandas.DataFrame(data=Data, index = rowlabels , columns = column labels)

`pandas.DataFrame` ⇒ Makes a row&column labeled dataset. One can refer to the labeled data in "method" way:

**Example:**

```
>>>a=[[2,4,5],[4,3,2]]

>>>rows = ['first','second']

>>>column = ['one','two','three']

>>>Data=pandas.DataFrame(a,index=rows,columns=column)

>>>Data
      one two three
first   2   4    5
second  4   3    2

>>>Data.one
first    2
second  4
Name: one, dtype: int64

>>>Data.three
first    5
second  2
Name: one, dtype: int64
```

#Notice that this method only works for column labels, not for rows



### 7.5 `read_csv(filename,names=Names)`

**Dataset = read\_csv(...)** ⇒ Makes a DataFrame out of a CSV file (Comma Separated Value file). It loads the dataset and tags the Names on each column of it. If the dataset exists in the current directory, filename is only the name of the file plus its extension; Otherwise, you have to refer the whole address of the dataset plus name and extension of the file. One can use a URL address instead of filename as well.

### 7.6 `DATA.head(#)`

**DATA.head(#)** ⇒ If DATA is a pandas DataFrame, this .head(#) method would show the first # rows of the dataset, in order to have a quick perspective.

### 7.7 `Data.dtypes`

**Data.dtypes** ⇒ Represents every column's data's types.

### 7.8 `Data.describe()`

**x=Data.describe()** ⇒ Gives the necessary statistical information on the data like "mean" , "standard deviation" , etc. You can also set some options related to the precision like : set\_option('precision',3) which sets the decimal point precision.

### 7.9 `Data.groupby('GROUP NAME').size()`

**Data.groupby('GROUP NAME').size()** ⇒ Returns the number of occurrence for each value in the specified group (column) of the data. "GROUP NAME" is the name of the column in the dataset.

### 7.10 `Data.pop('GROUP NAME')`

Picks the specified column from the dataset and stores it in any variable you want.

### 7.11 `Data.loc[index]`

Shows the specified row from the dataset with its full features.

### 7.12 `Data.isna()`

Shows "unknown" or "None" values in the dataset.

### 7.13 `Data.isna().sum()`

Counts "unknown" or "None" values in the dataset.

### 7.14 `Data.dropna()`

We'll get rid of "unknown" or "None" values in the dataset.

### 7.15 `Data.corr(method='pearson')`

`Data.corr(method='pearson')`  $\Rightarrow$  gives the correlation of the data with each other by using "pearson" method. There are other methods for calculating the correlation like "spearman" or "kendall".

### 7.16 `Plotting Correlation Graph`

Imagine you loaded the dataset and named it as DATA. Watch the example below to learn how to do it:

**Example:**

```
>>> correlations = DATA.corr()

>>> fig = plt.figure()
# Have to store in order to make changes in it.

>>> ax = fig.add_subplot(111)
# A 1 by 1 plot.

>>> cax = ax.matshow(correlations, vmin = -1, vmax = 1)
# Plots the correlation matrix

>>> fig.colorbar(cax)
# Places a colorbar next to the main plot relevant to the correlation matrix plotted there.

>>> names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
# Makes names

>>> ticks = np.arange(0,9,1)
# Arrange the numbers based on the classes of your database. Mine are nine classes.

>>> ax.set_xticks(ticks)
# You have to pass this step to correctly be able to label your data. Otherwise, the labels are gonna be tagged in a wrong manner. This would tell the correlation chart to count the classes from 0 to 9

>>> ax.set_yticks(ticks)
# Same manners like previous line of code.

>>> ax.set_xticklabels(names)
# Names all the columns

>>> ax.set_yticklabels(names)
# Names all the rows
```

```
>>>ax.set_xlabel('DATA CLASSES')
# Sets the main label for x-axis

>>>ax.set_ylabel('DATA CLASSES')
# Sets the main label for y-axis
```

## 7.17 Pipeline in pandas

Pipelines generally are used to do a number of functions sequentially. Imagine you have two different functions : func1 and func2 , whose parameters are (DataFrame , Column). If your Pandas.DataFrame structure name is "data" for example, the following command will apply all functions on the column "Age" of "data" DataFrame.

**Example:**

```
>>>automation = data.pipe(func1 , Col='Age') , data.pipe(func2 , Col='Age')

# In this way, you will have an automatic way of doing several functions on several groups of data.
```

## 7.18 Data.skew()

**Data.skew()**  $\Rightarrow$  Gives the skew of a Normal Gaussian distributed data. Poitive skew means that the data histogram is right taled; Better to say it's concentration is on the left part. And Negative skew means that the data histogram is left taled; It means that the concentration is on the right side. The skew formula is  $\frac{(mean - median)}{std}$

## 7.19 Normalizing

We can normalize the data:

**Example:**

```
Everything is clear from the code.

>>>des = dataset.describe().transpose()

>>>new_dataset = (dataset - des['mean']) / des['std']
```

## 7.20 Test,Train Split

You can do splitting on the dataset:

**Example:**

```
Everything is clear from the code.

>>>train = dataset.sample(frac=0.6 , random_state = seed)

>>>valid = dataset.drop(train.index)

>>>validation = valid.sample(frac=0.4 , random_state = seed)

>>>test = valid.drop(validation.index)
```

## 7.21 One Hot Encoding

You can do a one hot encoding by two ways for your dataset:

### Method I:

Imagine that the 'sex' feature of your dataset is a categorical feature containing only 'male' and 'female' as values.

```
>>>dataset['sex'] = dataset['sex'].map({'female':0 , 'male':1})  
  
# Easy peasy
```

### Method II:

You can do it automatically by this command of pandas:

```
>>>new_data = pd.get_dummies(dataset)  
  
# It's actually doing a one hot encoding for it... And added them to your features.
```

### Method III:

This method is for categorical features that are represented as numbers. For example, the samples are in 4 classes determined by 0,1,2,3 digits. You can do a one hot encoding on this feature as below:

```
>>>dataset['class'].unique()# Checks the unique things in class. In our example, it will return 0,1,2,3  
  
array([0,2,1,3] , dtype=int64)  
  
>>>dataset['class'] = dataset['class'].map({0:'class1' , 1:'class2' , 2:'class3' , 3:'class4'})  
  
# First, we replace the 0,1,2,3 with class1,class2,class3,class4.  
  
>>>new_data = pd.get_dummies(dataset)  
  
# Finally, we change it.  
  
##### There are some arguments in pd.get_dummies() like prefix , prefix_sep. If you pass nothing,  
prefix will be the previous feature name and prefix_sep will be _ . You can change these like for example:  
prefix='@' , prefix_sep = '^'
```

## 7.22 data.hist()

**data.hist()** ⇒ Prepares the histogram plots for all the columns (groups or classes) of the data in one figure. This will be shown by the command `plt.show()`.

## 7.23 data.plot(Options)

**data.plot(kind = 'density' , subplots = True , layout = (subplot dimension... like (3,3)) , sharex = False or True , sharey = False or True)** ⇒ Would plot different figures for all the groups with a few settings. Like density figure which is declared in "kind" option. There are other "kind"s as well, like " line " , " bar " , " barh " , " box " , " kde " , "density" , "area" , "pie" , "scatter" , "hexbin". subplot option must be turned on if you want to plot for all of the groups, and the dimension should match the number of your sub classes. The last options are sharey and sharex that if they are set to "True", the axis x or y related to sharex or sharey, wouldn't be scaled properly. Otherwise, if you set it

to "False", the scale is shown. density plot, is like histogram with continuous lines. And box, is a special type of plot that draws a box around the middle 50% of the data; Meaning it covers the 25th percentile up to 75th percentile. Also there is a line inside the box remarking the median.

## 7.24 DATA.values

**x=DATA.values** ⇒ Turns a DataFrame into a numpy array. So, it would be indexed normally with `""[]`. Otherwise, if it was a DataFrame, it would have been referenced by its class names. (DATA.park or DATA['park'])

## 8 **seaborn Library**

### 8.1 **import seaborn as sns**

`import seaborn as sns`  $\Rightarrow$  Imports the seaborn library useful for statistical data analysis.

### 8.2 **Correlation Matrix**

`sns.heatmap(CorrMat , annot=True , vmin=-1 , vmax = 1)`  $\Rightarrow$  Returns a figure containing the correlation of all of the groups of a DataFrame within (-1,1). CorrMat can be `data.corr()`. `annot`, means the number in each cell. Run the code with and without it to understand.

### 8.3 **Scatter Matrix**

`sns.pairplot(dataset)`  $\Rightarrow$  Returns the scatter-matrix related to the dataset.

### 8.4 **Scatter Matrix II**

`sns.pairplot(dataset,hue='class')`  $\Rightarrow$  All attributes in relation with "class" for classification possibility checking

## 9 sklearn Library

### 9.1 from sklearn.preprocessing import MinMaxScaler

**from sklearn.preprocessing import MinMaxScaler**  $\Rightarrow$  Would import the "MinMaxScaler" algorithm from preprocessing part of sklearn library suitable for scaling the data between 0 and 1.

**Example:**

```
>>>scaler = MinMaxScaler(feature_range(0,1))# Sets the range of normalization
>>>rescaled = scaler.fit_transform(DATA)# Applies the normalization to the DATA.
```

### 9.2 from sklearn.preprocessing import StandardScaler

**from sklearn.preprocessing import StandardScaler**  $\Rightarrow$  Imports the StandardScaler library that will scale the dataset with the mean being 0 and the std being 1.**Linear Regression or Logistic Regression are the algorithms the StandardScaler is useful and the times one wants to make a standard Gaussian Distributed input.**

**Example:**

```
>>>scaler = StandardScaler().fit(DATA)# Sets the required setting.
>>>rescaled = scaler.transform(DATA)# Applies the changes.
```

### 9.3 from sklearn.preprocessing import Normalizer

**from sklearn.preprocessing import Normalizer**  $\Rightarrow$  Imports the library to normalize the data.**Normalizing would be used for sparsed Data and K-Nearest Algorithms.**

**Example:**

```
>>>scaler = Normalizer().fit(DATA)# Sets the required setting.
>>>normalized = scaler.transform(DATA)# Applies the changes.
```

### 9.4 from sklearn.preprocessing import Binarizer

**from sklearn.preprocessing import Binarizer**  $\Rightarrow$  Imports the library to make a binary representation of the data.

**Example:**

```
>>>scaler = Binarizer(threshold=0.0).fit(DATA)# Sets the required setting.
>>>binarized_data = scaler.transform(DATA)# Applies the changes.
```

### 9.5 SelectKBest(score\_func=chi2,k=4)

**The above code**  $\Rightarrow$  Would be used for feature selection. It would use the scoring statistical function "chi2" in order to select the k best scored features and keep them as the necessary features for all of the classes in dataset. **It will select the features which has the strongest relationships with the output.** You will have to import two libraries and then apply this algorithm on the training and target

sets as below:

**Example:**

```
>>>from sklearn.feature_selection import SelectKBest# Imports the library SelectKBest

>>>from sklearn.feature_selection import chi2# Imports the scoring function chi2

>>>array = DATA.values# Turns the pandas DataFrame into an array

>>>X = array[:,0:8]
# Selects 8 classes of the dataset to be as training sets

>>>Y = array[:,8]
# Puts the last to be as the target set.

>>>test = SelectKBest(score_func_chi2 , k=4)
#Enables the algorithm that selects the 4 best classes for all of the dataset with chi2 statistical algorithm

>>>fit = test.fit(X,Y)
# The algorithm is applied to X,Y... So best scores to get from X to Y are selected

>>>print(fit.scores_)
# Prints out the scores related to each class of the dataset X. The top score classes are goanna be selected.

>>>features = fit.transform(X)
# Now applies the transform to X. So we have the features we want for training.

>>>print(array.shape)
# Main dataset dimension

(768,9)

>>>print(X.shape)
# Training dataset dimension

(768,8)

>>>print(features.shape)
# The selected features for training shape.

(768,4)
```

## 9.6 **PCA(n\_components=N)**

**The code**  $\Rightarrow$  Is for PCA algorithm to reduce data size. It chooses N

**Example:**

```
>>>Python code
# Comment

result
```



## 9.7 OneHotEncoding

Makes a categorical thing into numbers. Like male,female seperation which names them 01,10 for example.

```
>>>from sklearn.preprocessing import OneHotEncoder
# Imports library

>>>OneHotEncoder().fit(X).transform(X).toarray()
# Returns the encoded array

>>>OneHotEncoder().fit_transform(X).toarray()
# Returns the encoded array (Just like above)
```

## 9.8 the code

**x=code** ⇒ explanations

**Example:**

```
>>>Python code
# Comment

result
```

## 9.9 the code

**x=code** ⇒ explanations

**Example:**

```
>>>Python code
# Comment

result
```

## 10 OpenCV Library

This section, is mainly from "Mastering OpenCV4 with Python" from Alberto Fernandez Villian.

### 10.1 import cv2

**import cv2** ⇒ Imports the OpenCV library.

### 10.2 Primary Works on Image

Due to a lot of codes, I'll be explaining this part a little bit different. Mostly, one line code one line comment, and further, with examples.

```
>>>image = cv2.imread('1.jpg')
# Imports the image in the directory. Becareful that imported image (frames of the videos as well), are
in BGR format; Not RBG!! Because it was more popular with cameras.
```

```
>>>gray_image = cv2.imread('1.jpg' , cv2.IMREAD_GRAYSCALE)
# Imports the image in gray-scale
```

```
>>>height,width,channels=image.shape
# Gives hieght, width, and number of channels of the image (RGB num of channels:3 grayscale:1)
```

```
>>>total_pixels = image.size
# Number of pixels
```

```
>>>image_type = image.dtype
# Image type
```

```
>>>image_grayed = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
# Turns an image into gray. It's a kind of process and you can change any image in everywhere of your
code. Just notice at BGR2GRAY. If you had an RGB image, write RGB2GRAY.
```

```
>>>modified_image = image
>>>modified_image[400:500,400:500,:] = image[600:700,600:700,:]
# You can modify images.
```

```
>>>B,G,R = cv2.split(image)
# You can split the channels of an image
```

```
>>>RGB_image = cv2.merge([R,G,B])
# You can merge the channels and create a colorful image with any format you like! RGB or BGR. In
this code, we created an RGB image.
```

```
>>>b = image[:, :,0]
>>>g = image[:, :,1]
>>>r = image[:, :,2]
>>>rgb_image = image[:, :, :-1]
>>>rgb_image_gray = cv2.cvtColor(rgb_image,cv2.COLOR_RGB2GRAY)
# Working with Numpy library to split channels, and build an RGB image. Last line of code is making
the RGB image in gray-scale.
```

```
>>>con_image_1 = np.concatenate((gray_image,image_grayed),axis=1)
>>>con_image_2 = np.concatenate((RGB_image,rgb_image),axis=1)
>>>con_image_3 = np.concatenate((RGB_image_gray,rgb_image_gray),axis=1)
```

```
# A little numpy to stick a few pictures together horizontally(axis = 1) or vertically(axis = 0)

>>>cv2.imshow("Title",image)
# Shows the image. The title must be set. It's the title of the windows displaying the image.

>>>cv2.waitKey()
# Without this command, you can not see the displayed image. You can set time in mili-seconds inside
parantheses to specify the duration of the display.

>>>cv2.destroyAllWindows()
# It's necessary if you want to get rid of the display window and shut it with any keys on your keyboard.

>>>plt.figure,plt.subplot(nmp),plt.imshow(image)
# You can display images with pyplot.imshow and subplots too.

>>>cv2.imwrite('Gray.jpg',image_grayed)
# Finally, you can export the image with cv2.imwrite command, in your working directory. It needs a
name and the image to export as arguments.
```

### 10.3 Primary Works on Video

Explanations will be like the last sub-section.

```
>>>video = cv2.VideoCapture('1.mp4')
# Imports the video. If you put a URL address related to your public camera, it would get that video
for you. Or, if you put 0 (zero) as the argument, it would get your webcam as the input video.

>>>video.isOpened()
# Checks if video is loaded correctly and available to process

>>>ret, frame = video.read()
# reads the first frame and saves it as frame. and ret is True if this action is done
```

Codes below would give you properties about the video. The **Bold** ones are more important than the others.

```
>>>video.get(cv2.CAP_PROP_FRAME_WIDTH)
>>>video.get(cv2.CAP_PROP_FRAME_HEIGHT)
>>>video.get(cv2.CAP_PROP_FPS)
>>>video.get(cv2.CAP_PROP_POS_MSEC)
>>>video.get(cv2.CAP_PROP_POS_FRAMES)
>>>decode_fourcc(capture.get(cv2.CAP_PROP_FOURCC))
>>>video.get(cv2.CAP_PROP_FRAME_COUNT)
>>>video.get(cv2.CAP_PROP_MODE)
>>>video.get(cv2.CAP_PROP_BRIGHTNESS)
>>>video.get(cv2.CAP_PROP_CONTRAST)
>>>video.get(cv2.CAP_PROP_SATURATION)
>>>video.get(cv2.CAP_PROP_HUE)
>>>video.get(cv2.CAP_PROP_GAIN)
>>>video.get(cv2.CAP_PROP_EXPOSURE)
>>>video.get(cv2.CAP_PROP_CONVERT_RGB)
>>>video.get(cv2.CAP_PROP_RECTIFICATION)
>>>video.get(cv2.CAP_PROP_ISO_SPEED)
>>>video.get(cv2.CAP_PROP_BUFFERSIZE)
```

Most of them are clear. The important ones (Bolded ones) are width, height, and FPS of the image; Along side with FOURCC command wich will give the codec of the video. The function "decode\_fourcc"

is as below:

```
>>>def decode_fourcc(fourcc):
    """Decodes the fourcc value to get the four chars identifying it"""
    # Convert to int:
>>>    fourcc_int = int(fourcc)
    # We can do this one line command below, or go the usual way as for loop:
    # One line:
>>>    return "".join([chr((fourcc_int >> 8 * i) & 0xFF) for i in range(4)])
    # Typical:
>>>    fourcc_decode = ""
>>>    for i in range(4):
>>>        int_value = fourcc_int >> 8 * i & 0xFF
>>>        fourcc_decode += chr(int_value)
>>>    return fourcc_decode
# It's the way to decode the fourcc code into the ASCII mode we all know. We have XDIV, AVC1, AVI
etc fourcc codes.
```

### 10.3.1 How to display a video

```
>>>while video.isOpened():
>>>    ret,frame = video.read()
>>>    cv2.imshow('Main Video',frame)
>>>    gray_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
>>>    cv2.imshow('Gray Video',gray_frame)
>>>    if cv2.waitKey(30) & 0xFF == ord('q'):
# By 0xFF == ord('q') command, you can end and exit the video-displaying window by pressing the
'q' button. It's just a complicated thing to explain why, so just learn it. You can not put cv2.waitKey()
outside of if, otherwise by pressing 'q' button, you'll only go to the next frame. You can not replace
"ord('q')" with 'q' either because 'ord' is necessary.
>>>        break
# It'll break out of the while loop.
>>>video.release()
>>>cv2.destroyAllWindows()
# This is the way to show a video; Frame by frame. Because cv2.waitKey() is a method (function) so
break doesn't work in it,You have to bring cv2.waitKey() inside "if" procedure.
```

### 10.3.2 How to save desired frames

You can save videos too. Just enter the codes inside while loop, before "if cv2.waitKey(30) & 0xFF == ord('q')" code. The piece of code is like this example:

```
# Press c on keyboard to save current frame
>>>if cv2.waitKey(20) & 0xFF == ord('c'):
>>>frame_name = "camera_frame-{}.png".format(frame_index)
>>>gray_frame_name = "grayscale_camera_frame-{}.png".format(frame_index)
>>>cv2.imwrite(frame_name, frame)
>>>cv2.imwrite(gray_frame_name, gray_frame)
>>>frame_index += 1
# This frame_index is only a growing number for naming the frame you want to save. Remember to put
its value to 0 (zero) before the while loop.
```

### 10.3.3 How to backward-play a video

Now, a code to review the video in reverse:

```

# We get the index of the last frame of the video file:
>>>frame_index = capture.get(cv2.CAP_PROP_FRAME_COUNT) - 1
>>>print("starting in frame: '{}'".format(frame_index))

# Read until video is completed:
>>>while capture.isOpened() and frame_index != 0:

# We set the current frame position:
>>>    capture.set(cv2.CAP_PROP_POS_FRAMES, frame_index)

# Capture frame-by-frame from the video file:
>>>    ret, frame = capture.read()

    >>>    if ret is True:

# Print current frame number per iteration
#print("CAP_PROP_POS_FRAMES : '{}'".format(capture.get(cv2.CAP_PROP_POS_FRAMES)))

# Get the timestamp of the current frame in milliseconds
# print("CAP_PROP_POS_MSEC : '{}'".format(capture.get(cv2.CAP_PROP_POS_MSEC)))

# Display the resulting frame:
>>>    cv2.imshow('Original frame', frame)

# Convert the frame to grayscale:
>>>    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# Display the grayscale frame:
>>>    cv2.imshow('Grayscale frame', gray_frame)
# Decrement the index to read next frame:
>>>    frame_index = frame_index - 1
>>>    print("next index to read: '{}'".format(frame_index))
# Press q on keyboard to exit the program:
>>>    if cv2.waitKey(25) & 0xFF == ord('q'):
>>>        break
# Break the loop
>>>    else:
>>>        break
>>>video.release()
>>>cv2.destroyAllWindows()
# Hope everything is clear :)

```

### 10.3.4 How to export a video file

You first have to specify the fourcc code you want the video to be coded into. The whole procedure is displayed to you below:

```

>>>fourcc = cv2.VideoWriter_fourcc(*'XVID')
# '*'XVID' returns 'X' 'V' 'I' 'D'. You have to enter the fourcc code like that.
>>>out_gray = cv2.VideoWriter("PATH", fourcc, int(fps), (int(frame_width), int(frame_height)),
False)
# Last one is for grayscale (True = grayscale... False = Colorful; This doesn't work without cv2.cvtColor(frame,cv2.COLO

>>>while capture.isOpened():
# Read the frame from the camera
>>>    ret, frame = capture.read()
>>>    if ret is True:

```

```

# Convert the frame to grayscale
>>> gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# Write the grayscale frame to the video
>>> out_gray.write(gray_frame)
# We show the frame (this is not necessary to write the video)
# But we show it until 'q' is pressed
>>> cv2.imshow('gray', gray_frame)
>>> if cv2.waitKey(1) & 0xFF == ord('q'):
>>>     break
>>> else:
>>>     break

# Release everything:
>>> capture.release()
>>> out_gray.release()
>>> cv2.destroyAllWindows()

```

## 11 Pillow Library

### 11.1 from PIL import Image

The code imports the library we want.

### 11.2 Image.open(image)

`myimage = Image.open('file.jpg')` ⇒ Opens the image and stores it in myimage.

### 11.3 myimage.show()

Shows the image.

### 11.4 myimage.save('filename.anyformat')

You can store the loaded image in any formats you like; .png, .jpg, .jpeg, etc.

### 11.5 Resize(reduce the size only)

`myimage.thumbnail(size_in_tuple)` ⇒ Resizes myimage with the size in tuple, like (300,300), passed in the thumbnail function.

### 11.6 Rotate

`myimage.rotate(90)` ⇒ Rotates the image for you :)

### 11.7 Convert

`myimage.convert(model='argument')` ⇒ You have to pass the type of conversion you want as 'argument' to get it. A list of arguments is as below:

- `1` (1 – bitpixels, blackandwhite, storedwithonepixelperbyte)
- `L` (8 – bitpixels, blackandwhite)
- `P` (8 – bitpixels, mappedtoanyothermodeusingacolorpalette)
- `RGB` (3x8 – bitpixels, truecolor)
- `RGBA` (4x8 – bitpixels, truecolorwithtransparencymask)
- `CMYK` (4x8 – bitpixels, colorseparation)
- `YCbCr` (3x8 – bitpixels, colorvideoformat)

etc...

## 11.8 Filtering

**from PIL import ImageFilter**  $\Rightarrow$  With this library you can put so many filters on your picture.

**Example:**

```
>>>from PIL import Image ImageFilter
# Imports both "Image" and "ImageFilter" libraries.

>>>myimage = Image.open('image.jpg')

>>>new_image = myimage.filter(ImageFilter.GaussianBlur(#))
# Makes it blur and more blur as # increases.
```



## 12 **Pickle Library**

### 12.1 **Pickle**

Good for loading and a kind of algorithm to serialize and save a data. Let's say we have something like a dictionary to save. Let's go as below:

**Example:**

```
>>>import pickle# Imports the library

# To Save a Pickle File

>>>outpickle=pickle.open('Mydesiredpickle','wb')# Opens a pickle protocol in writing mode to save
our pickle with desired name in CWD.

>>>pickle.dump(dictionary,outpickle)# Creates the pickle for us.

>>>outpickle.close()# Closes the outpickle. You have to use this command at the end to close the
protocol.

# To Load a Pickle File

>>>picklein = open(filename,'rb')# Opens the pickled file in reading mode (rb stands for readig
binary and good for numeric data)

>>>mydict = pickle.load(picklein)# Loads and decodes the pickle file into whatever it was.

>>>picklein.close()# To close the opened file.
```

Note that, if you want to compress the file alongside saving that, all you need to do is to import a library called "bz2" standing for Bzip2 and do as follows:

**Example:**

```
# To Save the Pickle for Compressed File

>>>import bz2# Imports the library

>>>zipped = bz2.BZ2File(filename , 'w')# Opens a zip protocoled file.

>>>pickle.dump(mydict,zipped)# Pickles mydict dictionary into the compressed file.

>>>zipped.close()# Closes the file.

# To Load the Pickle for Compressed File

>>>zipped = bz2.BZ2File(filename , 'rb')# Opens the zipped file.

>>>mydict=pickle.load(zipped)# Loads and decodes the pickle file into whatever it was.

>>>zipped.close()# Closes the file.
```

## 13 Pathlib Library

### 13.1 Pathlib

`from pathlib import Path` ⇒ Very handy in data controlling

### 13.2 Home & CWD

`Path.cwd()` & `Path.home()` ⇒ These two will print current working directory and home directory respectively.

**Example:**

```
>>>print(f'cwd is {Path.cwd()}')

cwd is C:\\Users\\Mahdi
>>>print(f'cwd is {Path.home()}')# Prints home path.
```

### 13.3 Testing that if this is a real address

`path.is_dir()` ⇒ Does the job. Pay attention to the example:

**Example:**

```
>>>px = Path()

>>>px = px / 'GUBOLI'
# There is no folder GUBOLI. But px will have the address.

>>>px
# To observe px

WindowsPath('GUBOLI')

>>>px.is_dir()
# Shows that it's not a real address.

False
```

### 13.4 Joining paths, getting all subfolders and return back to parent folder

Code below does it:

**Example:**

```
>>>path = Path()
# path becomes current working directory

>>>path = Path('E:/subfolder/subsubfolder')
# path becomes the required directory. But, in order to change CWD, you must use os library and
nothing else.

>>>directory = [x for x in path.iterdir() if x.is_dir()]
# All the subfolders inside the current working directory, if they are real addresses will be joined to
```

directory list.

```
>>>file = [x for x in path.iterdir() if x.is_file()]
#All the files inside the current working direcotry, if they are files not folders.

>>>path = path / 'newfolder'
# Now, our path is in the newfolder. Like we opened newfolder in windows. If you run that path.iterdir()
code, you will get the things inside the newfolder.

>>>path = path.parent
# Returns to the parent folder

>>>path = path.parent.parent.parent
# You can use whatever number of .parents you like to get back and further back
```

### 13.5 Making a new directory (New folder)

`path.mkdir()`  $\Rightarrow$  Makes it. Look at the example:

**Example:**

```
>>>mypath = Path()

>>>mypath = mypath / 'Mynewfolder'# This folder doesn't exist yet!

>>>mypath.mkdir()# The folder will be created in the path.
```

### 13.6 Copying files

Done by importing `shutil` library. Straight to examples:

**Example:**

```
>>>from shutil import copyfile

>>>path = Path()

>>>path = path / 'Mahdi' / 'Desktop'

>>>Source = path / '1.mp4'# Selects an mp4 file as source file

>>>Destination = path / '2.mp4'# Selects the destination

>>>copyfile(Source, Destination)# Copy is done :)
```

### 13.7 Creating an empty file

We can create empty files. MP3, MP4, txt and whatever format you pass to your path. Imagine path is in the Desktop folder. **Example:**

Just go to your desired path and make sure your CWD is the path you like by `Path.cwd()` command. I must recall that you can go to the desired path and select it as your CWD by `os` library like `os.chdir(MyDesiredAddress)`. And if you type `Path.cwd()`, you will be in there.

```

# You are in the place.

>>>Path('myfile.txt').touch()# Creates the file for you in your place.

# You are not in the place, act like below

>>>address = os.getcwd()# saves CWD in address

>>>newaddress = os.path.join(address,'Desktop')# If address was C : \\Users\\UserName , you
can add Desktop to it as well.

>>>os.chdir(newaddress)# Now CWD is on Desktop

>>>Path('myfile.txt').touch()# Creates the file for you in your Desktop.

```

### 13.8 Rename a file

Imagine you are in the directory the file exists.

**Example:**

```

>>>path = Path('file1.txt')# Selects file1

>>>path.rename('file100.txt')# Renames it to file100

```

### 13.9 Remove a file or folder

Imagine you are in the directory the file exists.

**Example:**

```

>>>path = Path('file1.txt')# Selects file1

>>>path.unlink()# Removes the file

>>>path = Path('Foler')# Selects Folder

>>>path.rmdir()# Removes the Folder

```

### 13.10 Prints paths cleaner

Use path.as\_posix()

### 13.11 path parts

Can see the root, drive and parts of the path

**Example:**

```

>>>path = Path.cwd()

>>>path.parts# Shows the path parts

```

```

('C : \\', 'Users', 'Mahdi', 'Desktop')

>>>path.drive# The drive in which the path is

'C:'

>>>path.root# It's most used in Linux systems but, for windows it only shows a simple thing:

'\\'

```

### 13.12 Globbing and finding specific types of files

Glob patterns specify sets of filenames with wildcard characters. For example, the \*.txt represents all files with names ending in .txt. The \* is a wildcard standing for any string of characters. Read comments because they provide you with important materials.

#### Example:

```

>>>for e in path.rglob('*.py'):
>>>    print(e)

# Prints all the python files in the CWD AND in the subfolders of the CWD.

>>>path.rglob('*.py') == path.glob('**/*.py')# The two commands are equivalent.

>>>for e in path.glob('*.py'):
>>>    print(e)

# Prints all the python files only in the CWD and not in subfolders.

>>>for e in path.glob('image_*'):
>>>    print(e)

# Prints all the files that their name starts with image_. Good for dataset loading in machine learning!!

```

### 13.13 Counting files by extension

Very Interesting:

#### Example:

```

>>>import collections# Imports the library collections. We can use it to count for us.

>>>files = [x.suffix for x in docs.iterdir() if path.is_file() and x.suffix]# If (and x.suffix) is not placed
in code, the files that have the extension will be selected. But now, the extension alone are selected.

>>>data = collections.Counter(files)

>>>print(data)# See the result yourself :) .

# If you want to only have the counts and not to have Counter({ at the beginning, do as follows:

>>>for key, val in data.items():
>>>    print(f'{key}: {val}')

```

### 13.14 Read & Write a text file

As below:

**Example:**

```
# To read a text file :

>>>path = Path('1.txt')# Selects 1.txt file in CWD.

>>>content = path.read_text()# Reads the whole text

>>>print(content)# The text will be shown :)

# To write a text file :

>>>path = Path('2.txt')# Selects 2.txt file which is not in CWD and not created yet.

>>>path.touch()# Creates the file.

>>>path.write_text('Hello Friends')# It's done!
```

### 13.15 To Loop over files in a directory

Do as below:

**Example:**

```
>>>pathlist = Path(directory_in_str).glob('**/*.asm')
# Lists whatever .asm file is in the directory (directory_in_str is means you have to insert directory as a
string).

>>>for path in pathlist:

>>>    path_in_str = str(path)
# Since path is not string and is the file itself.

>>>    print(path_in_str)
```

### 13.16 Acknowledgements about pathlib tutorial

For further reading about pathlib, go to : <http://zetcode.com/python/pathlib/>  
It really gave useful information. Thanks zetcode.

## 14 **Tensorflow**

### 14.1 **Importing Libraries**

Below are a few libraries to import

**Example:**

```
>>>import tensorflow as tf
# Main one

>>>from tensorflow import keras
# Keras library

>>>from tensorflow.keras import layers
# Importing Layers for easier handlings

>>>import tensorflow_datasets as tfds
# Importing tfds useful library

>>>import tensorflow_docs as tfdocs
# Documentation Library

>>>import tensorflow_docs.plots as plotting
# tf plotter

>>>from tensorflow.keras import regularizers
# Imports regularizers for avoiding overfitting
```

### 14.2 **General Commands**

A list of general commands:

**Example:**

```
>>>a = tf.Variable([[2,3,4],[1,6,7]])
# Builds a tensorflow variable.

>>>a.shape
# Gives its shape

>>>tf.rank(a)
# Gives the rank (somehow depth of it) of the tensor.

>>>b = tf.reshape(a , [-1,2])
# We specify the important dimension for us and we leave the other one as -1 for tensorflow to calculate what should be placed in there... E.g : A tensor is 6*8 dimensioned so it has got 48 elements. We want a tensor with 2 columns only, we write tf.reshape(tensor , [-1,2]) , so automatically -1 would be  $48/2 = 24$ 

>>>tf.ones([shape])
# Creates a tensor full of ones with the shape.

>>>tf.zeros([shape])
# Creates a tensor full of zeros with the shape.
```

### 14.3 Tensorflow Datasets

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

### 14.4 Keras Library

#### 14.4.1 Keras Datasets

Keras has a couple of datasets up in the cloud and it's possible for you to download and work on them. For getting them, use below commands:

**Example:**

```
>>>(xtrain , ytrain) , (xtest , ytest) = keras.datasets.mnist.load_data()
# Loads the train and test sets.
```

#### 14.4.2 Keras URLs

It's possible to download a dataset from a URL:

**Example:**

```
>>>DATA = keras.utils.get_file('name' , 'URL address')
# Downloads the data file.
```

#### 14.4.3 Keras Model

This procedure is a little bit long.

**Example:**

```
>>>Python code
# Comment

result
```

### 14.5 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```



## 14.6 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

## 14.7 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

## 15 The End

### 15.1 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

### 15.2 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

### 15.3 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

### 15.4 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

### 15.5 Importing Libraries

Below are a few libraries to import

**Example:**

```
>>>Python code
# Comment

result
```

## 16 The End

### 16.1 Acknowledgements

Dear readers. Thank you for reading this article. I will not continue this documentation and the rest of the things, related to each library we discussed, will be upon yourselves.

I wanted to gather a meaningful and handy pack to show you some basics and little bit of everything from most useful libraries in python. Although most of them are not listed like Tensorflow, Pytorch, etc. But, these would cover a good range of python materials.

### 16.2 Final Words

To sum it up, I should encourage you all to get used to read documentations. There are a lot of persons out there gathering too much informations in details to provide you with. This is one of the main reasons I decided to finish this article. It's too much time consuming to work like this. Just get used to work with documentations and grasp the ones that fits you.

*With Best Regards*  
*Mahdi Dor Emami*