READING CSV FILES

Basically, by using the Panda library it is is possible to read CSV_FILES into data frames ,that are complex data types containing table.

Therefore, the first step in order to read CSV_FILES is to import the Panda Library .

```
 8 import pandas as pd
 9
10 df=pd.read_csv( 'us-500.csv', sep=',')
```

Therefore we use the function read_csv in order to read and load the csv file ,that in this case should be placed in the same folder of the source code.

Whereas, there's the chance to put the file in a folder.

In this case the syntax would be:pd.read_csv('namefolder/namefile.csv', sep ',')

The function has 2 arguments :the file that you would like to read and sep that let you state the delimiter of the columns in the csv.file.

In order to see how many rows and columns are contained in the csv.file you need to use the shape function that return the number of rows and columns in the csv file.

```
11 print(df.shape)
```

And the output is the following(which means that the us_500.csv files contain 500 rows and 12 columns):

```
In [1]: runfile('C:/Users/Andrea/.spyder-py3/
firststep.py', wdir='C:/Users/Andrea/.spyder-py3')
(500, 12)
```

Instead if you want to print the first five line of the file ,you have to use the head()function

```
12 print(df.head())
```

VISUALIZATION

The following program makes a visualization

```
from matplotlib import pyplot as plt
x=range(3)
y=[2,6,5]
plt.bar(x,y,0.9)
plt.xticks(x, ('A', 'B', 'C'))
plt.show()
```

In order to make a visualization we need to use the pyplote module belonging to matplot library.

Trough the range()function that returns 0,1,2 it is possible to define the numbers of bars .

The bar function position the bar's x coordinate, the bar's height and finally their width.

Finally ,the xstick assign to the x-axis labels of the bars and the show() method enables to display the graph.

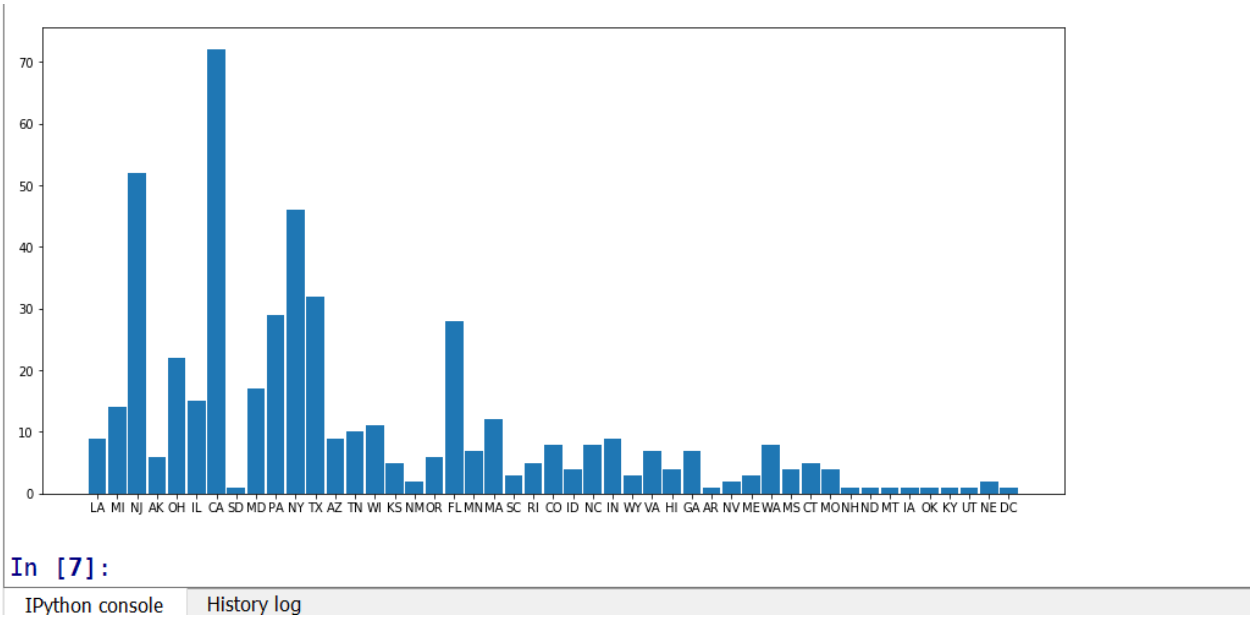
```
 8 from collections import Counter
 9 import pandas as pd
10 import numpy as np
11 from matplotlib import pyplot as plt
12
13 # Read csv-file from disk into dataframe object
14 df = pd.read_csv('data/us-500.csv', sep=',')
15
16 # Count number of entries per state
17 labels, values = zip(*Counter(df['state'].values).items())
18 x = range(len(labels))
19
20 # Set size, create plot, save plot to file and display plot
21 plt.figure(figsize=(15,7)) # Make the plot bigger
22 plt.bar(x, values, 0.9) # Create the plot with a little space between the bars
23 plt.xticks(x, labels)
24 plt.show()
25
```

The following program display a bar ,by reading a csv.file .

Firstly we read the file and then we count the number of states per values.

Then to the x variable is assigned the length of labels that is number of entries per state.

Then the program creates plot, save it to file and diplays it.



VISUALIZATION BY USING NUMPY LIBRARY

Usually data are read into dataframes and after are converted to ndarrays when calculations should take place.

NdArrays is a datatype defined in the Numpy library, and is similar to a list.

An advantage of using ndarray as a datatype is related to the fact that ndarray is multidimensional.

As a consequence ,it can represent tables of data .

The following programs creates an histogram and displays it.

In order to create the histogram we use the hist function that has 3 parameters:

1)an ndarray with data called values created using the numpy library

2)the number of bins

3)the lower and upper range of the bins

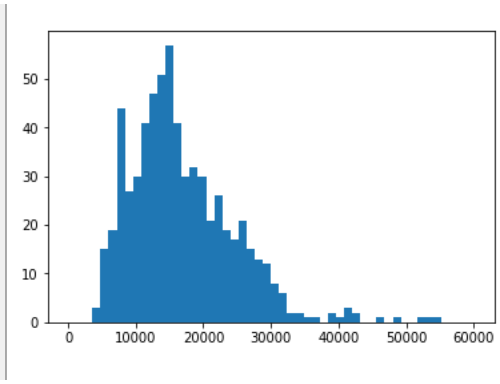
```
 8 import matplotlib.pyplot as plt
 9 import numpy as np
10
11 values = np.array([1,2,3,4,6,7]) # Create ndarray with data
12 plt.hist(values, bins=3, range=(1, 10))
13 plt.show()
14
```

The following program is similar to the one above, as it creates an histograms ,but compared to the previuous it works with real data as read a cvs file.Moreover,in this case some data are deleted trough the drop function that enable us to drop columns.

we needn't data rows with

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Read csv-file from disk into dataframe object
df = pd.read_csv('data/ssb-sq-prices.csv', sep=';', decimal=',', encoding='iso8859-1')

# We only want data rows with category price per square meter and non-null values
df = df[df.Variable == "Kvadratmeterpris (kr)"]
df = df.drop('Variable', axis=1) # We can drop this column now
df = df[df['Value'].notnull()]

# Put values from Value-column into ndarray
values = df['Value'].values

# Make plot
plt.hist(values, bins=50, range=(0, 60000)) # We remove 0-values
plt.show()

```

Below,is the output of the programme.



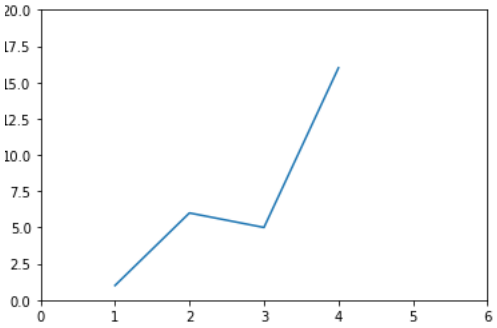
Another interesting graph is the line diagram and in order to plot you need just few lines of codes as you can see from the programme reported below.

```
# -*- coding: utf-8 -*-
plt.plot([1,2,3,4], [1,6,5,16])
plt.axis([0, 6, 0, 20])
plt.show()

```

The plot function takes two arrays in input:the first one contains the value of thex- coordinates of the main point of the line diagram whereas the second contains the values of the y-coordinate.

The axis function set the limits of the x axis and y axis(in this case the first two values are included in the x axis,while the last two values are respectively origin and the end of the y-axis.

OUTPUT:



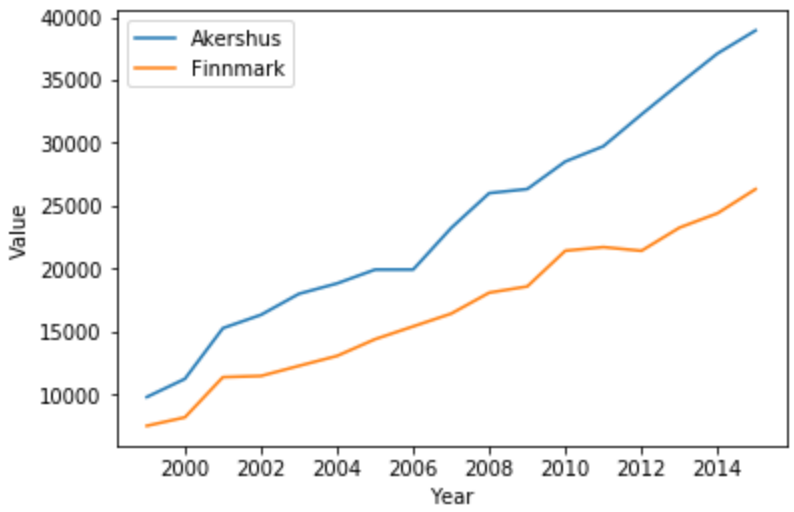
In python,it is also possible to draw line diagrams based on real data.

This is what the following program does by filtering only'enboliger' and printing the first five entris trough the function head.

```
# Read csv-file from disk into dataframe object
df = pd.read_csv('data/ssb-sq-prices.csv', sep=';', decimal=',', encoding='iso8859-1')

df = df[df.Type == "01 Nye eneboliger"] # Filter: only "eneboliger"
df = df.drop('Type', axis=1)  # We can drop this column now
print(df.head()) # View the current top 5 entries

```

This programme creates plots two lines,assigning them also labels.Moreover,labes are also assigned to x axis and y axis.

```
# Make line-diagram with real data
df = df[df.Type == "01 Nye eneboliger"] # Filter: only "eneboliger"
df = df.drop('Type', axis=1)  # We can drop this column now
print(df.head()) # View the current top 5 entries

# Make two different dataframes, using two different filters
df1 = df[df.Region == "02 Akershus"] # Filter: only "Akershus"
df2 = df[df.Region == "20 Finnmark - Finnmárku"] # Filter: only "Finnmark"

# Make 2 line-plots on same visualizations (they will be coloured separately)
plt.plot(df1['Date'].values, df1['Value'].values, label='Akershus')
plt.plot(df2['Date'].values, df2['Value'].values, label='Finnmark')
plt.legend() # Plot legends (the two labels)
plt.xlabel('Year') # Set x-axis text
plt.ylabel('Value') # Set y-axis text
plt.show() # Display plot
```



SIMPLE STATISTICS

In order to use statistical function you need to remove null values from datasets, as in the following program

```
import pandas as pd
import numpy as np

df = pd.read_csv('data/ssb-sq-prices.csv', sep=';', decimal=',', encoding='iso8859-1')
df = df[df.Variable == "Kvadratmeterpris (kr)"] # Use only rows with 'Variable' == given value
df = df[df['Value'].notnull()] # Filter out all null-values
data = df['Value'].values # Put the values from the 'Value'-column into an ndarray
```

```
import numpy as np
import pandas as pd

df = pd.read_csv('data/ssb-sq-prices.csv', sep=';', decimal=',', encoding='iso8859-1')
df = df[df.Variable == "Kvadratmeterpris (kr)"] # Use only rows with 'Variable' == given value
df = df[df['Value'].notnull()] # Filter out all null-values
data = df['Value'].values # Put the values from the 'Value'-column into an ndarray

mean = np.mean(data)
print("Mean: " + str(mean))




average = np.average([2,3,4], weights=[0.5, 0.25, 0.25])
print("Weighted average: " + str(average))




median = np.median(data)
print("Median: " + str(median))



print("Variance: " + str(np.var(data)))

print("Standard deviation: " + str(np.std(data)))


print("Min: " + str(np.min(data)))
print("Max: " + str(np.max(data)))

```

Note that in order to print the result we need the strin function that turns the object enclosed in parantesy into a string.

The following program calculates mean,weighted average,median,variance,standard deviation,min-max.

```
In [17]: runfile('C:/Users/Andrea/.spyder-py3/visualization2.py', wdir='C:/Users/Andrea/.spyder-py3')
Mean: 16898.97360248447
Weighted average: 2.75
Median: 15332.5
Variance: 60573642.96358888
Standard deviation: 7782.90710233579
Min: 4660.0
Max: 54567.0
```