SECOND TASK DESCRIPTION

FIRST PART

(CIBELLI MICHELE, FLORES GIACOMO, PICCARO FILIPPO)

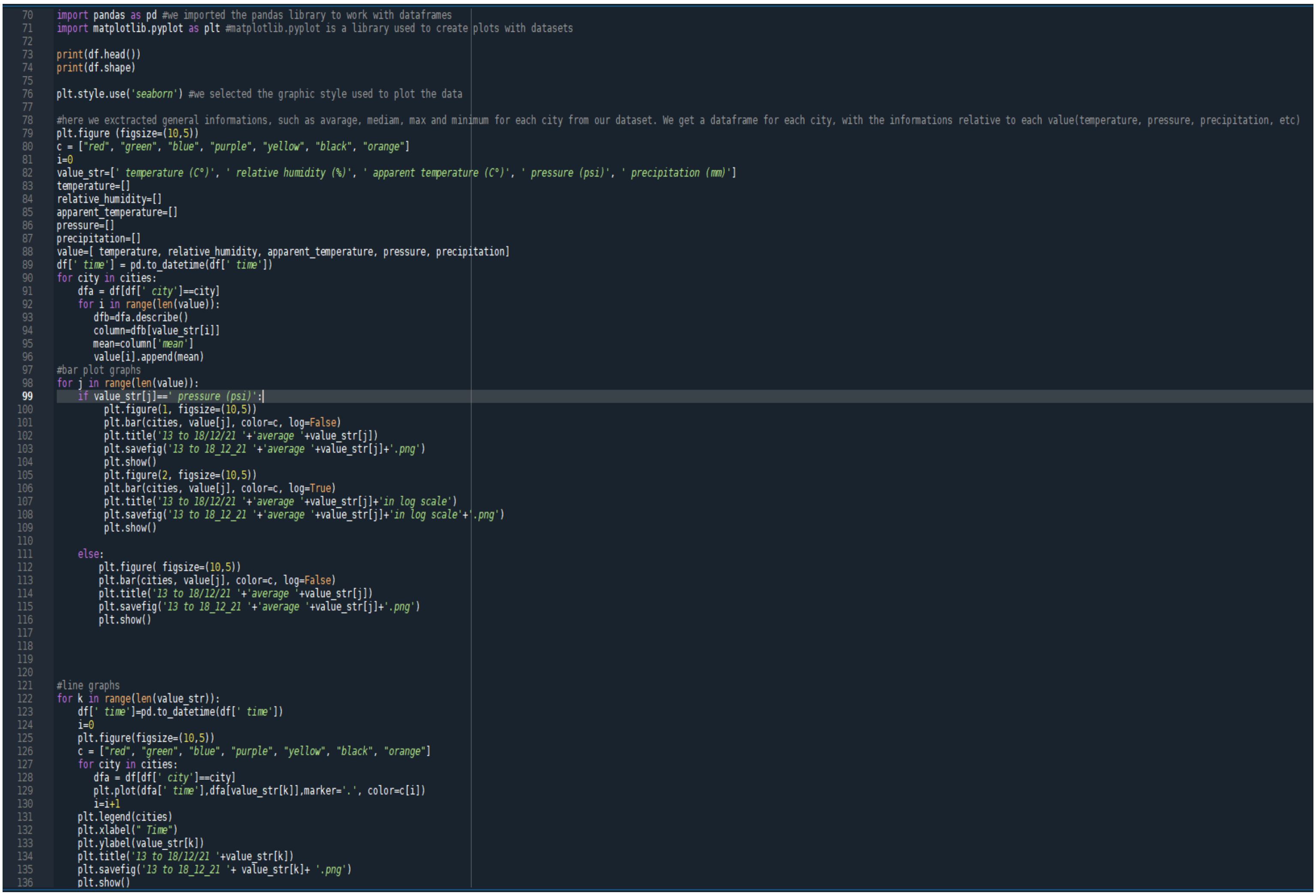
PYTHON CODE

With this program we have processed meteorological data previously extracted with the use of an API in the first task. The structure of our code is divided into three modules:

1) In the first module, we processed a dataset composed of data extracted during this period, and therefore related to current weather forecasts. We decided to plot them using bar and linear graphs.

2) In the second module, we decided to process data extracted from an older dataset, i.e. weather forecasts made in November. The graphs obtained are the same as in the first module.

3) In the third and last module we have done the same thing as the previous ones, but we have obtained graphs that are the union of the bar graphs generated previously: this allows us to compare the data obtained in the extraction of a month ago and that of today

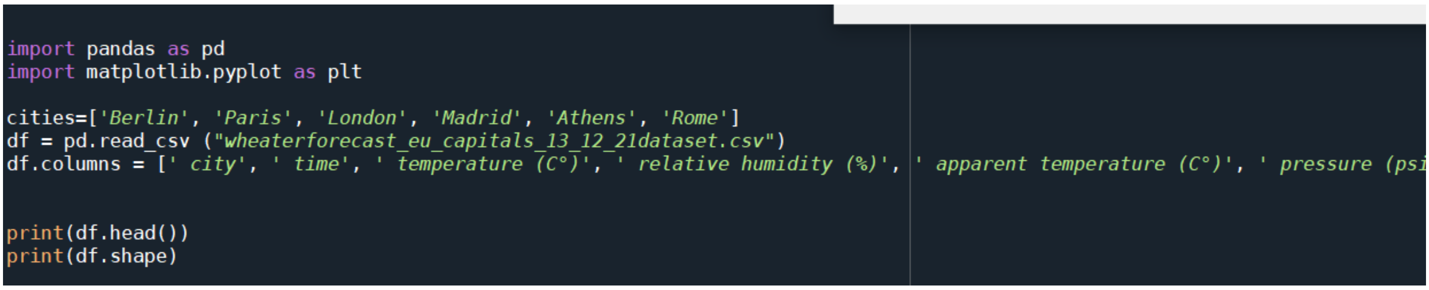


```python
import pandas as pd #we imported the pandas library to work with dataframes
import matplotlib.pyplot as plt #matplotlib.pyplot is a library used to create plots with datasets

print(df.head())
print(df.shape)

plt.style.use('seaborn') #we selected the graphic style used to plot the data

#here we exctracted general informations, such as avarage, mediam, max and minimum for each city from our dataset. We get a dataframe for each city, with the informations relative to each value(temperature, pressure, precipitation, etc)
plt.figure (figsize=(10,5))
c = ["red", "green", "blue", "purple", "yellow", "black", "orange"]
i=0
value_str=[' temperature (C°)', ' relative humidity (%)', ' apparent temperature (C°)', ' pressure (psi)', ' precipitation (mm)']
temperature=[]
relative_humidity=[]
apparent_temperature=[]
pressure=[]
precipitation=[]
value=[ temperature, relative_humidity, apparent_temperature, pressure, precipitation]
df[' time'] = pd.to_datetime(df[' time'])
for city in cities:
    dfa = df[df[' city']==city]
    for i in range(len(value)):
       dfb=dfa.describe()
       column=dfb[value_str[i]]
       mean=column['mean']
       value[i].append(mean)
#bar plot graphs
for j in range(len(value)):
    if value_str[j]==' pressure (psi)':
         plt.figure(1, figsize=(10,5))
         plt.bar(cities, value[j], color=c, log=False)
         plt.title('13 to 18/12/21 '+'average '+value_str[j])
         plt.savefig('13 to 18_12_21 '+'average '+value_str[j]+'.png')
         plt.show()
         plt.figure(2, figsize=(10,5))
         plt.bar(cities, value[j], color=c, log=True)
         plt.title('13 to 18/12/21 '+'average '+value_str[j]+'in log scale')
         plt.savefig('13 to 18_12_21 '+'average '+value_str[j]+'in log scale'+'.png')
         plt.show()

    else:
        plt.figure( figsize=(10,5))
        plt.bar(cities, value[j], color=c, log=False)
        plt.title('13 to 18/12/21 '+'average '+value_str[j])
        plt.savefig('13 to 18_12_21 '+'average '+value_str[j]+'.png')
        plt.show()




#line graphs
for k in range(len(value_str)):
    df[' time']=pd.to_datetime(df[' time'])
    i=0
    plt.figure(figsize=(10,5))
    c = ["red", "green", "blue", "purple", "yellow", "black", "orange"]
    for city in cities:
       dfa = df[df[' city']==city]
       plt.plot(dfa[' time'],dfa[value_str[k]],marker='.', color=c[i])
       i=i+1
    plt.legend(cities)
    plt.xlabel(" Time")
    plt.ylabel(value_str[k])
    plt.title('13 to 18/12/21 '+value_str[k])
    plt.savefig('13 to 18_12_21 '+ value_str[k]+ '.png')
    plt.show()
```

This is the code of the first part.

As we said before in this part of the code, we elaborated the dataset and plotted it to get different types of graphs. Specifically in the code, the first thing we have done was to import the necessary libraries: we imported the pandas library to work with data frame, and matplotlib.pyplot library in order to create plots with our datasets.

We created a list with the cities names in the order witch they appear in the dataset and then we extracted the data from the respective csv file and converted it in data frame, defining the names of the columns.



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

cities=['Berlin', 'Paris', 'London', 'Madrid', 'Athens', 'Rome']
df = pd.read_csv ("wheaterforecast_eu_capitals_13_12_21dataset.csv")
df.columns = [' city', ' time', ' temperature (C°)', ' relative humidity (%)', ' apparent temperature (C°)', ' pressure (psi


print(df.head())
print(df.shape)
```

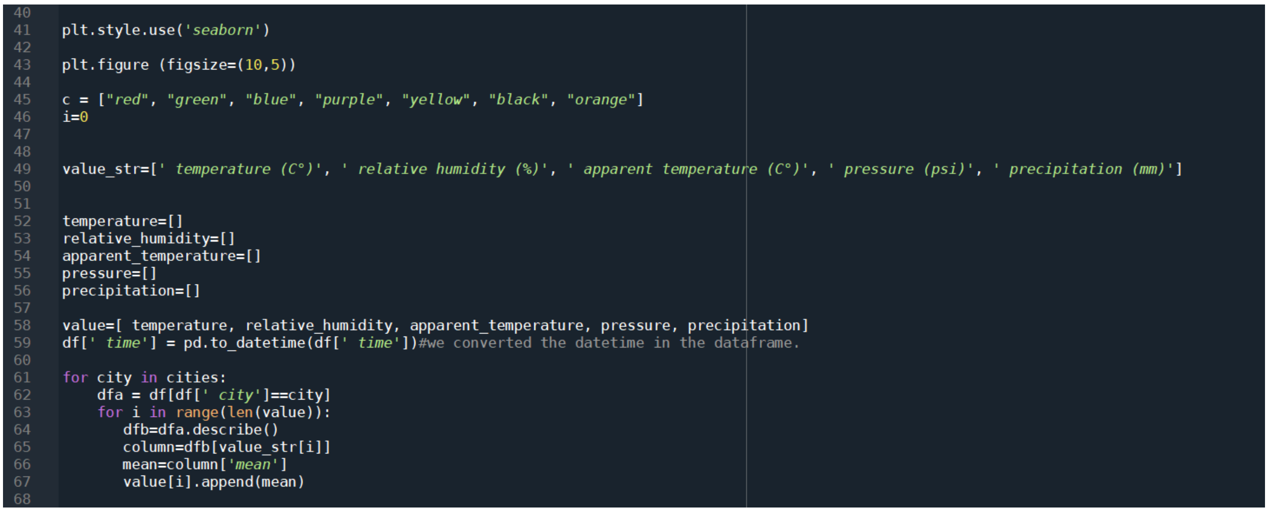
We selected the graphic style used to plot the data, and then we extracted general information, such as average, median, max and minimum values for each city from our dataset.

To do so we had to elaborate and extract the data form our dataframe: we decide to extract the information relative to the average of each value in each city (average temperature, average pressure, average precipitation, etc)

The first thing done we at first created a list with the names of the values we extracted, that we will use this for calling the desired data and to give names to the plot created.

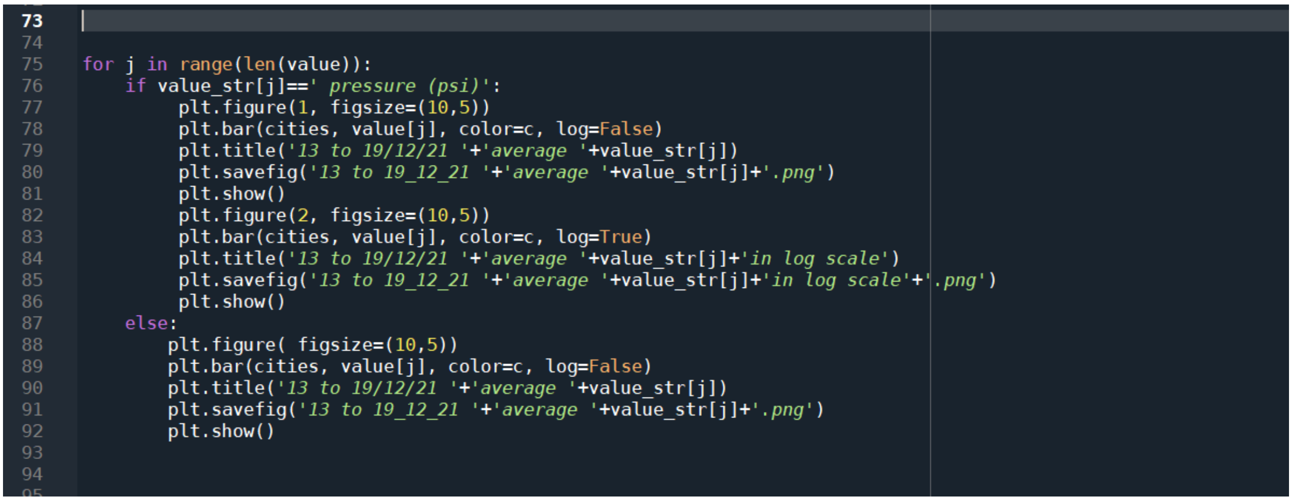
After that we created an empty list for each of the values (temperature, humidity, apparent temperature, pressure and precipitations). We then took these empty lists and put them in another list called value.

Then we created a for loop inside another for loop. With this two for loops we firstly selected the data relative to a single city, for each loop, then we extracted general information, such as avarage, mediam, max and minimum for each city from our dataset. Afterwards we selected only the averages for each value in each city and put it in the relative list of the relative value.



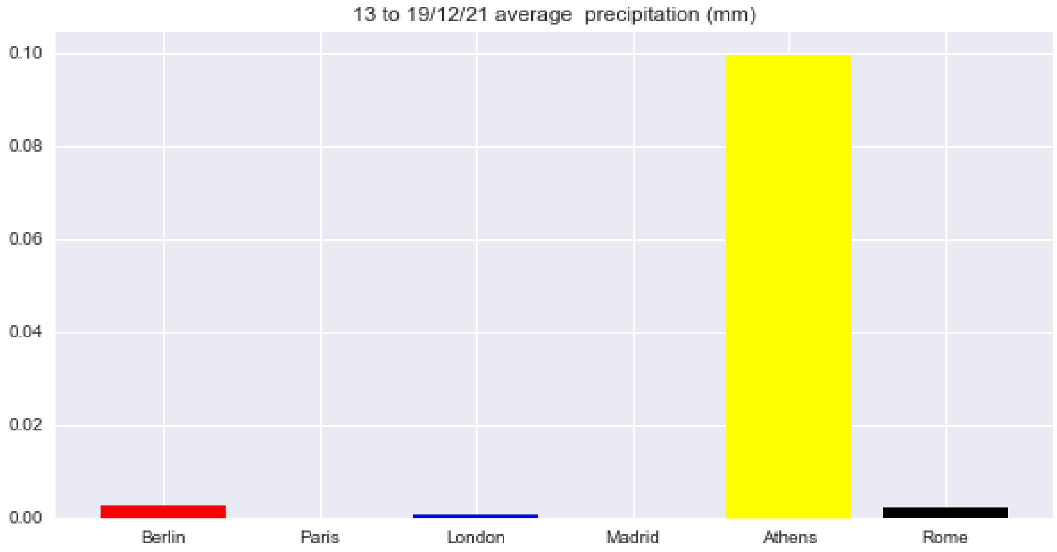
```

plt.style.use('seaborn')

plt.figure (figsize=(10,5))

c = ["red", "green", "blue", "purple", "yellow", "black", "orange"]
i=0


value_str=[' temperature (C°)', ' relative humidity (%)', ' apparent temperature (C°)', ' pressure (psi)', ' precipitation (mm)']


temperature=[]
relative_humidity=[]
apparent_temperature=[]
pressure=[]
precipitation=[]

value=[ temperature, relative_humidity, apparent_temperature, pressure, precipitation]
df[' time'] = pd.to_datetime(df[' time'])#we converted the datetime in the dataframe.

for city in cities:
    dfa = df[df[' city']==city]
    for i in range(len(value)):
       dfb=dfa.describe()
       column=dfb[value_str[i]]
       mean=column['mean']
       value[i].append(mean)

```

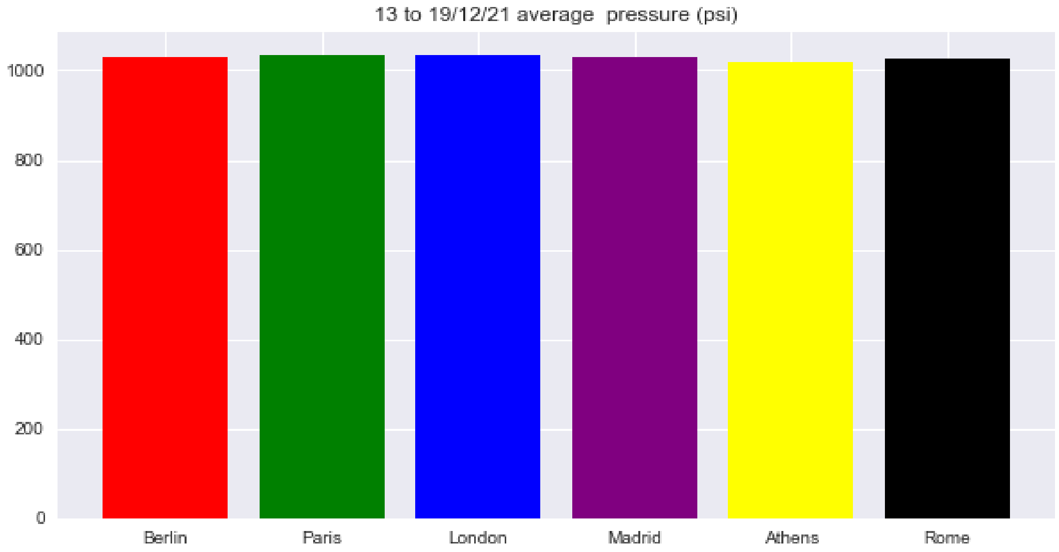
In this part instead we created a bar chart graph for each average value using a for loop to automate the process. In the bar chart generated, each bin is referred to a city. For the average pressure we created two different bar charts: one with the average pressure data plotted with the normal y axis values, and one with the same value but in logarithmic scale.

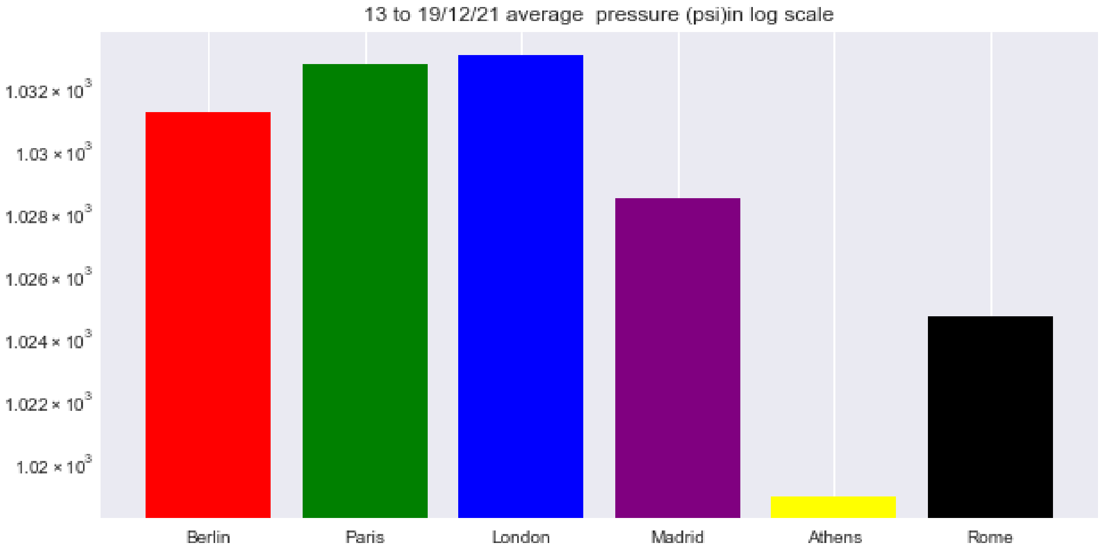


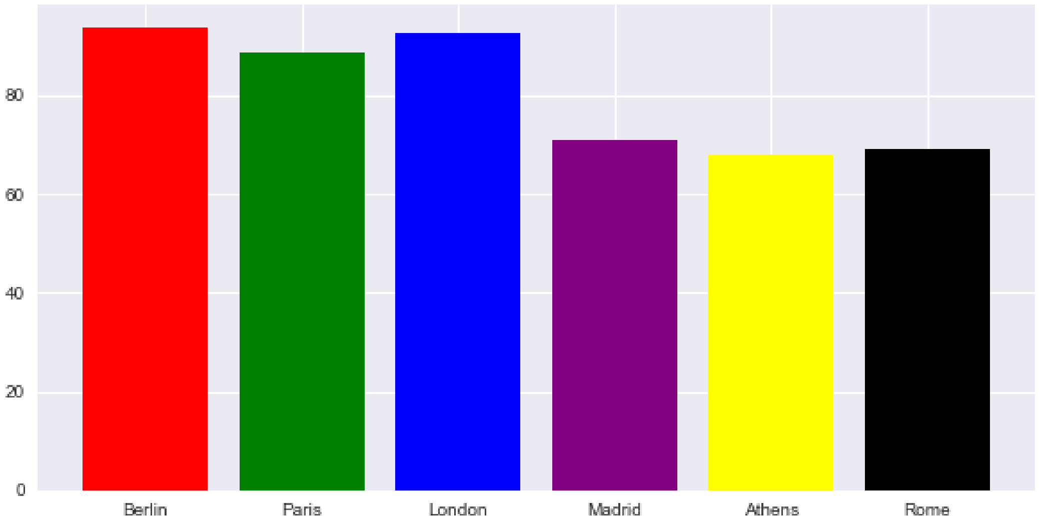
```

for j in range(len(value)):
    if value_str[j]==' pressure (psi)':
         plt.figure(1, figsize=(10,5))
         plt.bar(cities, value[j], color=c, log=False)
         plt.title('13 to 19/12/21 '+'average '+value_str[j])
         plt.savefig('13 to 19_12_21 '+'average '+value_str[j]+'.png')
         plt.show()
         plt.figure(2, figsize=(10,5))
         plt.bar(cities, value[j], color=c, log=True)
         plt.title('13 to 19/12/21 '+'average '+value_str[j]+'in log scale')
         plt.savefig('13 to 19_12_21 '+'average '+value_str[j]+'in log scale'+'.png')
         plt.show()
    else:
        plt.figure( figsize=(10,5))
        plt.bar(cities, value[j], color=c, log=False)
        plt.title('13 to 19/12/21 '+'average '+value_str[j])
        plt.savefig('13 to 19_12_21 '+'average '+value_str[j]+'.png')
        plt.show()
```

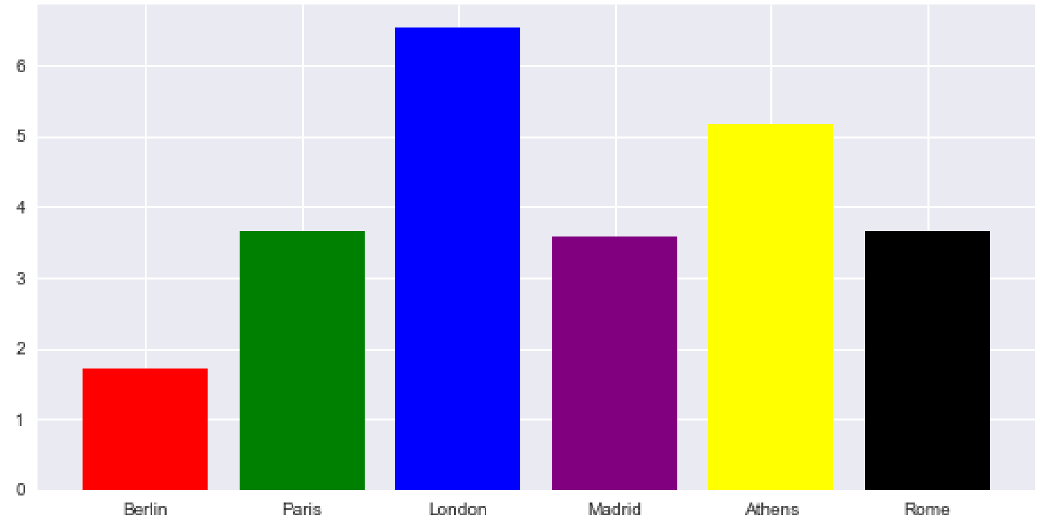
The resulting graphs are the following:

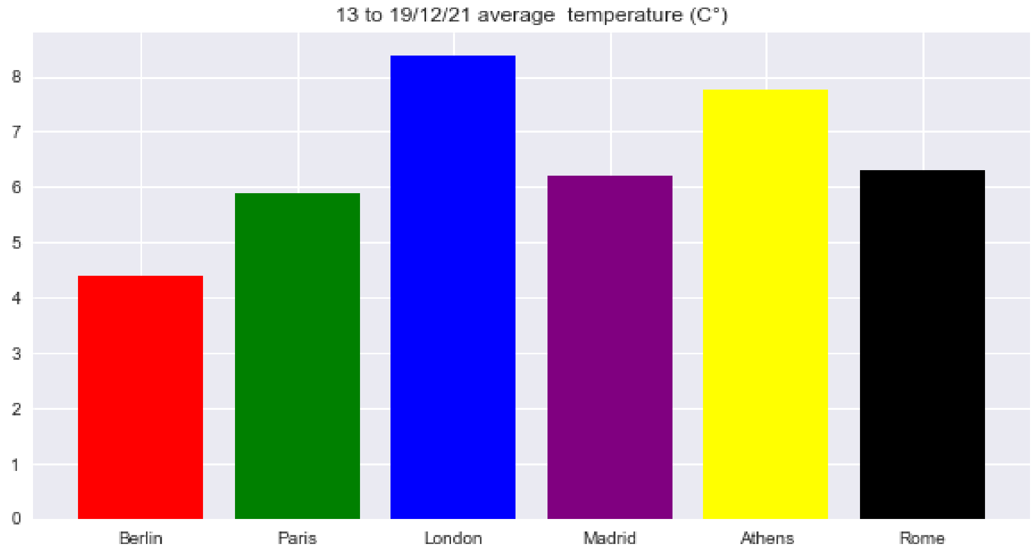



13 to 19/12/21 average pressure (psi)
1000
800
600
400
200
0
Berlin
Paris
London
Madrid
Athens
Rome

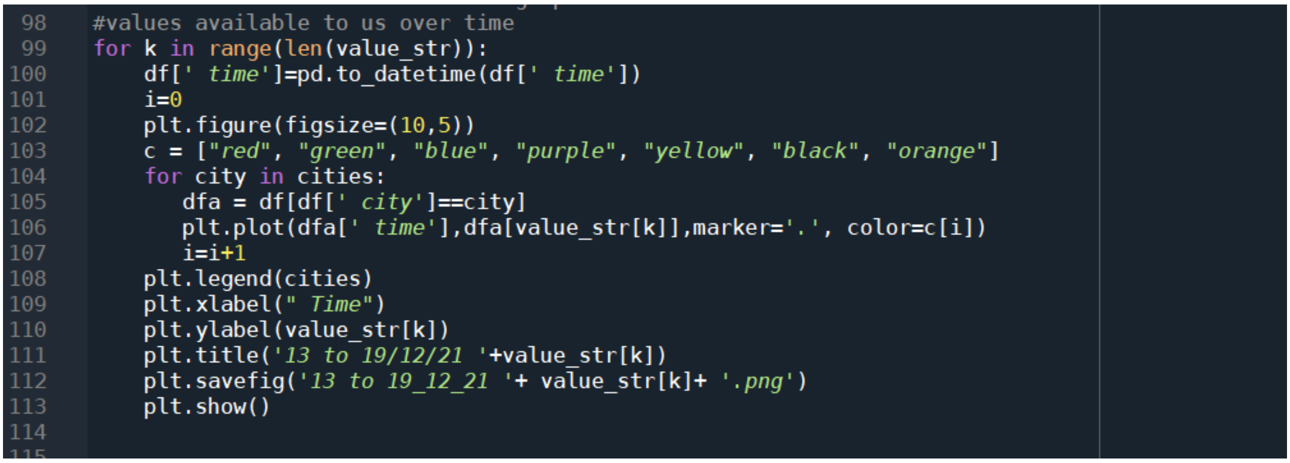

13 to 19/12/21 average pressure (psi)in log scale
1.032 × 10^3
1.03 × 10^3
1.028 × 10^3
1.026 × 10^3
1.024 × 10^3
1.022 × 10^3
1.02 × 10^3
Berlin
Paris
London
Madrid
Athens
Rome


13 to 19/12/21 average relative humidity (%)


13 to 19/12/21 average apparent temperature (C°)

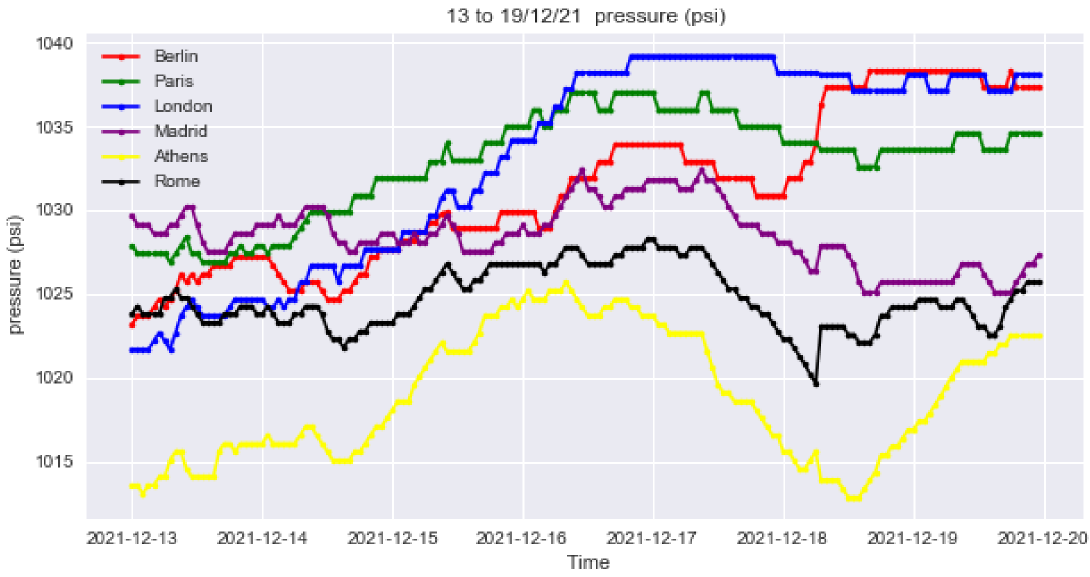


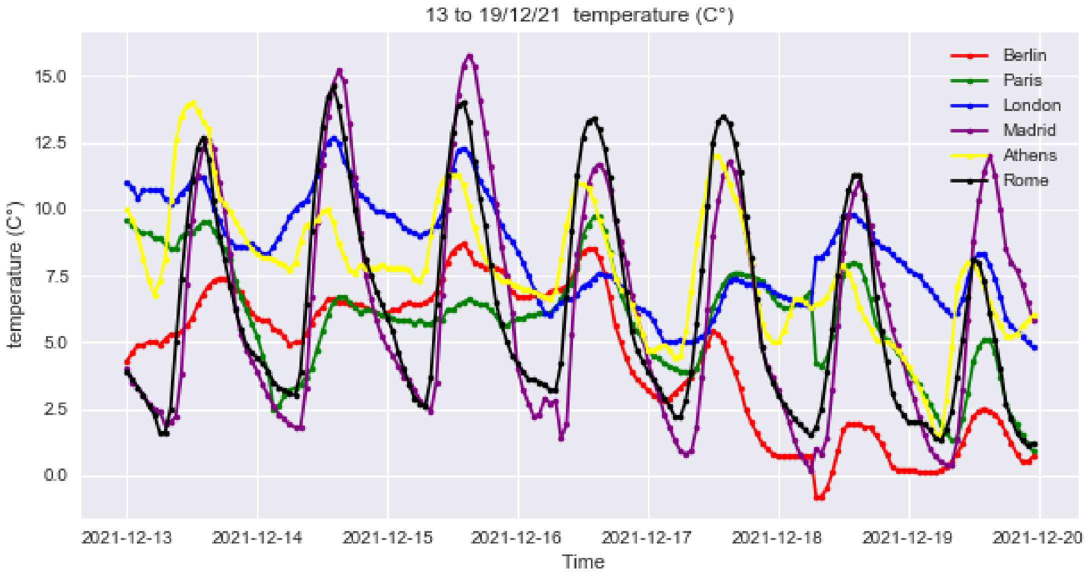
In the final part of the first module we, instead, created a for loop in order to generate linear graphs that trace the trend of the available values, plotted over the considered time.

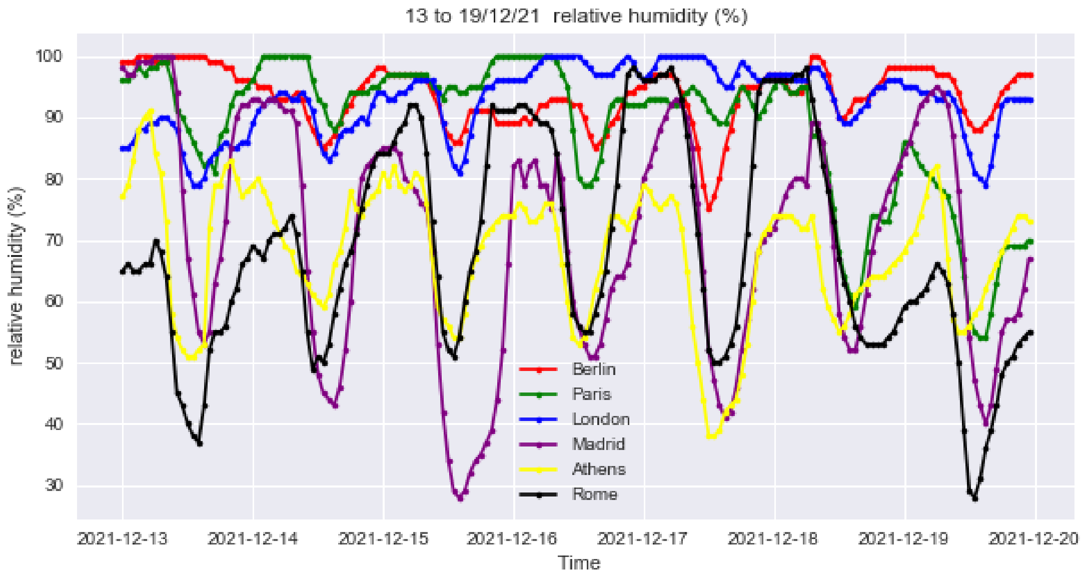


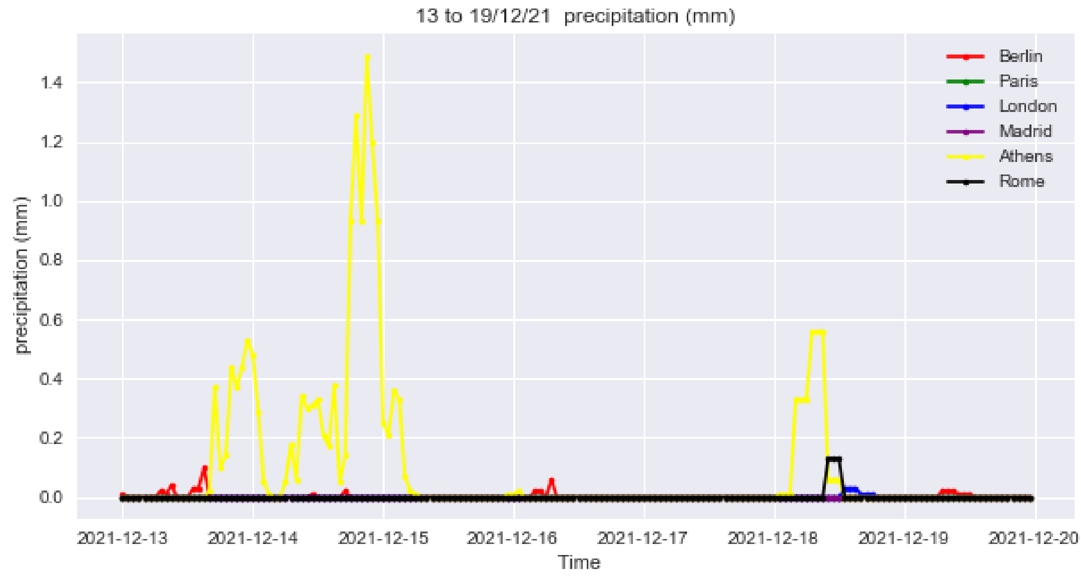
```
#values available to us over time
for k in range(len(value_str)):
    df[' time']=pd.to_datetime(df[' time'])
    i=0
    plt.figure(figsize=(10,5))
    c = ["red", "green", "blue", "purple", "yellow", "black", "orange"]
    for city in cities:
       dfa = df[df[' city']==city]
       plt.plot(dfa[' time'],dfa[value_str[k]],marker='.', color=c[i])
       i=i+1
    plt.legend(cities)
    plt.xlabel(" Time")
    plt.ylabel(value_str[k])
    plt.title('13 to 19/12/21 '+value_str[k])
    plt.savefig('13 to 19_12_21 '+ value_str[k]+ '.png')
    plt.show()
```

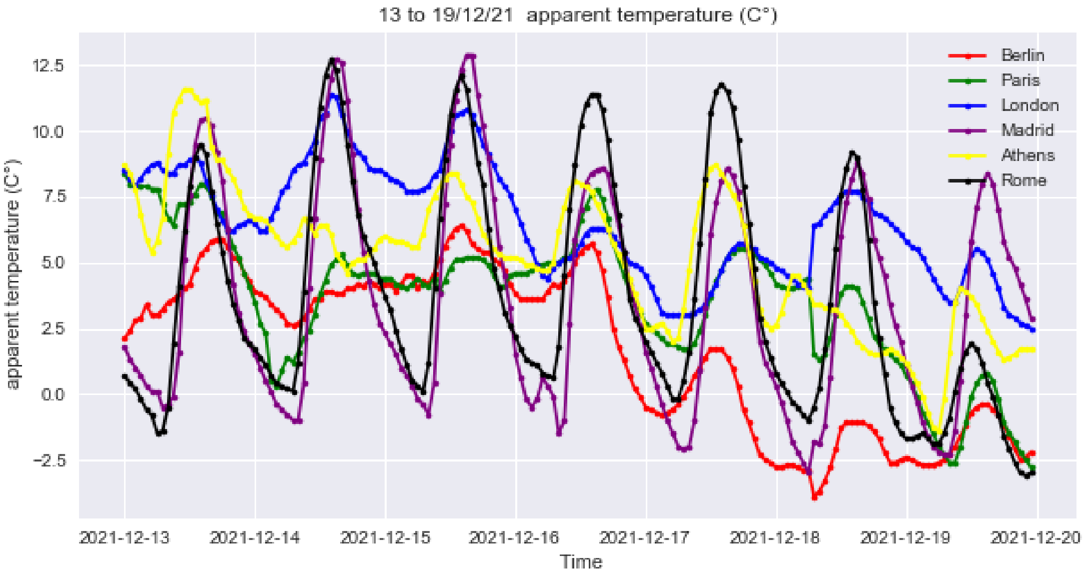
The resulting linear graphs that we got in output are the following. As we can se we get the trend of different meteorological indicators in different cities.





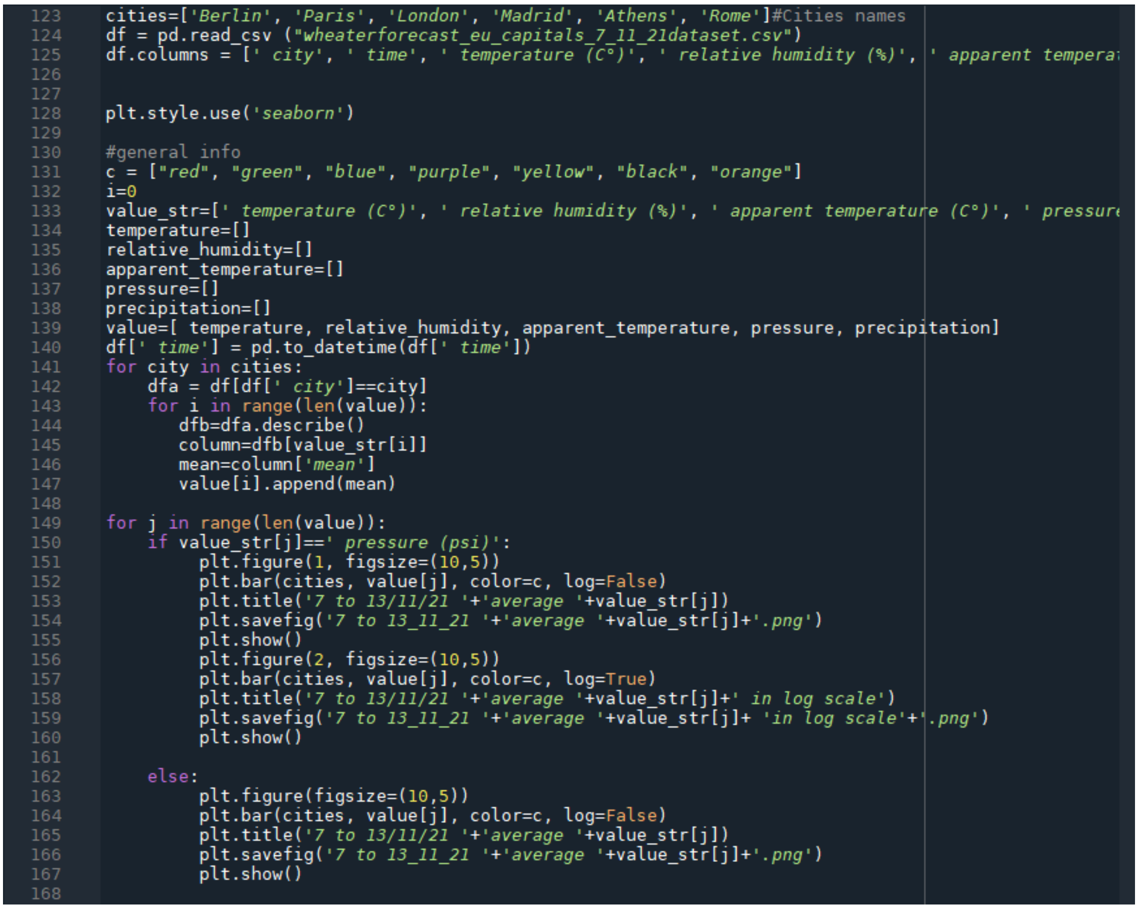

13 to 19/12/21 relative humidity (%)
relative humidity (%)
100
90
80
70
60
50
40
30
2021-12-13
2021-12-14
2021-12-15
2021-12-16
2021-12-17
2021-12-18
2021-12-19
2021-12-20
Time
Berlin
Paris
London
Madrid
Athens
Rome


13 to 19/12/21 precipitation (mm)
precipitation (mm)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2021-12-13
2021-12-14
2021-12-15
2021-12-16
2021-12-17
2021-12-18
2021-12-19
2021-12-20
Time
Berlin
Paris
London
Madrid
Athens
Rome

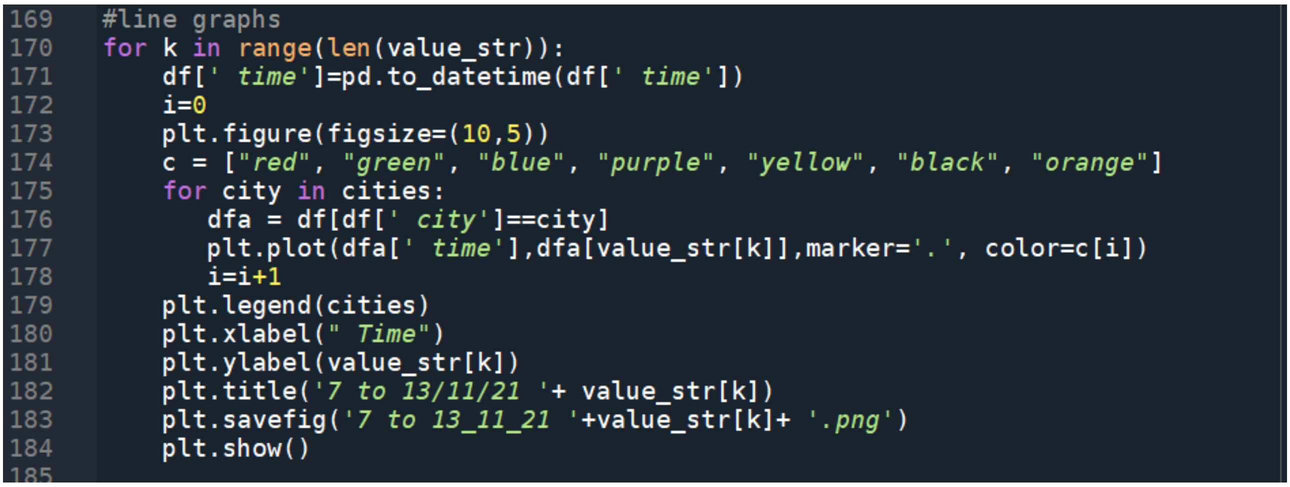


In the second module of the programme we have basically done the same as before, this time, instead, we used an older dataset: the one extracted in November to complete the first task.

The following is the obtained code:

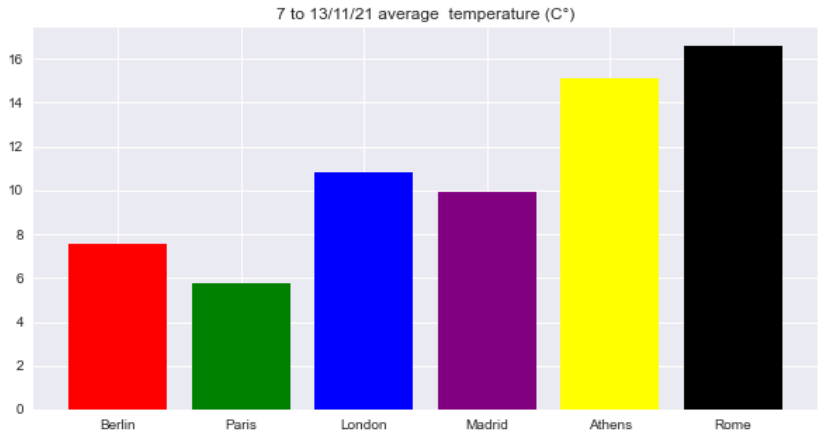


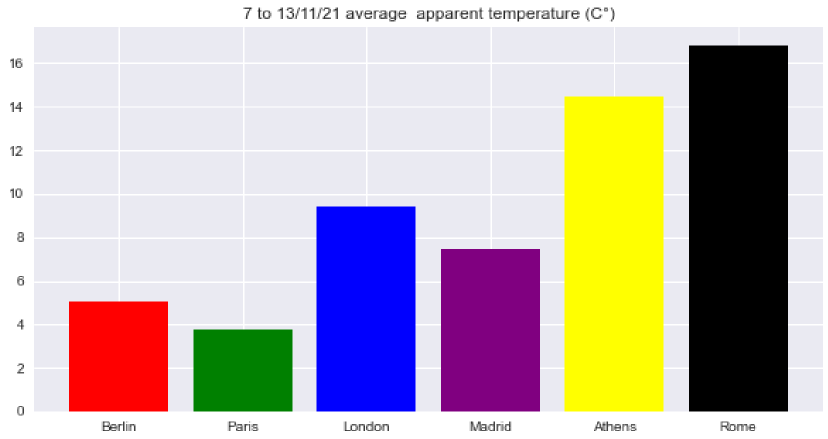
```
cities=['Berlin', 'Paris', 'London', 'Madrid', 'Athens', 'Rome']#Cities names
df = pd.read_csv ("wheaterforecast_eu_capitals_7_11_21dataset.csv")
df.columns = [' city', ' time', ' temperature (C°)', ' relative humidity (%)', ' apparent tempera


plt.style.use('seaborn')

#general info
c = ["red", "green", "blue", "purple", "yellow", "black", "orange"]
i=0
value_str=[' temperature (C°)', ' relative humidity (%)', ' apparent temperature (C°)', ' pressure
temperature=[]
relative_humidity=[]
apparent_temperature=[]
pressure=[]
precipitation=[]
value=[ temperature, relative_humidity, apparent_temperature, pressure, precipitation]
df[' time'] = pd.to_datetime(df[' time'])
for city in cities:
    dfa = df[df[' city']==city]
    for i in range(len(value)):
       dfb=dfa.describe()
       column=dfb[value_str[i]]
       mean=column['mean']
       value[i].append(mean)

for j in range(len(value)):
    if value_str[j]==' pressure (psi)':
         plt.figure(1, figsize=(10,5))
         plt.bar(cities, value[j], color=c, log=False)
         plt.title('7 to 13/11/21 '+'average '+value_str[j])
         plt.savefig('7 to 13_11_21 '+'average '+value_str[j]+'.png')
         plt.show()
         plt.figure(2, figsize=(10,5))
         plt.bar(cities, value[j], color=c, log=True)
         plt.title('7 to 13/11/21 '+'average '+value_str[j]+' in log scale')
         plt.savefig('7 to 13_11_21 '+'average '+value_str[j]+ 'in log scale'+'.png')
         plt.show()

    else:
         plt.figure(figsize=(10,5))
         plt.bar(cities, value[j], color=c, log=False)
         plt.title('7 to 13/11/21 '+'average '+value_str[j])
         plt.savefig('7 to 13_11_21 '+'average '+value_str[j]+'.png')
         plt.show()

```

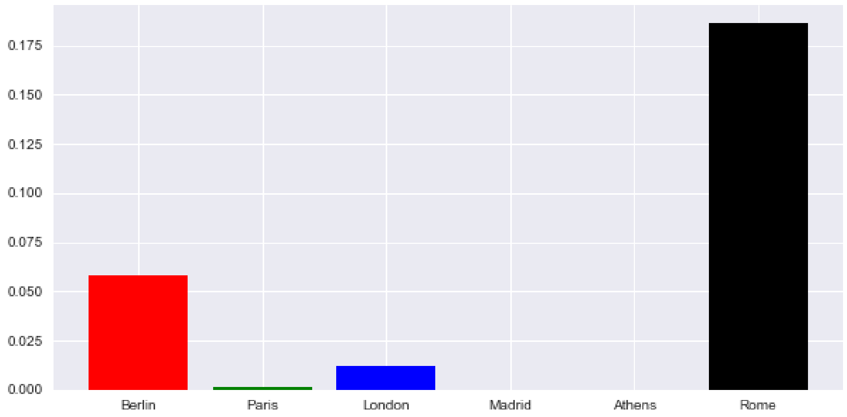


```python
#line graphs
for k in range(len(value_str)):
    df[' time']=pd.to_datetime(df[' time'])
    i=0
    plt.figure(figsize=(10,5))
    c = ["red", "green", "blue", "purple", "yellow", "black", "orange"]
    for city in cities:
       dfa = df[df[' city']==city]
       plt.plot(dfa[' time'],dfa[value_str[k]],marker='.', color=c[i])
       i=i+1
    plt.legend(cities)
    plt.xlabel(" Time")
    plt.ylabel(value_str[k])
    plt.title('7 to 13/11/21 '+ value_str[k])
    plt.savefig('7 to 13_11_21 '+value_str[k]+ '.png')
    plt.show()
```

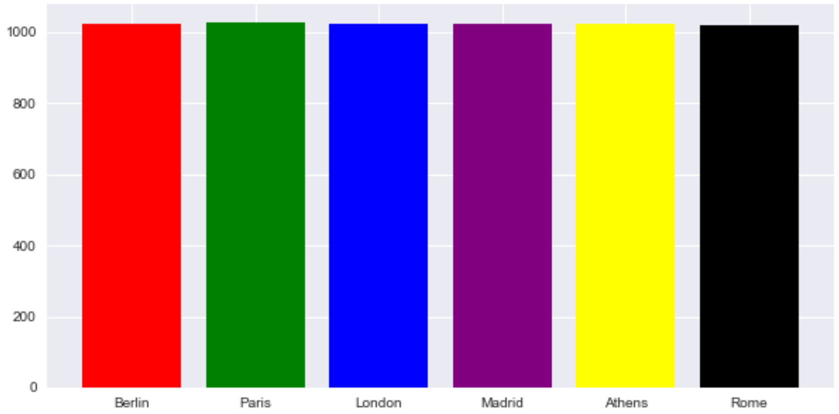
The resulting plots are the following:



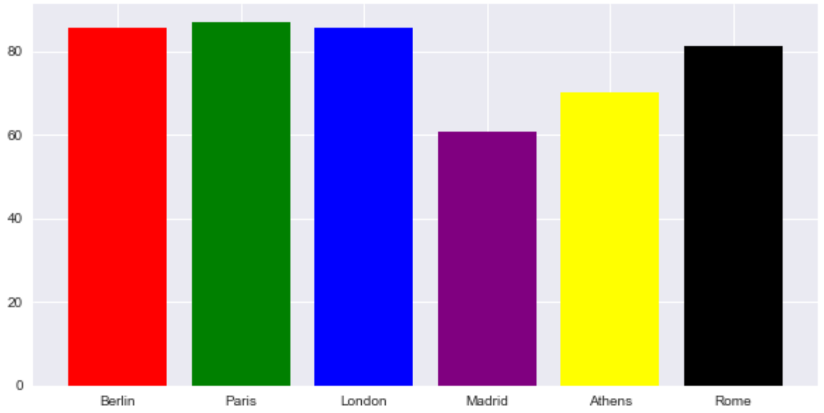




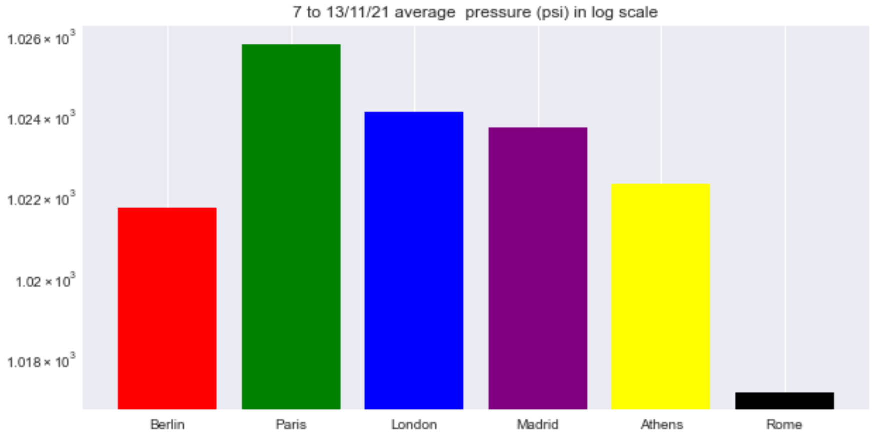
7 to 13/11/21 average precipitation (mm)

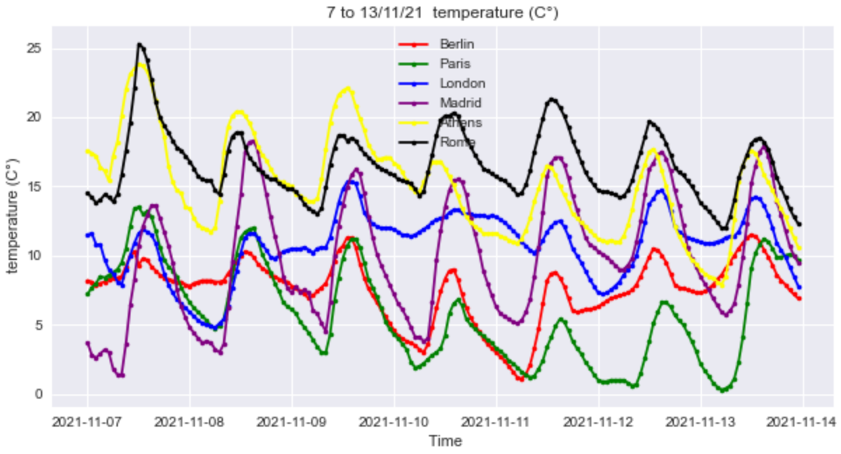


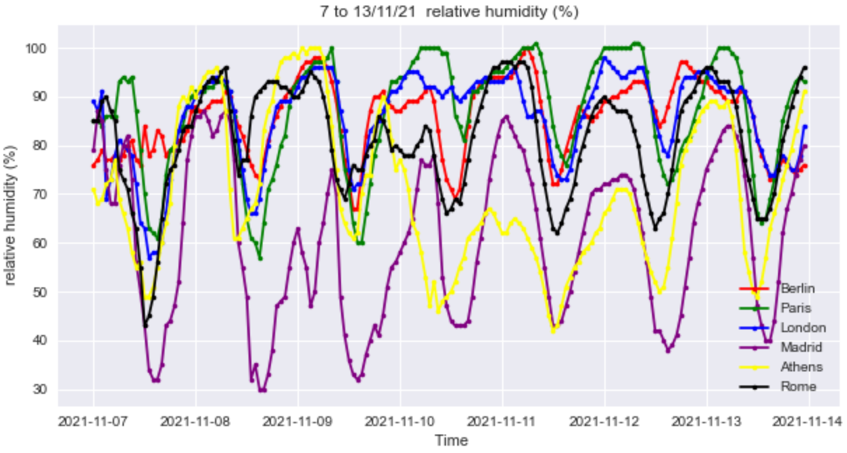
7 to 13/11/21 average pressure (psi)

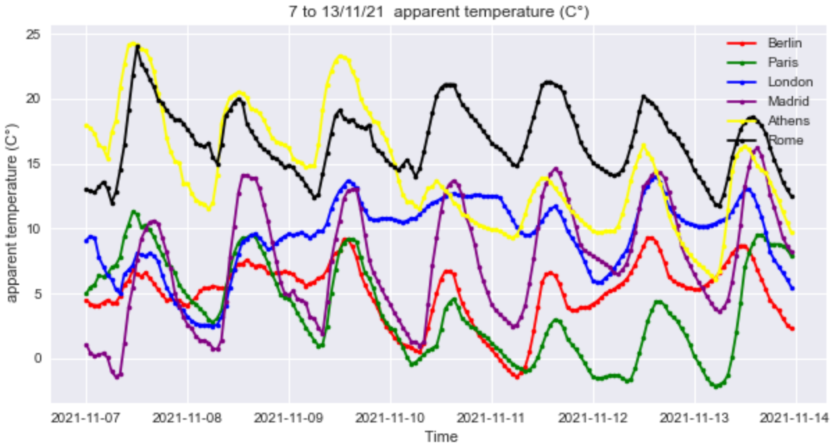


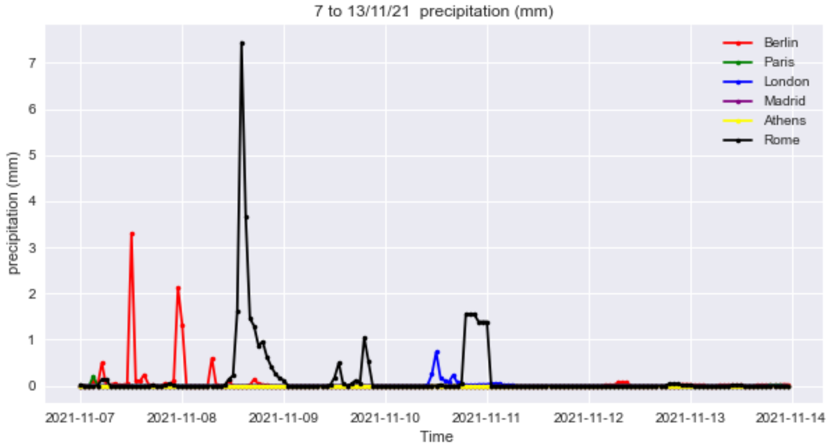
7 to 13/11/21 average relative humidity (%)

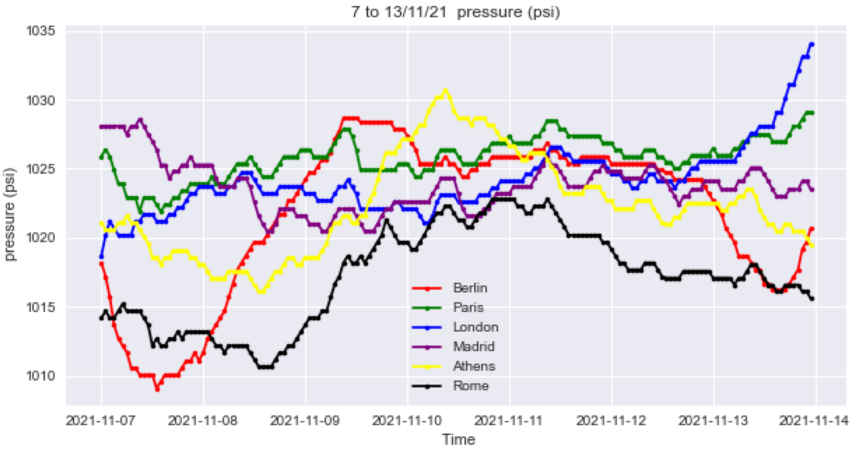

7 to 13/11/21 average pressure (psi) in log scale
1.026 × 10^3
1.024 × 10^3
1.022 × 10^3
1.02 × 10^3
1.018 × 10^3
Berlin
Paris
London
Madrid
Athens
Rome


7 to 13/11/21 temperature (C°)
Berlin
Paris
London
Madrid
Athens
Rome
temperature (C°)
25
20
15
10
5
0
2021-11-07
2021-11-08
2021-11-09
2021-11-10
2021-11-11
2021-11-12
2021-11-13
2021-11-14
Time


7 to 13/11/21 relative humidity (%)
relative humidity (%)
100
90
80
70
60
50
40
30
Berlin
Paris
London
Madrid
Athens
Rome
2021-11-07
2021-11-08
2021-11-09
2021-11-10
2021-11-11
2021-11-12
2021-11-13
2021-11-14
Time

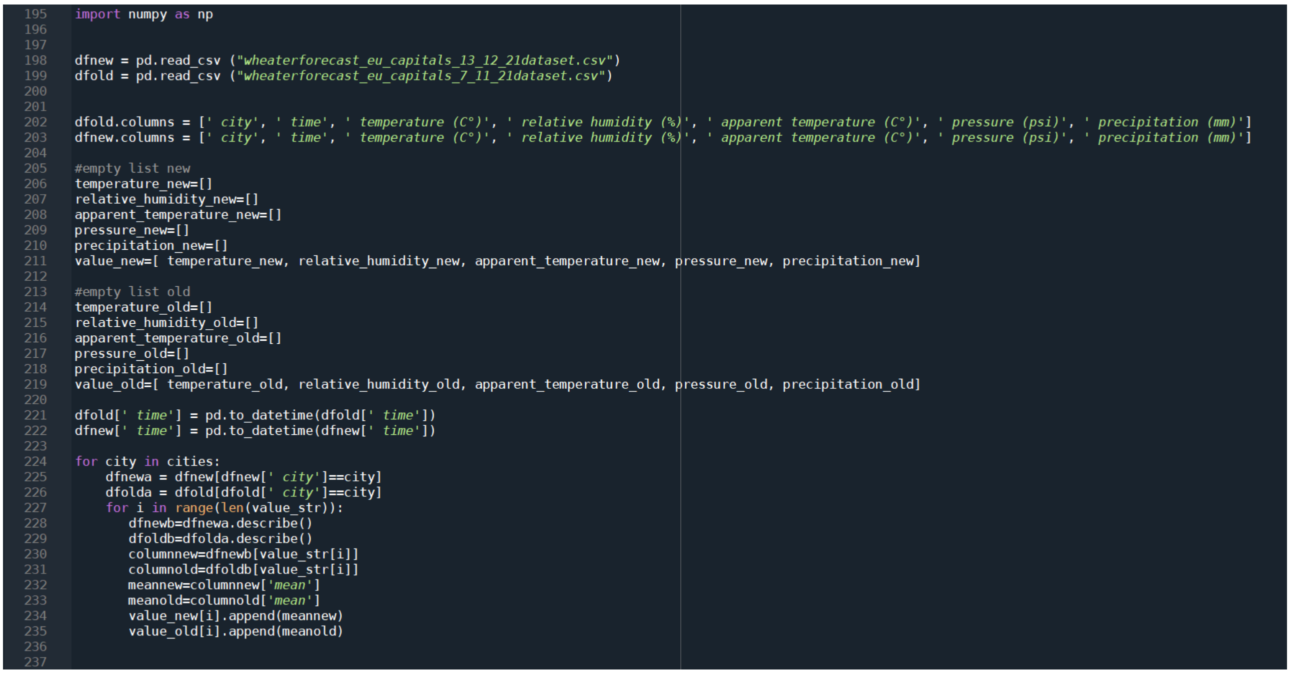

7 to 13/11/21 apparent temperature (C°)
apparent temperature (C°)
25
20
15
10
5
0
2021-11-07
2021-11-08
2021-11-09
2021-11-10
2021-11-11
2021-11-12
2021-11-13
2021-11-14
Time
Berlin
Paris
London
Madrid
Athens
Rome


7 to 13/11/21 precipitation (mm)
precipitation (mm)
7
6
5
4
3
2
1
0
2021-11-07
2021-11-08
2021-11-09
2021-11-10
2021-11-11
2021-11-12
2021-11-13
2021-11-14
Time
Berlin
Paris
London
Madrid
Athens
Rome

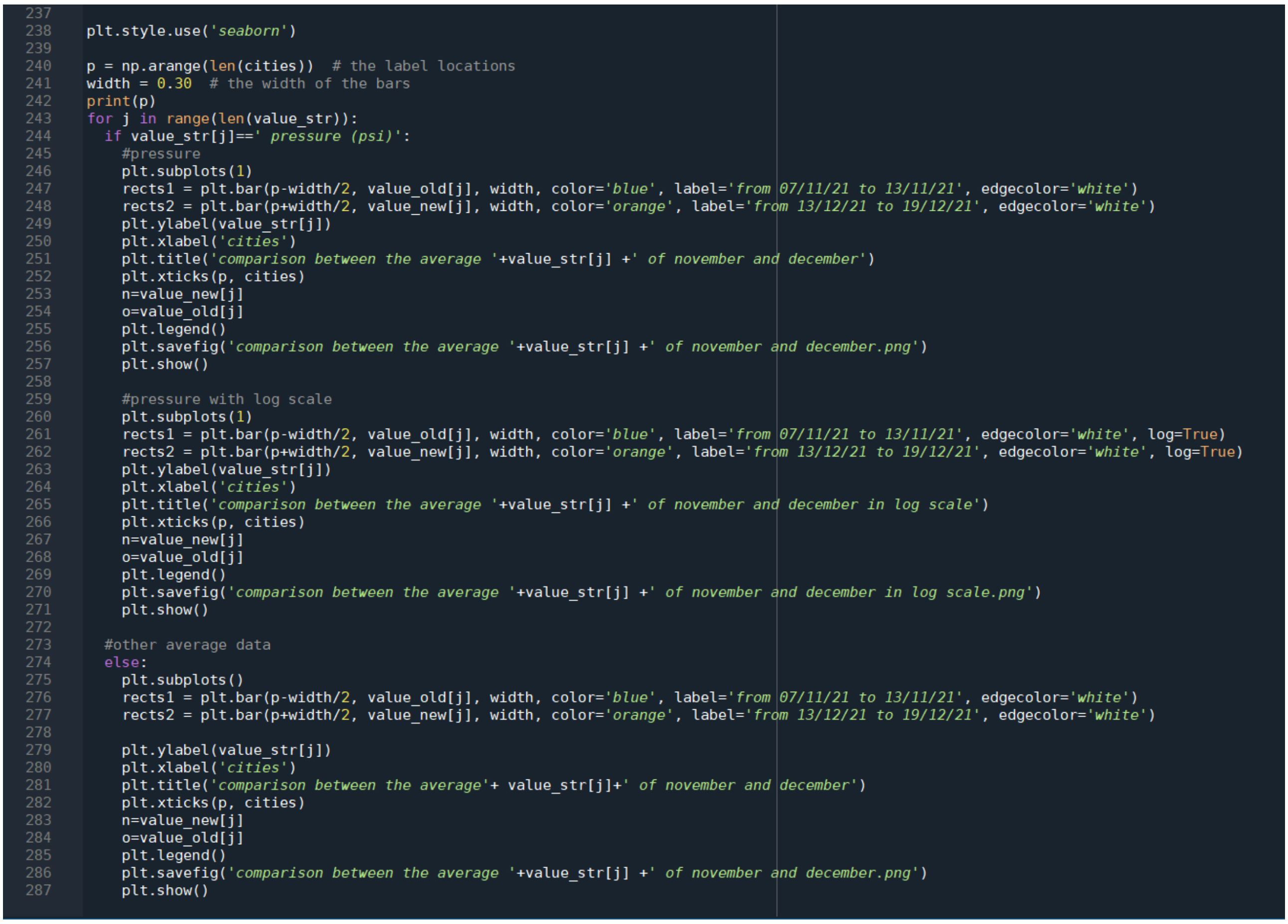

7 to 13/11/21 pressure (psi)
pressure (psi)
1035
1030
1025
1020
1015
1010
2021-11-07
2021-11-08
2021-11-09
2021-11-10
2021-11-11
2021-11-12
2021-11-13
2021-11-14
Time
Berlin
Paris
London
Madrid
Athens
Rome

In the third and last module of the code we have substantially redone what we had done in the two previous ones, only in this case the output will be formed by a series of bar graphs in which we had compared the averages of the values extracted in two different periods in the same city.

This is the code:

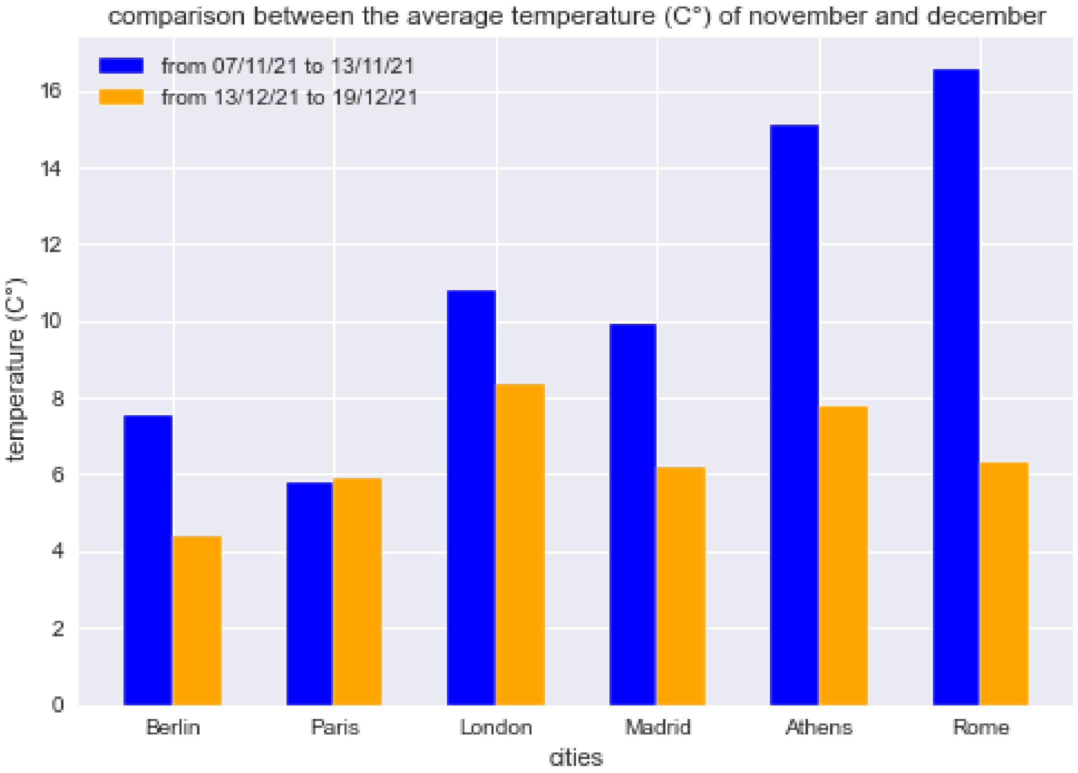


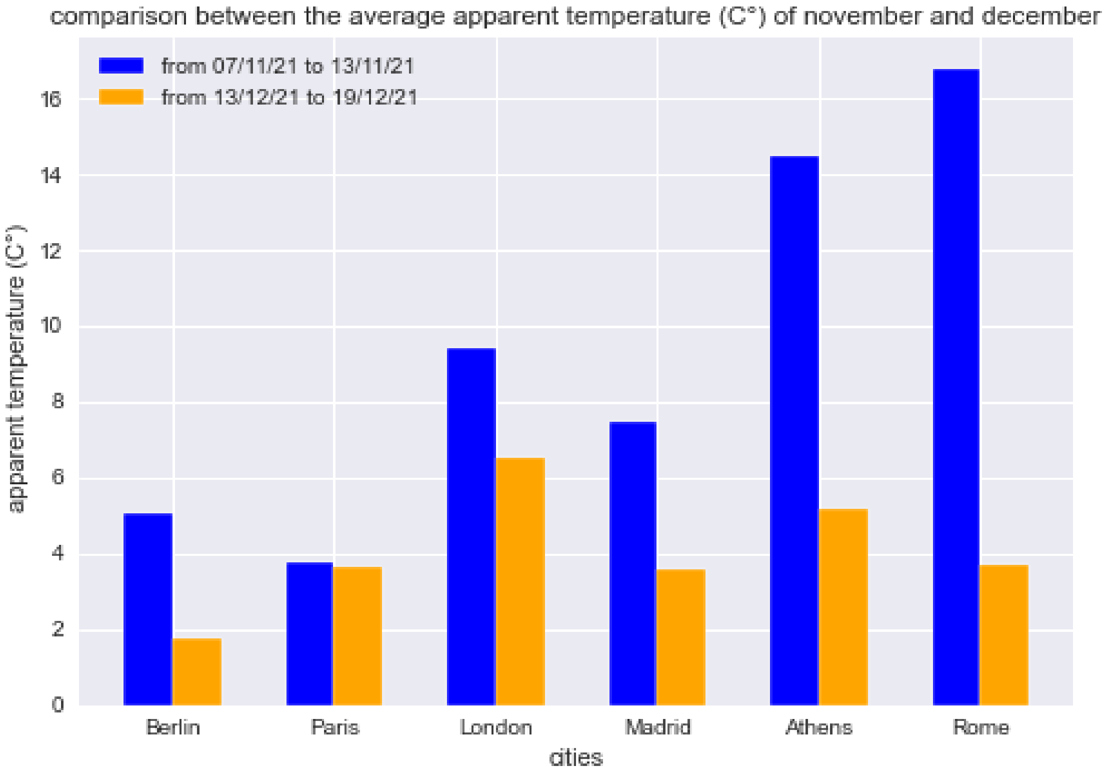
```python
import numpy as np


dfnew = pd.read_csv ("wheaterforecast_eu_capitals_13_12_21dataset.csv")
dfold = pd.read_csv ("wheaterforecast_eu_capitals_7_11_21dataset.csv")


dfold.columns = [' city', ' time', ' temperature (C°)', ' relative humidity (%)', ' apparent temperature (C°)', ' pressure (psi)', ' precipitation (mm)']
dfnew.columns = [' city', ' time', ' temperature (C°)', ' relative humidity (%)', ' apparent temperature (C°)', ' pressure (psi)', ' precipitation (mm)']

#empty list new
temperature_new=[]
relative_humidity_new=[]
apparent_temperature_new=[]
pressure_new=[]
precipitation_new=[]
value_new=[ temperature_new, relative_humidity_new, apparent_temperature_new, pressure_new, precipitation_new]

#empty list old
temperature_old=[]
relative_humidity_old=[]
apparent_temperature_old=[]
pressure_old=[]
precipitation_old=[]
value_old=[ temperature_old, relative_humidity_old, apparent_temperature_old, pressure_old, precipitation_old]

dfold[' time'] = pd.to_datetime(dfold[' time'])
dfnew[' time'] = pd.to_datetime(dfnew[' time'])

for city in cities:
    dfnewa = dfnew[dfnew[' city']==city]
    dfolda = dfold[dfold[' city']==city]
    for i in range(len(value_str)):
       dfnewb=dfnewa.describe()
       dfoldb=dfolda.describe()
       columnnew=dfnewb[value_str[i]]
       columnold=dfoldb[value_str[i]]
       meannew=columnnew['mean']
       meanold=columnold['mean']
       value_new[i].append(meannew)
       value_old[i].append(meanold)


```

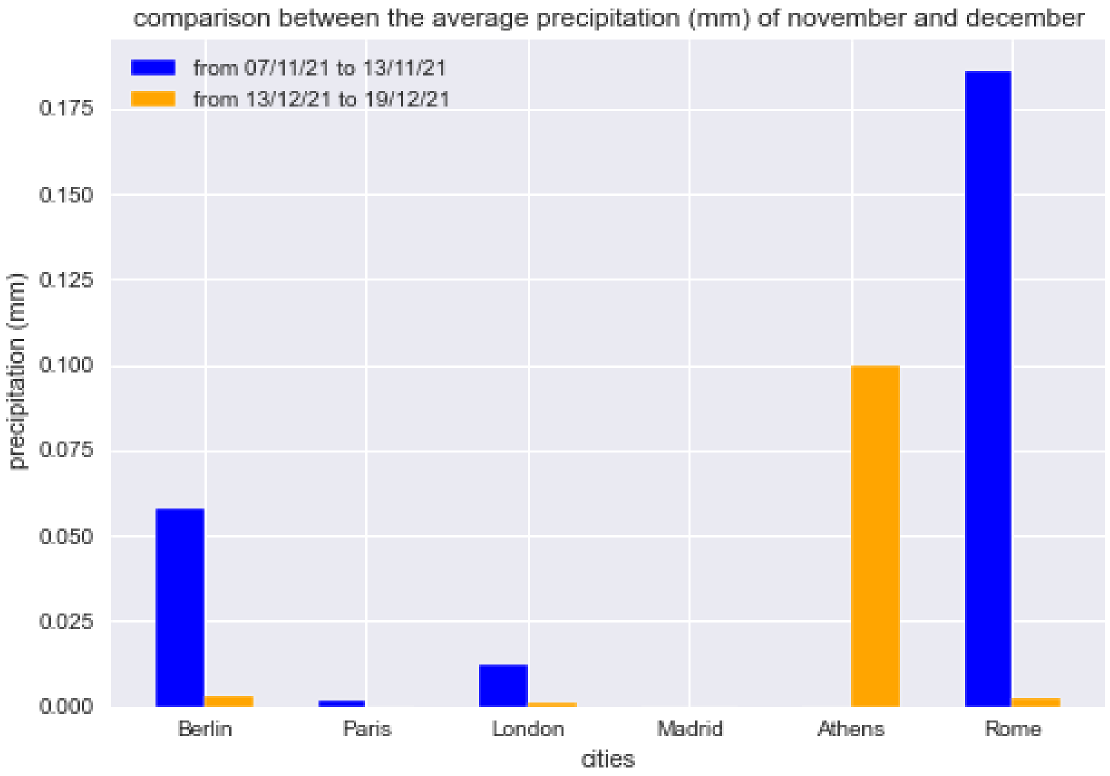


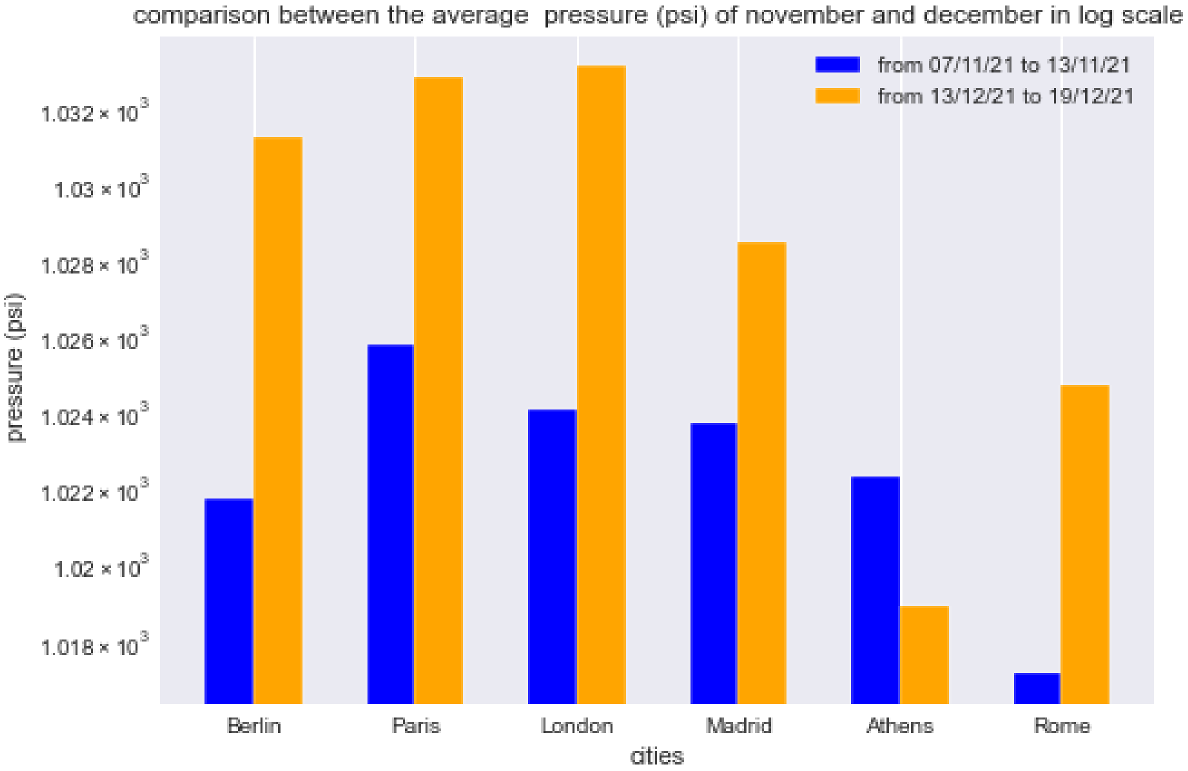
```python

plt.style.use('seaborn')

p = np.arange(len(cities))  # the label locations
width = 0.30  # the width of the bars
print(p)
for j in range(len(value_str)):
  if value_str[j]==' pressure (psi)':
    #pressure
    plt.subplots(1)
    rects1 = plt.bar(p-width/2, value_old[j], width, color='blue', label='from 07/11/21 to 13/11/21', edgecolor='white')
    rects2 = plt.bar(p+width/2, value_new[j], width, color='orange', label='from 13/12/21 to 19/12/21', edgecolor='white')
    plt.ylabel(value_str[j])
    plt.xlabel('cities')
    plt.title('comparison between the average '+value_str[j] +' of november and december')
    plt.xticks(p, cities)
    n=value_new[j]
    o=value_old[j]
    plt.legend()
    plt.savefig('comparison between the average '+value_str[j] +' of november and december.png')
    plt.show()

    #pressure with log scale
    plt.subplots(1)
    rects1 = plt.bar(p-width/2, value_old[j], width, color='blue', label='from 07/11/21 to 13/11/21', edgecolor='white', log=True)
    rects2 = plt.bar(p+width/2, value_new[j], width, color='orange', label='from 13/12/21 to 19/12/21', edgecolor='white', log=True)
    plt.ylabel(value_str[j])
    plt.xlabel('cities')
    plt.title('comparison between the average '+value_str[j] +' of november and december in log scale')
    plt.xticks(p, cities)
    n=value_new[j]
    o=value_old[j]
    plt.legend()
    plt.savefig('comparison between the average '+value_str[j] +' of november and december in log scale.png')
    plt.show()

  #other average data
  else:
    plt.subplots()
    rects1 = plt.bar(p-width/2, value_old[j], width, color='blue', label='from 07/11/21 to 13/11/21', edgecolor='white')
    rects2 = plt.bar(p+width/2, value_new[j], width, color='orange', label='from 13/12/21 to 19/12/21', edgecolor='white')

    plt.ylabel(value_str[j])
    plt.xlabel('cities')
    plt.title('comparison between the average'+ value_str[j]+' of november and december')
    plt.xticks(p, cities)
    n=value_new[j]
    o=value_old[j]
    plt.legend()
    plt.savefig('comparison between the average '+value_str[j] +' of november and december.png')
    plt.show()
```

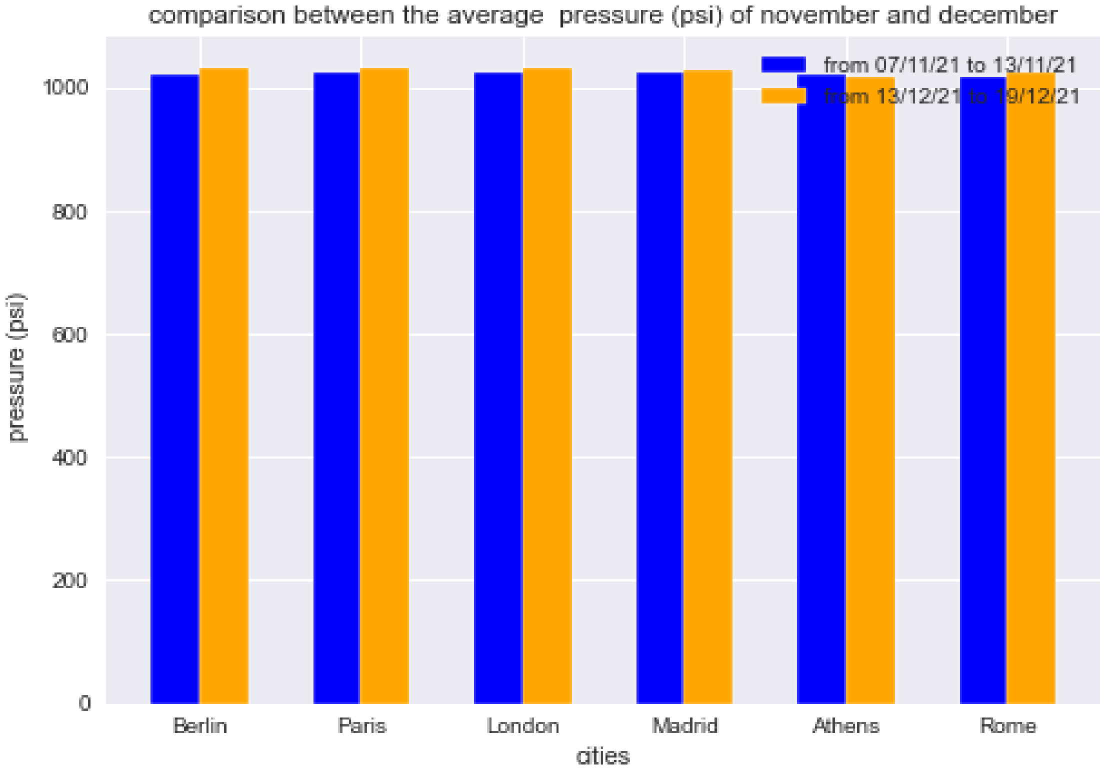
These instead are the resulting bar chart plots:

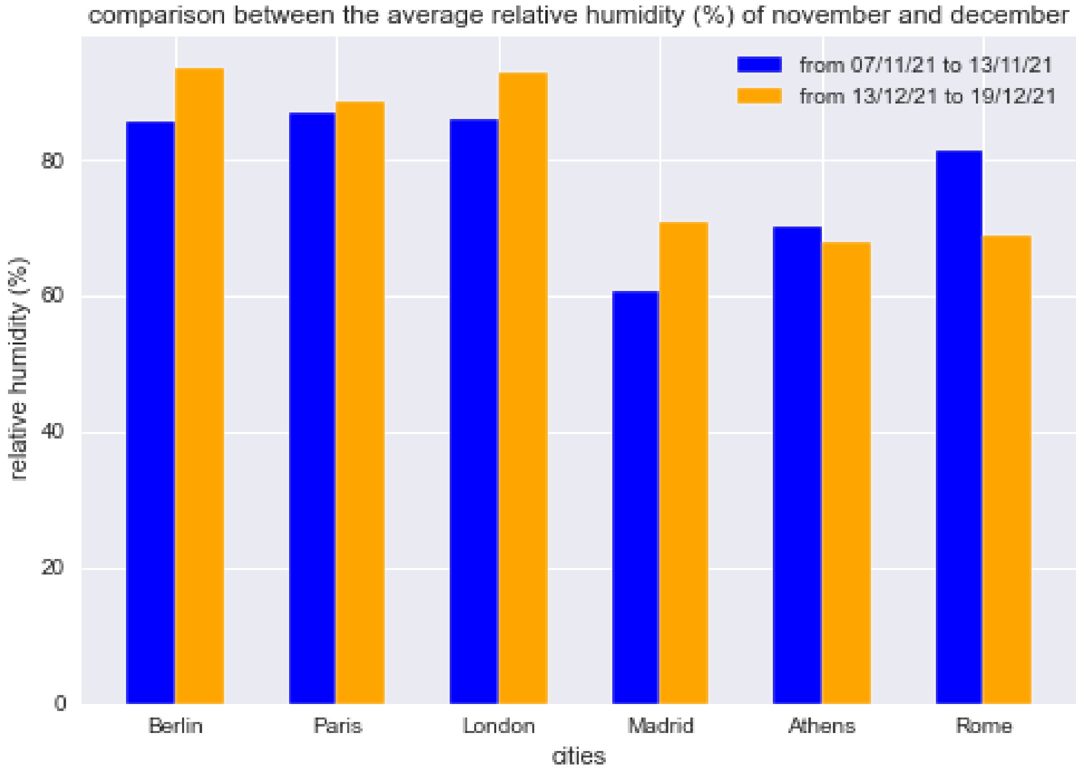





comparison between the average precipitation (mm) of november and december
from 07/11/21 to 13/11/21
from 13/12/21 to 19/12/21
precipitation (mm)
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
Berlin
Paris
London
Madrid
Athens
Rome
cities


comparison between the average pressure (psi) of november and december in log scale
from 07/11/21 to 13/11/21
from 13/12/21 to 19/12/21
pressure (psi)
1.018 × 10^3
1.02 × 10^3
1.022 × 10^3
1.024 × 10^3
1.026 × 10^3
1.028 × 10^3
1.03 × 10^3
1.032 × 10^3
Berlin
Paris
London
Madrid
Athens
Rome
cities


comparison between the average pressure (psi) of november and december
pressure (psi)
0
200
400
600
800
1000
Berlin
Paris
London
Madrid
Athens
Rome
cities
from 07/11/21 to 13/11/21
from 13/12/21 to 19/12/21


comparison between the average relative humidity (%) of november and december
relative humidity (%)
0
20
40
60
80
Berlin
Paris
London
Madrid
Athens
Rome
cities
from 07/11/21 to 13/11/21
from 13/12/21 to 19/12/21