

Föreläsning 6

CSV-filer, matplotlib.

Genomgång av lektion 8 (intro OU4)

- [CSV-filer](#)
- [OU4-uppgift A](#)
- [Matplotlib](#)
- [OU4-uppgift B](#)
- [OU4-uppgift C](#)

Läsa csv-data

Python har modulen `csv` för att läsa och skriva sk csv-data, *comma separated values*. Formatet csv är det vanligaste import- och export-formatet för kalkylblad och databaser. Ex: filen `people.csv`:

Filens innehåll:

```
No, Name, country  
1, Alex, USA  
2, Erik, Sweden  
3, Cheng, China
```

Pythonkod läser csv-filen

=>

och skriver ut innehållet:

```
['No', ' Name', ' country']  
['1', ' Alex', ' USA']  
['2', ' Erik', ' Sweden']  
['3', ' Cheng', ' China']
```

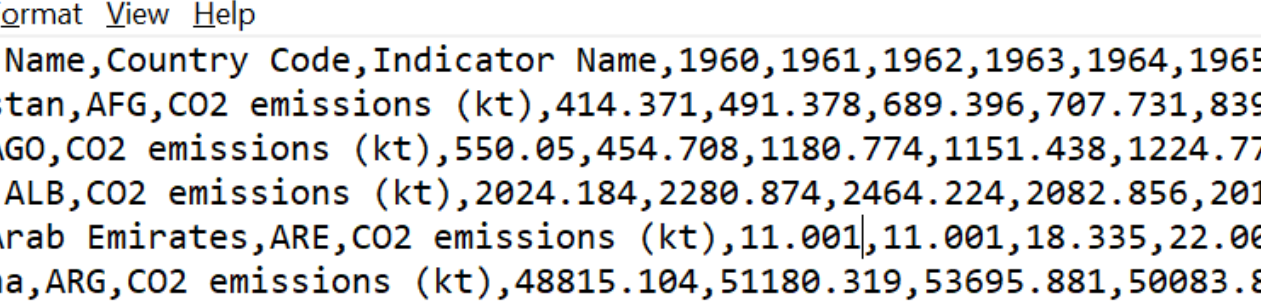
Med följande kod: [readPeoplefile.py](#)

```
import csv  
  
with open('people.csv', 'r') as csvFile:  
    reader = csv.reader(csvFile)  
    for row in reader:  
        print(row)
```

Öppna filen som en csv-fil för att läsa från.
Skapa ett s.k. *reader-objekt* med vilket man kan iterera över varje rad i filen, och returnerar **listor**, en per rad, där elementen är **strängar**. Elementen separeras med komma.

OU4, uppgift A

Givet är en csv-fil **CO2Emissions_filtered.csv** som innehåller data för CO2-utsläppen för 150 st länder under åren 1960 – 2014. Filen finns att ladda ned från kurssidan under lektion 8. Filen går att öppna med exvis Notepad och den ser ut så här i början:



CO2Emissions_filtered.csv - Notepad

File Edit Format View Help

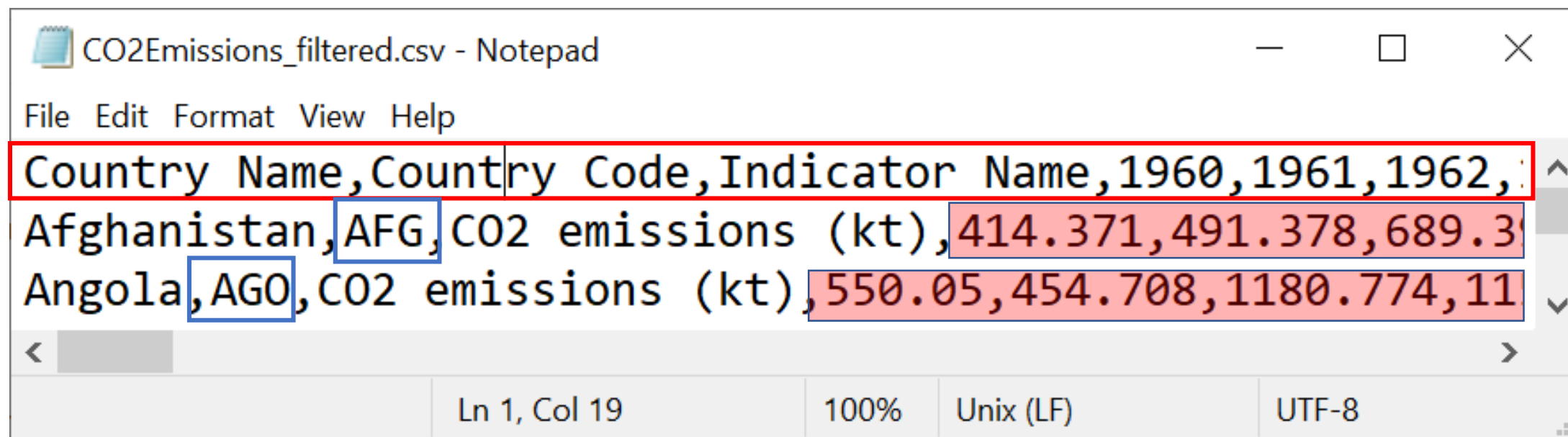
Country Name	Country Code	Indicator Name	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016																																																																																																																											
Afghanistan	AFG	CO2 emissions (kt)	414.371	491.378	689.396	707.731	839.743	1014.371	1114.371	1214.371	1314.371	1414.371	1514.371	1614.371	1714.371	1814.371	1914.371	2014.371	2114.371	2214.371	2314.371	2414.371	2514.371	2614.371	2714.371	2814.371	2914.371	3014.371	3114.371	3214.371	3314.371	3414.371	3514.371	3614.371	3714.371	3814.371	3914.371	4014.371	4114.371	4214.371	4314.371	4414.371	4514.371	4614.371	4714.371	4814.371	4914.371	5014.371	5114.371	5214.371	5314.371	5414.371	5514.371	5614.371	5714.371	5814.371	5914.371	6014.371																																																																																																																												
Angola	AGO	CO2 emissions (kt)	550.05	454.708	1180.774	1151.438	1224.778	1188.774	1151.438	1114.102	1076.766	1039.430	1002.094	964.758	927.422	890.086	852.750	815.414	778.078	740.742	703.406	666.070	628.734	591.398	554.062	516.726	479.390	442.054	404.718	367.382	330.046	292.710	255.374	218.038	180.702	143.366	106.030	68.694	31.358	-5.978	-43.642	-81.306	-118.970	-156.634	-194.298	-231.962	-269.626	-307.290	-344.954	-382.618	-420.282	-457.946	-495.610	-533.274	-570.938	-608.602	-646.266	-683.930	-721.594	-759.258	-796.922	-834.586	-872.250	-909.914	-947.578	-985.242	-1022.906	-1060.570	-1098.234	-1135.898	-1173.562	-1211.226	-1248.890	-1286.554	-1324.218	-1361.882	-1399.546	-1437.210	-1474.874	-1512.538	-1550.202	-1587.866	-1625.530	-1663.194	-1700.858	-1738.522	-1776.186	-1813.850	-1851.514	-1889.178	-1926.842	-1964.506	-2002.170	-2039.834	-2077.498	-2115.162	-2152.826	-2190.490	-2228.154	-2265.818	-2303.482	-2341.146	-2378.810	-2416.474	-2454.138	-2491.802	-2529.466	-2567.130	-2604.794	-2642.458	-2680.122	-2717.786	-2755.450	-2793.114	-2830.778	-2868.442	-2906.106	-2943.770	-2981.434	-3019.098	-3056.762	-3094.426	-3132.090	-3169.754	-3207.418	-3245.082	-3282.746	-3320.410	-3358.074	-3395.738	-3433.402	-3471.066	-3508.730	-3546.394	-3584.058	-3621.722	-3659.386	-3697.050	-3734.714	-3772.378	-3810.042	-3847.706	-3885.370	-3923.034	-3960.698	-3998.362	-4036.026	-4073.690	-4111.354	-4149.018	-4186.682	-4224.346	-4262.010	-4299.674	-4337.338	-4374.902	-4412.566	-4450.230	-4487.894	-4525.558	-4563.222	-4600.886	-4638.550	-4676.214	-4713.878	-4751.542	-4789.206	-4826.870	-4864.534	-4902.198	-4939.862	-4977.526	-5015.190	-5052.854	-5090.518	-5128.182	-5165.846	-5203.510	-5241.174	-5278.838	-5316.502	-

Vi noterar att första raden i filen är en **rubrik**.

Varje ord i rubriken har motsvarande värde på de andra raderna:

Country Code → AFG → AGO → ...

Varje land har en **landskod**, alla **CO2-värden** för ett land är på en rad.

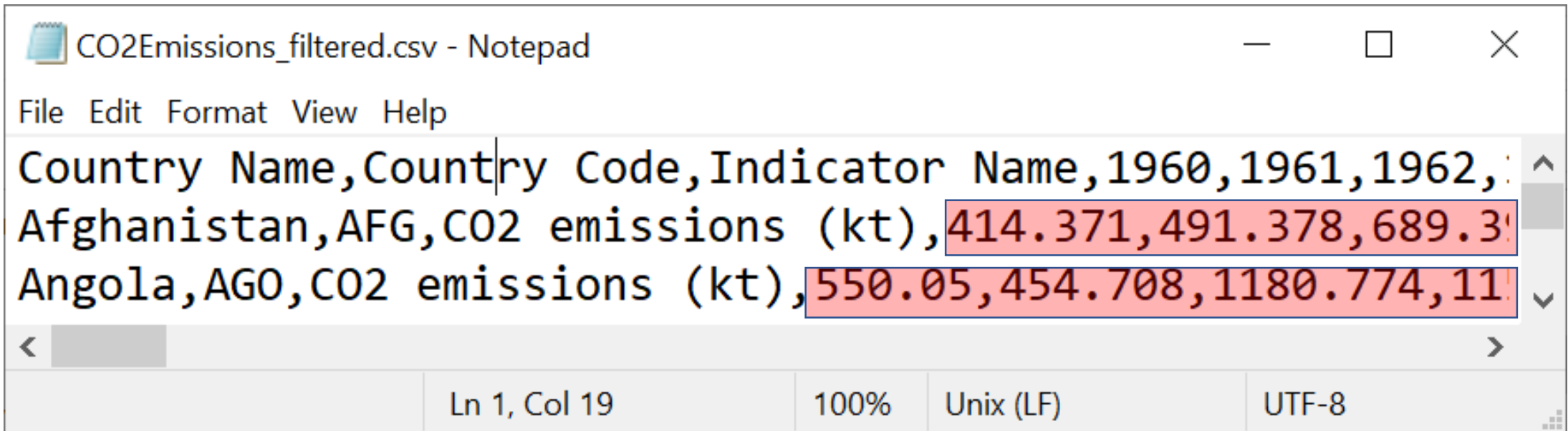


```
CO2Emissions_filtered.csv - Notepad
File Edit Format View Help
Country Name,Country Code,Indicator Name,1960,1961,1962,...
Afghanistan,AFG,CO2 emissions (kt),414.371,491.378,689.3
Angola,AGO,CO2 emissions (kt),550.05,454.708,1180.774,11
Ln 1, Col 19 100% Unix (LF) UTF-8
```

Uppgift A

Skriv en funktion `load_csv(filename)` som skapar och returnerar ett lexikon. Nycklarna är ländskoder och värdena är listor med CO2-utsläppen (flyttal) för åren 1960 – 2014:

`{AFG: [414.371, 491.378, .....], AGO: [550.05, 454.708, ...], ALB: [...]}`



```
Country Name,Country Code,Indicator Name,1960,1961,1962,
Afghanistan,AFG,CO2 emissions (kt),414.371,491.378,689.3
Angola,AGO,CO2 emissions (kt),550.05,454.708,1180.774,11
```

Ett skal (load_csv_skal.txt) till funktionen load_csv

Det fattas detaljer som du skall skriva...

```
import csv
```

```
def load_csv(filename): # 'CO2Emissions_filtered.csv'
    lex = {}
    with open(filename, 'r') as csvFile: # Öppna filen för läsning
        reader = csv.reader(csvFile) # Skapa ett s.k. reader-objekt
        for row in reader: # row blir en lista med strängar.

            country_code = ? # Det andra elementet på raden. Alltid?

            data = ? # Gör en lista med alla CO2-data på raden.
                    # Start-kolumn? Alltid? Tips: skrivning

            # Lägg in det nya key-value-paret i lex
            ...      # ?

    return lex
```

Alternativ:

```
import csv

def load_csv(filename): # 'CO2Emissions_filtered.csv'
    lex = {}
    # Alternativ:
    with open(filename, 'r') as csvFile:
        # Skapa en lista där varje element är en lista med orden
        #      rows = list(csv.reader(csvFile))
        ...

    return lex
```

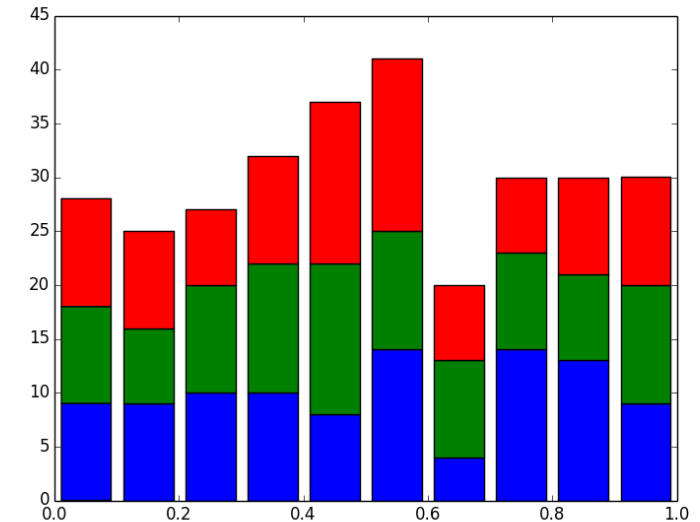
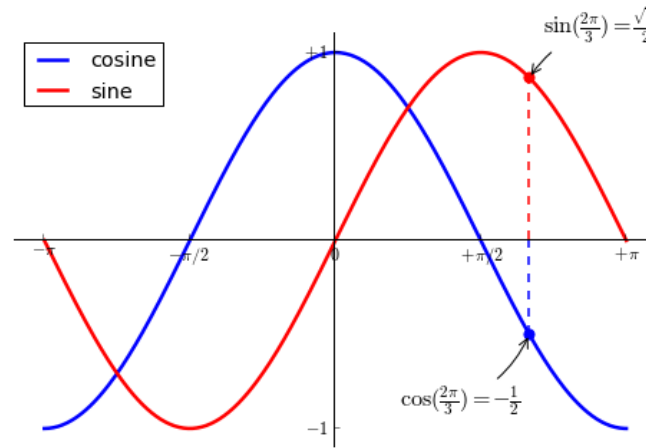
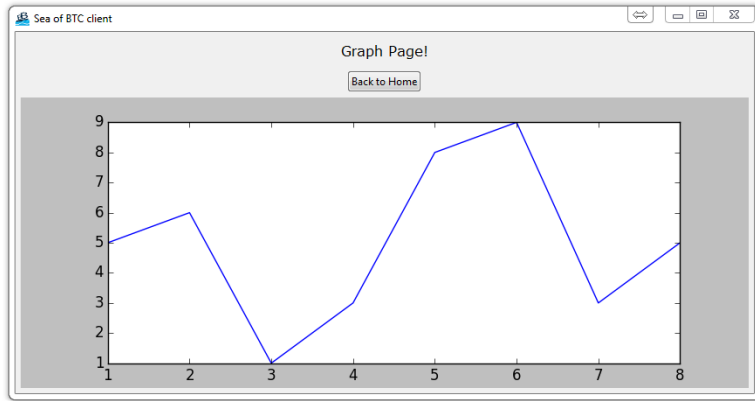
Att fundera på:

- Titta på filen `CO2Emissions_filtered.csv`. Finns det “skräp” i datat? (t ex rad 15, 29, 30). Hur kan man hantera att raderna ser olika ut? (Ni får inte ändra i csv-filen)
- Vilken typ har datat?
- Fattas det något steg i algoritmen? Kan läsa enstaka rad med satsen `next(reader)`, dvs iterera ett steg utan att ta hand om resultatet
- `reader`-objektet kan man iterera genom i en loop eller göra om direkt till en lista
- Hur ska man felsöka om programmet krashar?
Tips: Skriv ej ut hela dictionary'n. Den innehåller 150 länder.
Skriv istället ut delar av den om du önskar felsöka...



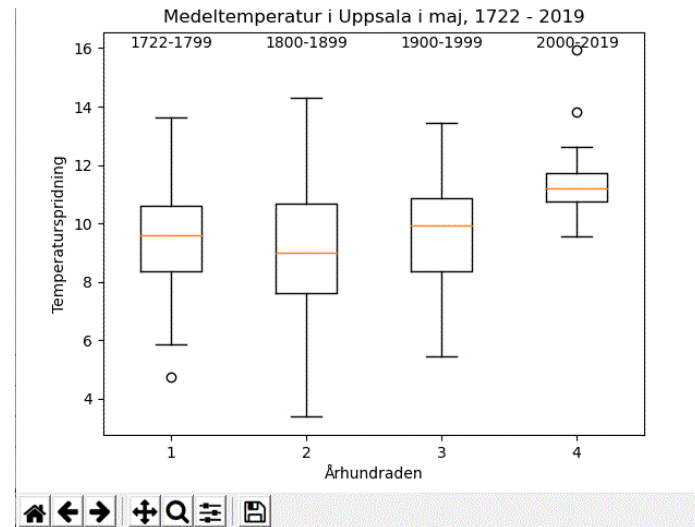
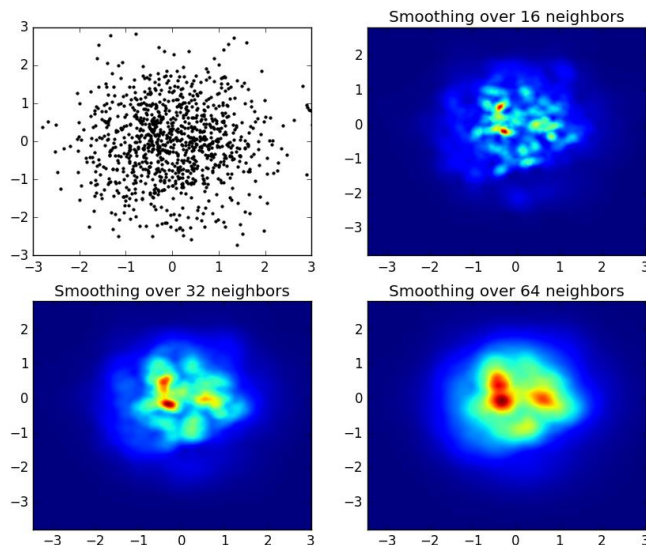
Datavisualisering med Python: matplotlib

“Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.” <https://matplotlib.org/>



[Det här](#)

[Det här f](#)



CSV-filer, matplotlib, Intro ou4 (Torsten Andersson, IT-inst,
Uppsala Universitet)

Tips på en tutorial, gör den själva.

<https://techvidvan.com/tutorials/matplotlib-tutorial/>: Python for Data Visualization.

This tutorial will help you learn about plots, kinds of plot, subplots, labels, and much more.

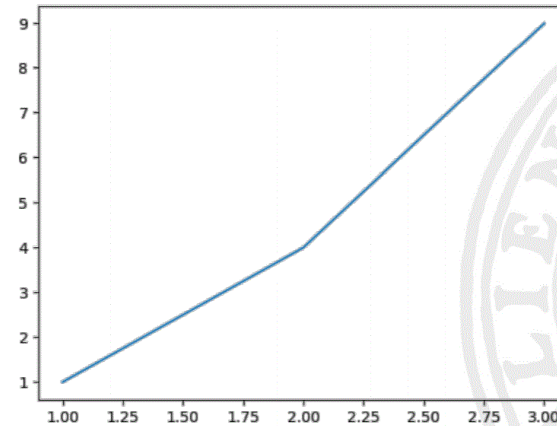
“Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.”

Matplotlib is the most extensively used Data Visualization library in Python programming. You can generate plots, graphs, bar graphs, scatter graphs, histograms, arrays etc. in a 2D form easily.

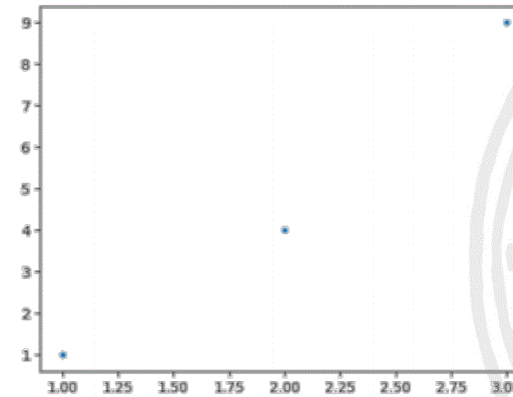
Några exempel

```
import matplotlib.pyplot as plt
# plt blir alias för matplotlib.pyplot
# pyplot är en modul i paketet matplotlib
# matplotlib.pyplot.plot([1,2,3],[1,4,9],"-")  Onödigt långt!
```

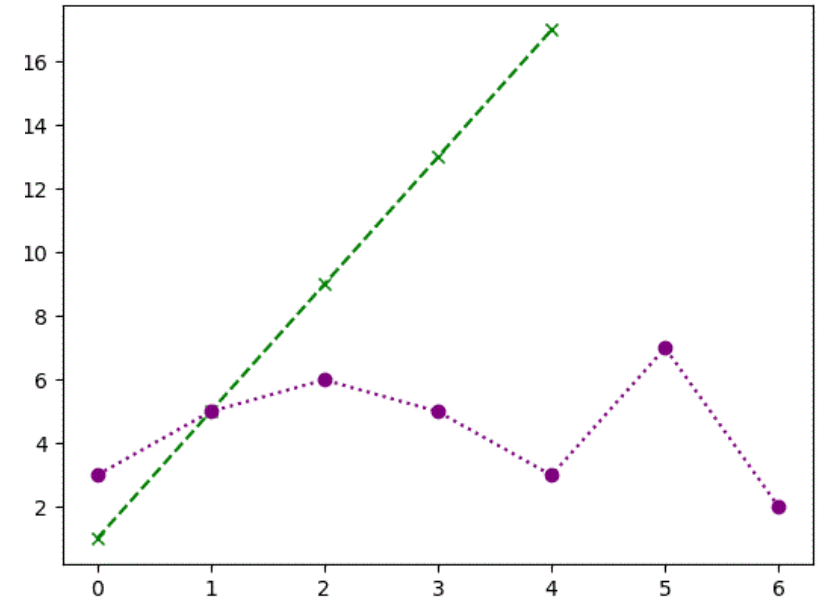
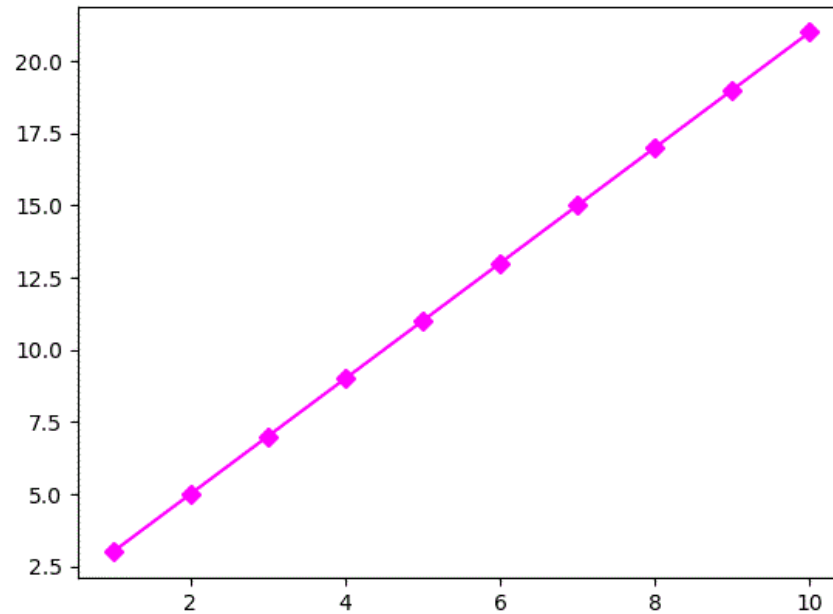
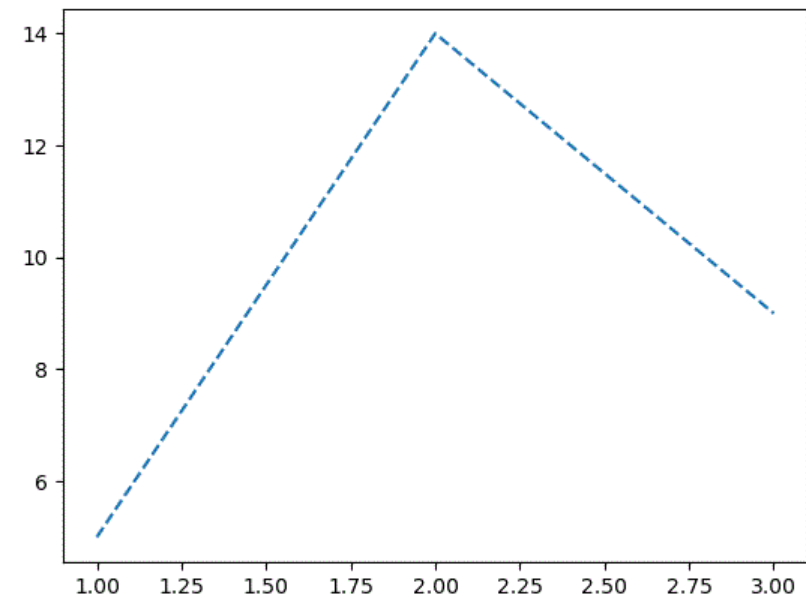
```
plt.plot([1,2,3],[1,4,9],"-")
# linje mellan (1,1), (2,4), (3,9)
plt.show()
```



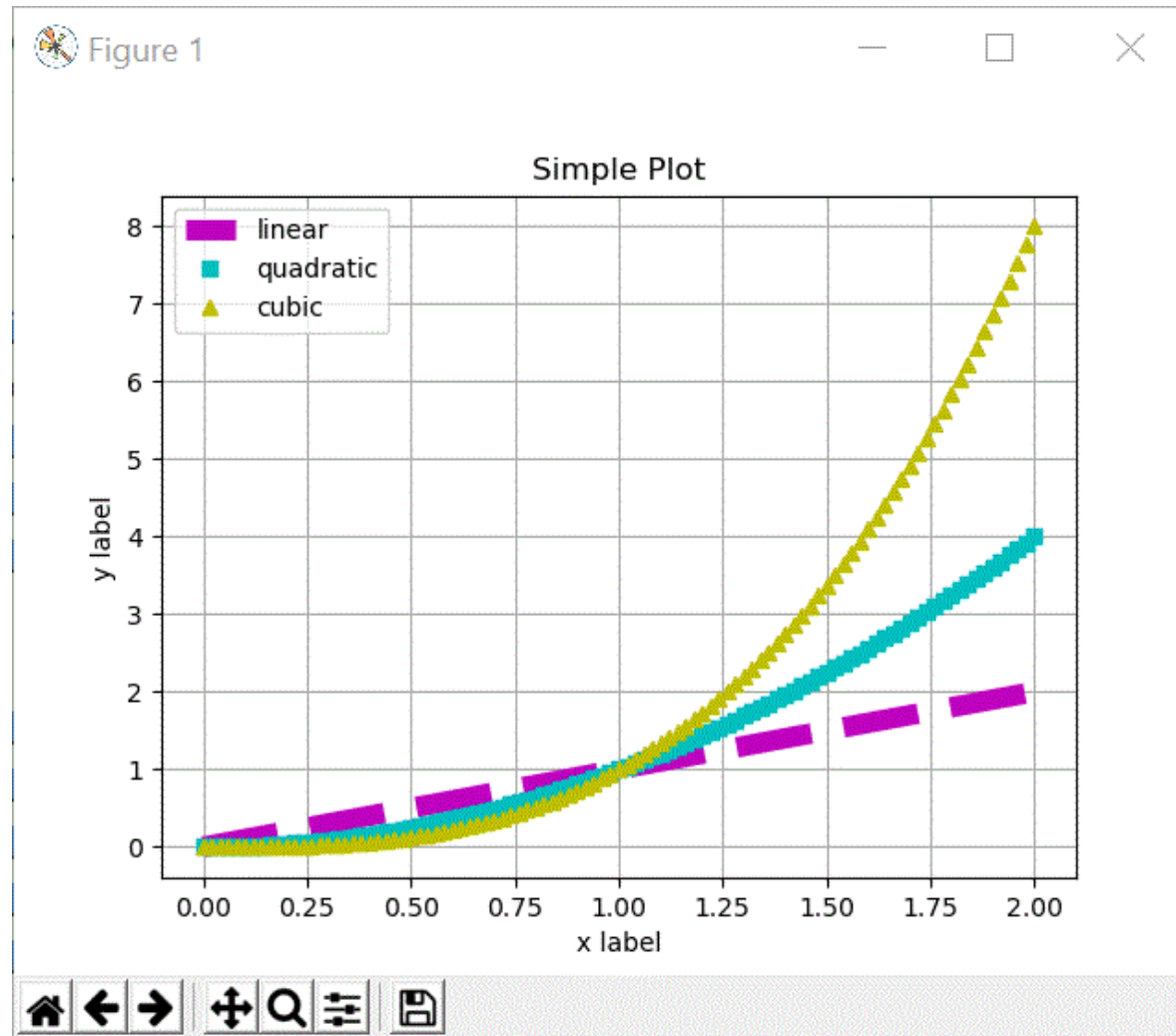
```
plt.plot([1,2,3],[1,4,9], ".")
# ritar punkterna
plt.show()
```



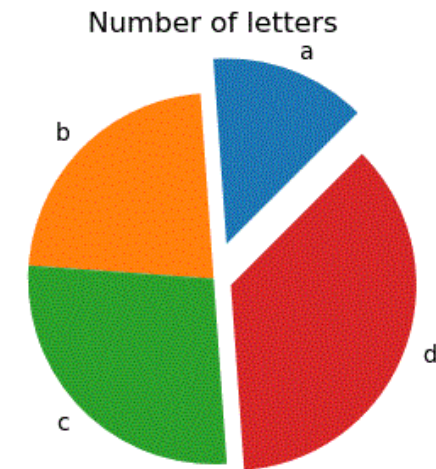
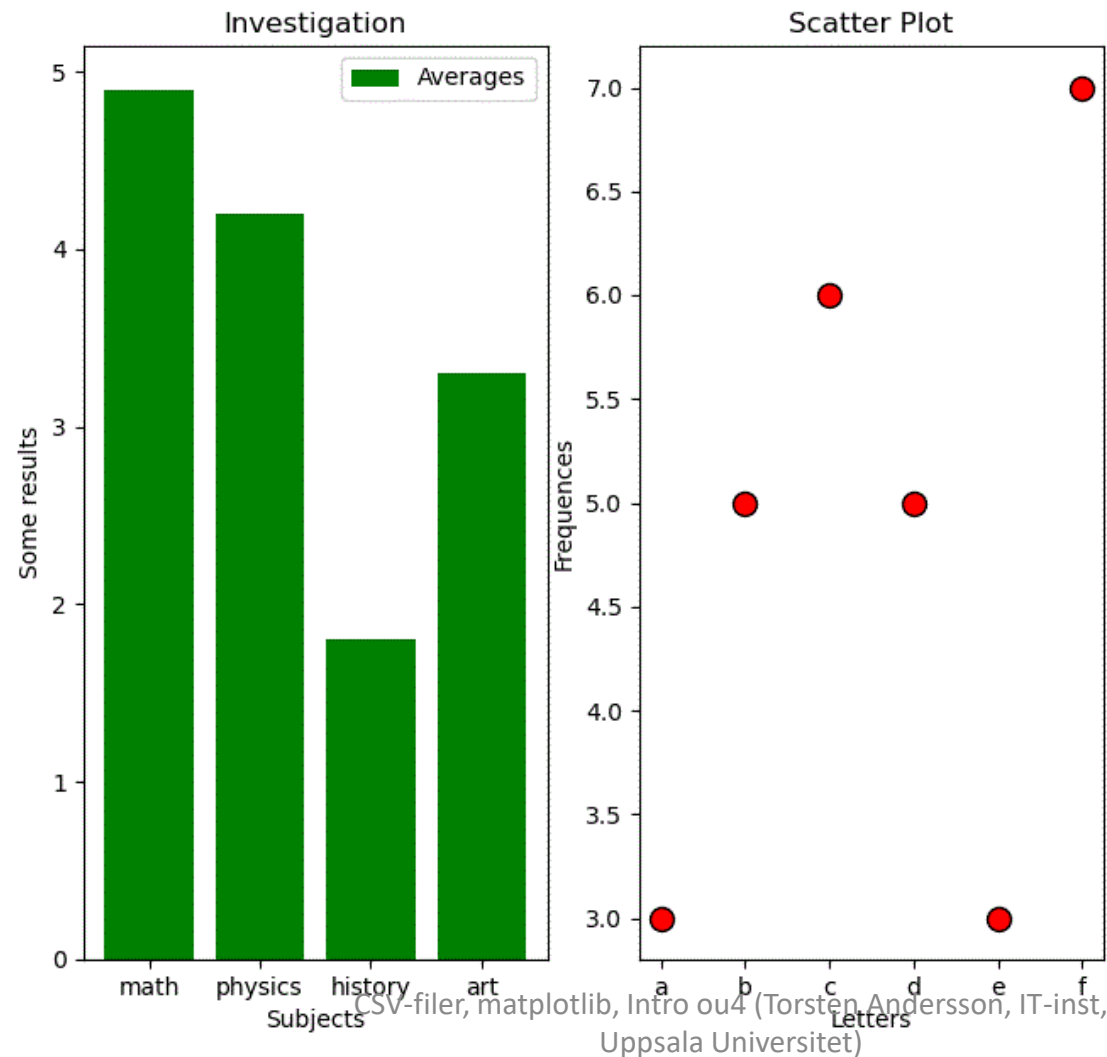
Ex1.py som skapar följande.



[Ex2.py](#) som skapar följande.



Ex3.py Visar hur man kan plotta flera diagram i samma fönster



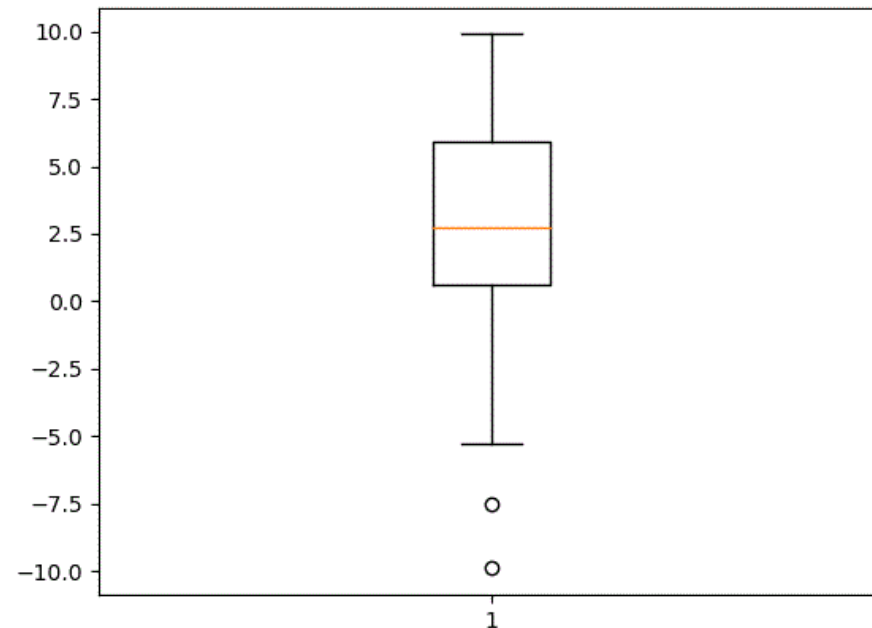
Boxplot (lådadiagram)

```
measures = [-1.2, 1.2, -5.3, -9.9, 0.7, -7.5, 0.4, 2.8, \
2.7, 4.8, 1.3, 1.1, 6.6, 4.2, 5.8, 6.5, 2.4, 3.4, 5.9, \
2.4, -0.9, -1.5, 4.1, 8.3, 7.9, 9.2, 9.9, 6.0]
```

```
plt.boxplot(measures)
```

```
plt.show()
```

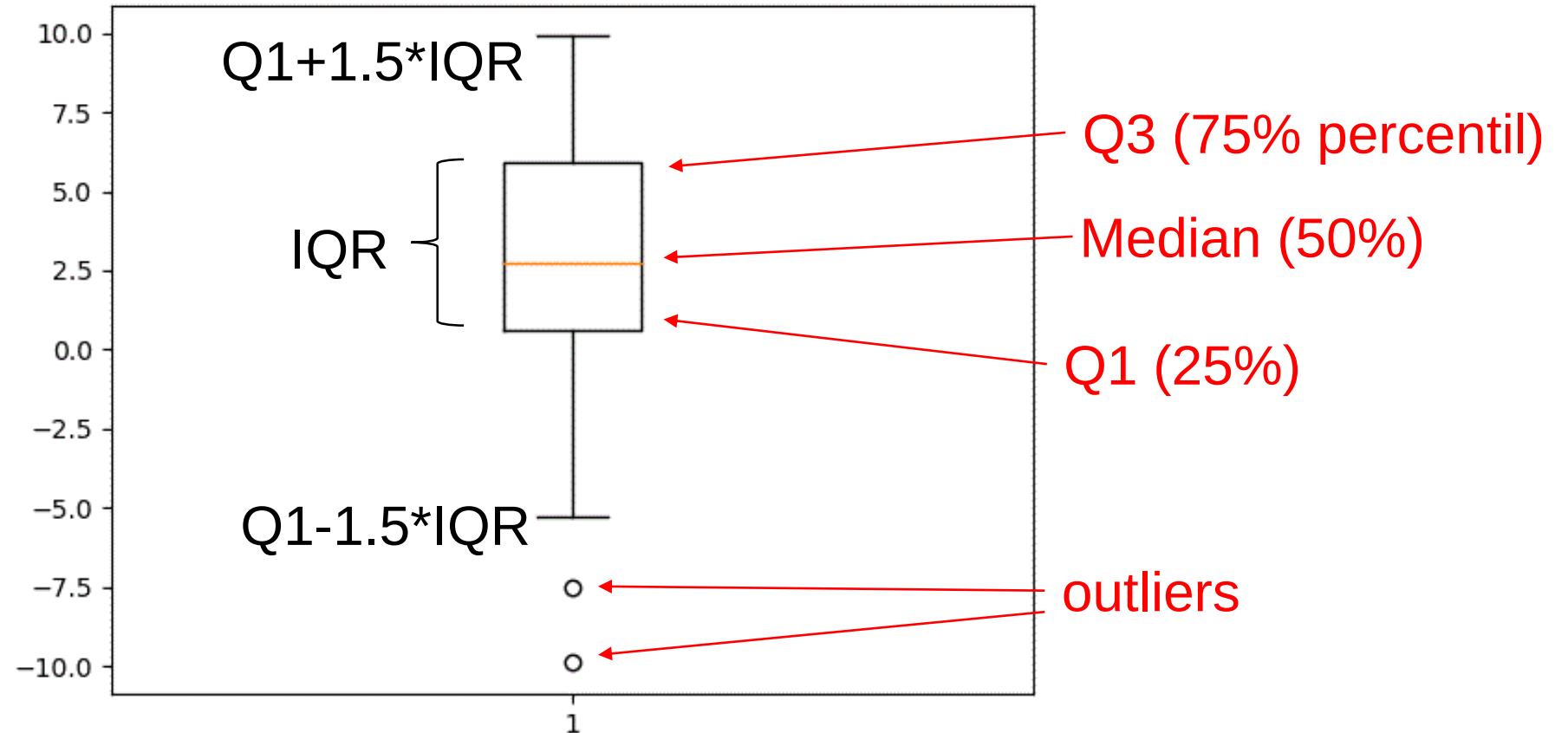
Figure 1



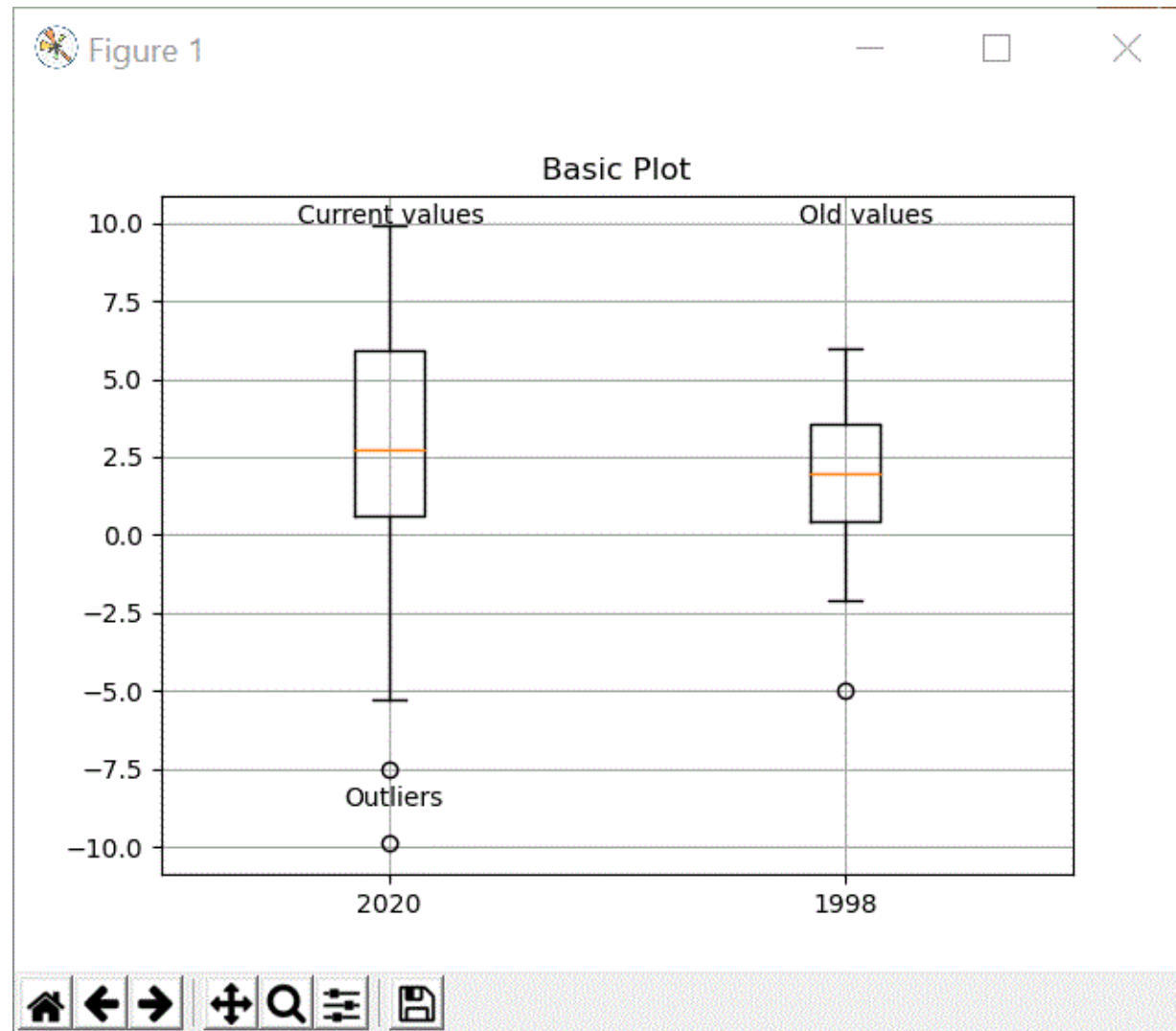
Förklaring ges på nästa sida →

```
measures = [-1.2, 1.2, -5.3, -9.9, 0.7, -7.5, 0.4, 2.8, \
2.7, 4.8, 1.3, 1.1, 6.6, 4.2, 5.8, 6.5, 2.4, 3.4, 5.9, \
2.4, -0.9, -1.5, 4.1, 8.3, 7.9, 9.2, 9.9, 6.0]
```

IQR = Q3-Q1
Interquartile
Range



Ex4.py (två boxplot i samma figur)



Några viktiga funktioner:

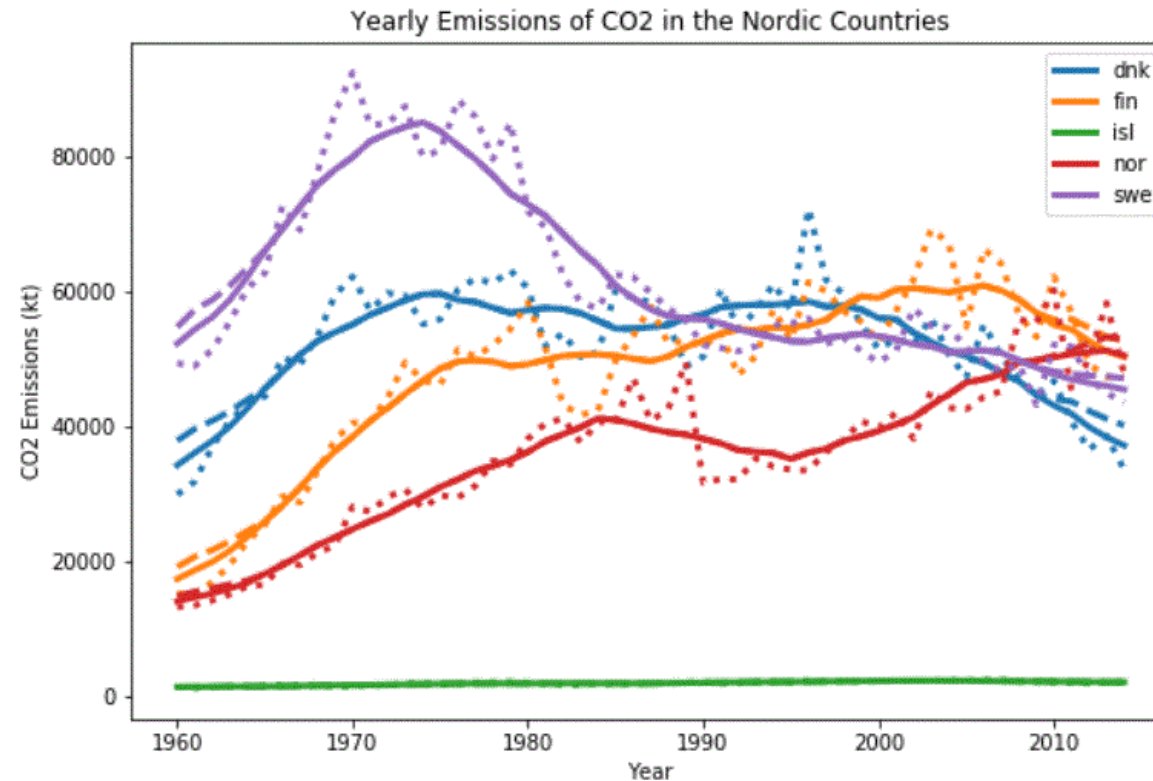
- `plot()` : Används för att plotta punkter eller 'linje'
- `label` : Används för att ge namn till graferna
- `xlabel()` / `ylabel()` : Används för att namnge axlarna (x och y) i grafen.
- `title()` : Sätter rubrik på grafen.
- `annotate()` : Lägg in text på angivna koordinater
- `legend()` : Används för att de label som satts ska visas.
- `grid()` : Ger (rut)nät
- `show()` : Används för att visa upp hela grafen.

Exempel på olika diagram:

- `plt.plot(...)` linjediagram
- `plt.bar(...)` stapeldiagram
- `plt.scatter(...)` punktdiagram/spridningsdiagram
- `plt.pie(...)` cirkeldiagram
- `plt.hist(...)` histogram
- `plt.boxplot()` lådagram: median, 1:a och 3:a kvartilen, mm.

OU4, uppgift B

1. Använd funktionen `load_csv` från uppgift A och rita med plot-funktionen `CO2-utsläppen` för Danmark, Finland, Island, Norge och Sverige för åren 1960 till och med 2014.
2. Minska bruset genom att (åter)använda funktionerna `smooth_a` och `smooth_b` från Lektion 6.



OU4, uppgift B

FÖRENKLAD ALGORITM, förslag (det fattas detaljer):

```
from filnamnet import load_csv # importera filen med fkn
```

Gör lista med landskoderna 'dnk', 'fin', 'isl', 'nor', 'swe'

Anropa **load_csv** → ger ett lexikon:

```
{AFG: [414.371, 491.378, .....], AGO: [550.05, 454.708, ...], ALB: [...]}
```

Loopa igenom lexikonet:

är det landskoden för ett nordiskt land? spara i så fall data i ett annat lexikon

```
time = list(range(1960, 2015))
```

```
fig, ax = plt.subplots() # skapar figurens ram.
```

Loopa igenom nya lexikonet. För varje värdepar:

```
ax.plot(...) # Vad ska vara på x-axeln och vad på y-axeln?  
            # Se uppgiften
```

```
ax.plot(...) # använd smooth_a
```

...

OU4, uppgift C

Rita upp lådagram (boxplot) för maj månads medeltemperaturer för Uppsala uppdelat på 1700-, 1800-, 1900- och 2000-talen. Hämta data från textfilen **uppsala_tm_1722-2019.dat** från kurswebben, lektion 8. (**OBS! Inte csv-fil**).

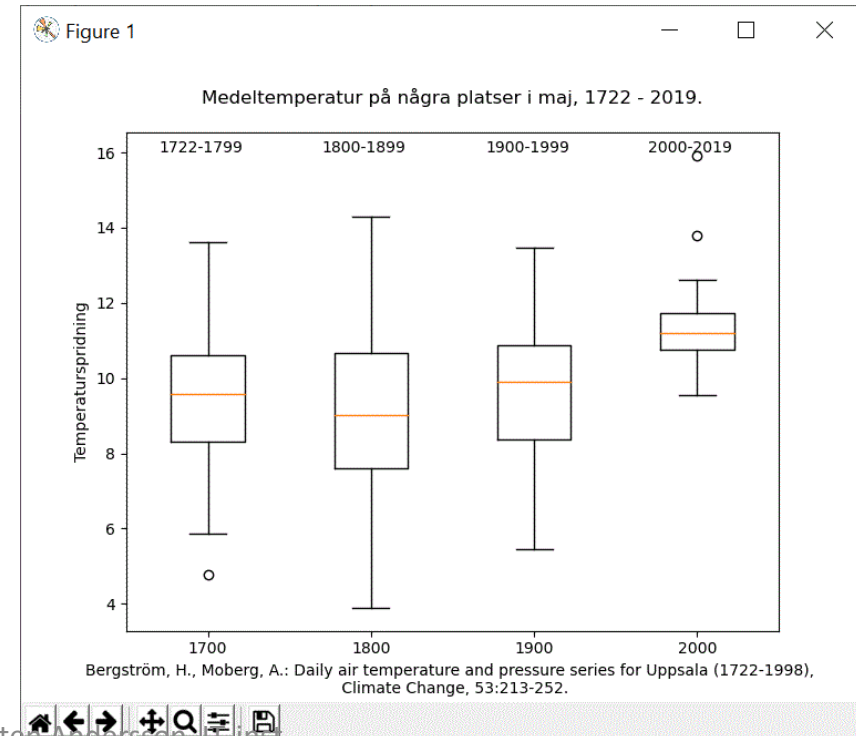
Förslag: Beräkna, spara i listor, dvs

[medeltemp maj 1722, medeltemp maj 1723,,medeltemp maj 1799]

```
list_average_per_year_1700=[9.8, 8.7, ... , 10.1] # för hela 1700-talet  
list_century.append(list_average_per_year_1700)
```

Filen **uppsala_tm_1722-2019.dat**:

1722	4	27	4.8	4.4	1
1722	4	28	4.8	4.4	1
1722	4	29	5.7	5.3	1
1722	4	30	5.7	5.3	1
1722	5	1	4.6	4.2	1
1722	5	2	3.9	3.5	1
1722	5	3	5.0	4.6	1
1722	5	4	5.6	5.2	1
1722	5	5	6.1	5.7	1
1722	5	6	9.0	8.6	1
1722	5	7	9.8	9.5	1
1722	5	8	11.7	11.4	1
1722	5	9	12.4	12.1	1
1722	5	10	12.3	12.0	1



OU4, uppgift C

1. Öppna filen `uppsala_tm_1722-2019.dat` för läsning (inte en csv-fil)
2. För varje rad i filen:
 - ta bort mellanslag - ta bara med decimaltal. Använd reguljära uttryck eller `split()`-funktionen.
 - lägg raden i en lista
3. För varje rad i listan med all data:
 - summera temperaturerna i kolumn 4 för maj månad
4. Beräkna medeltemperaturen.
5. Spara i en lista
6. Rita upp

Det fattas saker i algoritmen....



FÖRENKLAD 'ALGORITM', ett förslag (det fattas detaljer):

```
import matplotlib.pyplot as plt, (re)

# Läs filens data och lagra i en lämplig variabel
all_text = [] #Filens innehåll ska sparas i en lista
with open('uppsala_tm_1722-2019.dat', 'r') as infil:
    # Gör varje rad till en lista utan extra mellanslag:
    for line in infil:
        # ta ut alla decimaltal) # eller .split()
        line_text = ?
        ...      # Lägg in line_text i all_text
```

Forts nästa sida→


```
this_year = int(all_text[0][0]) # börjar år 1722
year_average_temp=[]
# Spara medeltemp per år för maj under ETT århundrade
century_average_temp = []
# Spara en lista per århundrade - blir 4 listor
#loopa över all_text:
    #om det är maj this_year:
        sum_temp += float(row[3]) #summera temperaturen i maj det året
# Nytt år?
# Lägg in (sum_temp/31) i year_average_temp,
# dvs medeltemp 1722, 1723, ... 1799
# Nytt århundrade?
# Spara listan i fyra århundradelistan century_average_temp

# Glöm inte 2000-talet!

# Rita upp de 4 listorna som ligger i århundradelistan som lådagram
```

Forts nästa sida→

Glöm inte att erkänna upphovsmakarna, se text i filen
uppsala_tm_1722-2019 genom att lägga en lämplig text i plotten.

