

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']
```

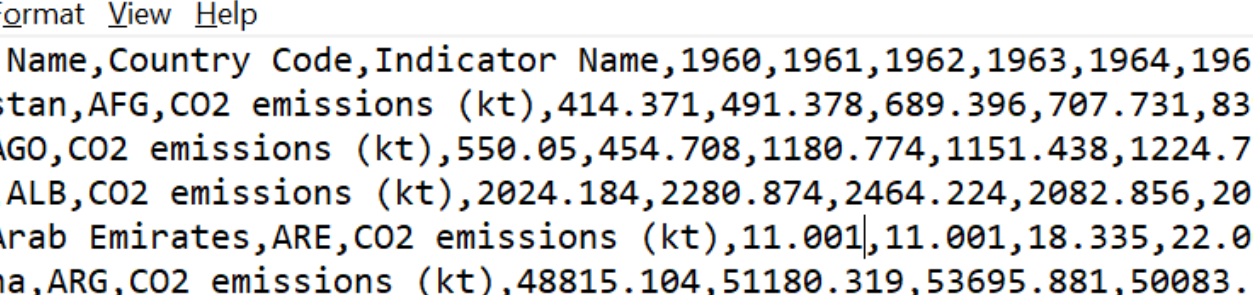
Med följande kod: [readPeoplefile.py](#)

```
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 returnera **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 med programmet 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
Afghanistan	AFG	CO2 emissions (kt)	414.371	491.378	689.396	707.731	839.743	10	
Angola	AGO	CO2 emissions (kt)	550.05	454.708	1180.774	1151.438	1224.778	1188.	
Albania	ALB	CO2 emissions (kt)	2024.184	2280.874	2464.224	2082.856	2016.85	21	
United Arab Emirates	ARE	CO2 emissions (kt)	11.001	11.001	18.335	22.002	18.33		
Argentina	ARG	CO2 emissions (kt)	48815.104	51180.319	53695.881	50083.886	5572		
Antigua and Barbuda	ATG	CO2 emissions (kt)	36.67	47.671	102.676	84.341	91.675		
Australia	AUS	CO2 emissions (kt)	88202.351	90589.568	94912.961	101029.517	108		
Austria	AUT	CO2 emissions (kt)	30821.135	31862.563	33905.082	36992.696	38943.		
Belgium	BEL	CO2 emissions (kt)	91000.272	92793.435	98117.919	105781.949	10366		
Benin	BEN	CO2 emissions (kt)	161.348	128.345	135.679	121.011	143.013	150.347	
Burkina Faso	BFA	CO2 emissions (kt)	44.004	91.675	84.341	88.008	110.01	102.67	
Bulgaria	BGR	CO2 emissions (kt)	22295.36	25973.361	30736.794	34411.128	42863.		
Bahrain	BHR	CO2 emissions (kt)	575.719	1771.161	1591.478	1195.442	1598.812	12	
Bahamas, The	BHS	CO2 emissions (kt)	410.704	546.383	726.066	707.731	1085.45		

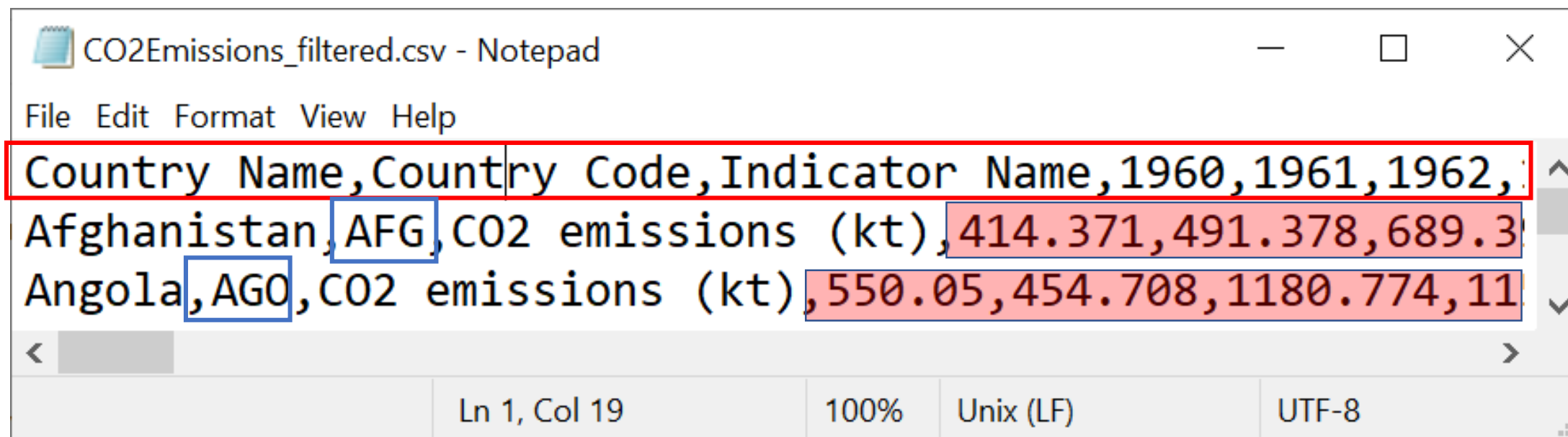
Ln 5, Col 51 100% Unix (LF) UTF-8

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

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

Exempelvis: Country Code → AFG → AGO → ...

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

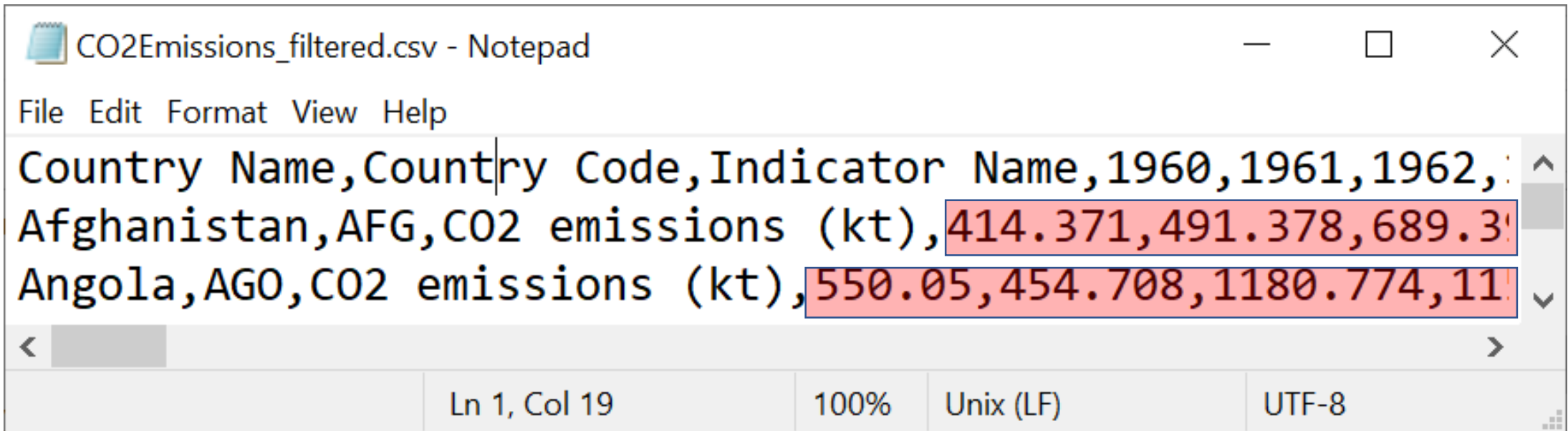


```
Country Name, Country Code, Indicator Name, 1960, 1961, 1962, ...
Afghanistan, AFG, CO2 emissions (kt), 414.371, 491.378, 689.3
Angola, AGO, CO2 emissions (kt), 550.05, 454.708, 1180.774, 11
```

Uppgift A

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

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



```
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
```

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
```

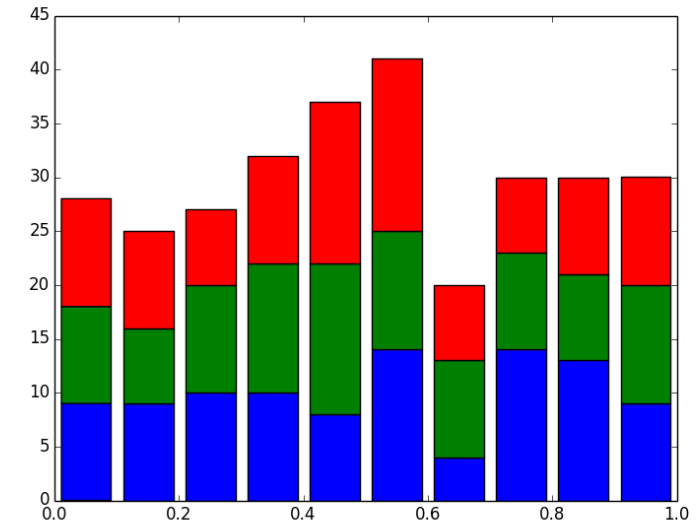
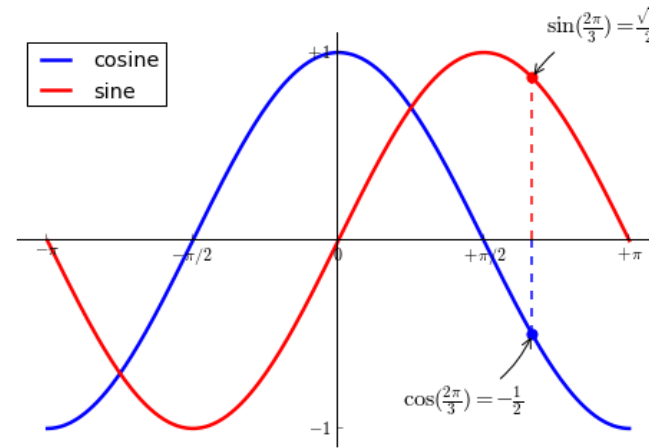
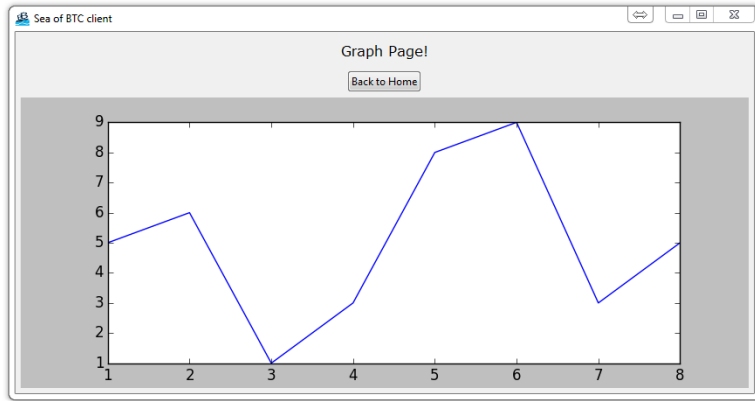
Att fundera på:

- Titta på filen `CO2Emissions_filtered.csv`. Finns det “skräp” i datat? (t ex rad 15, 29, 30). Kan programmet 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...



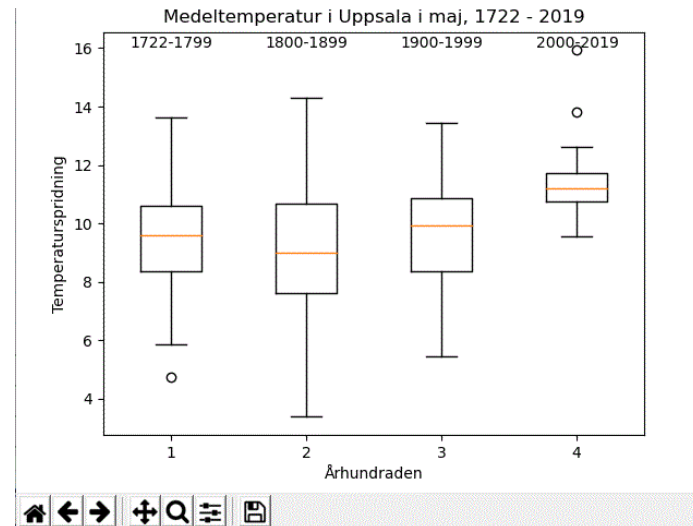
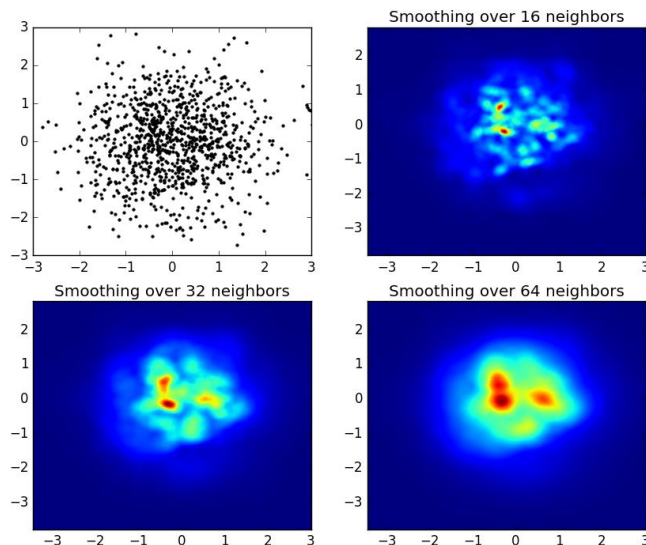
Datavisualisering med Python: matplotlib

“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.

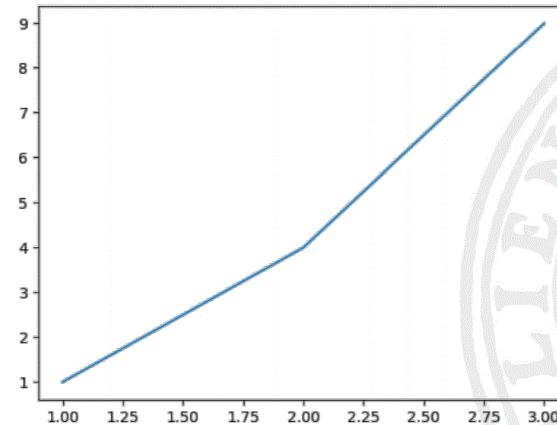
“Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.”

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.

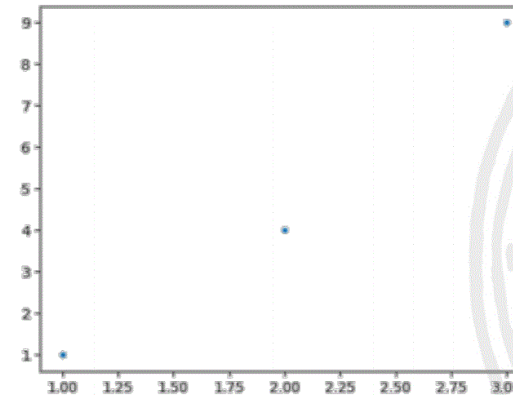
Några exempel

```
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!
```

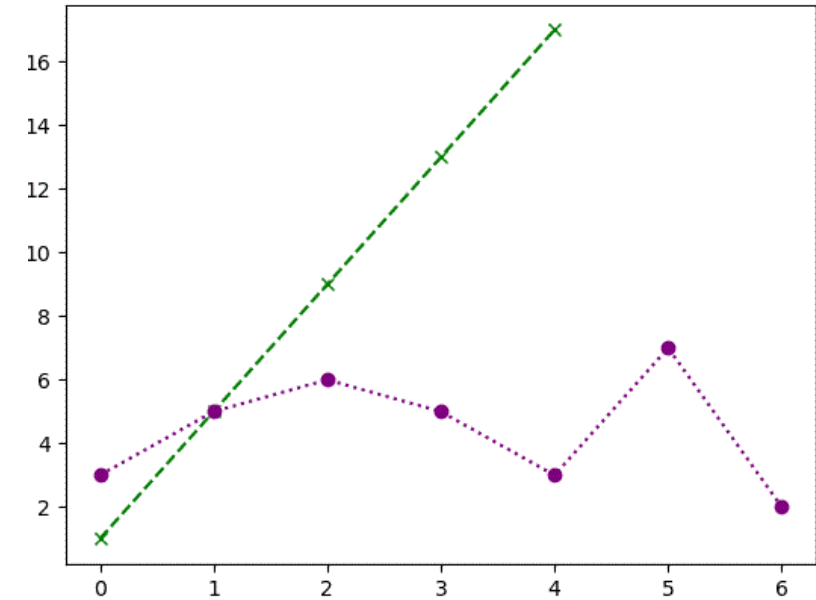
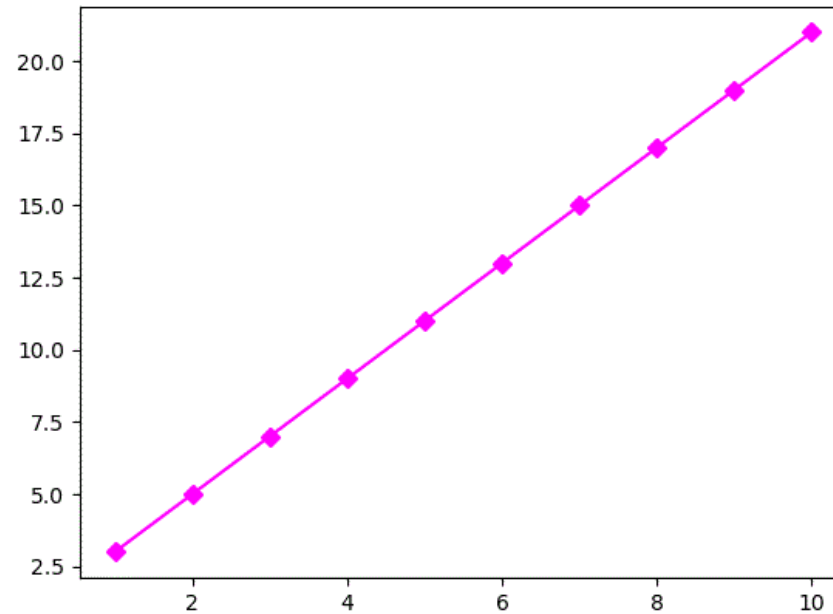
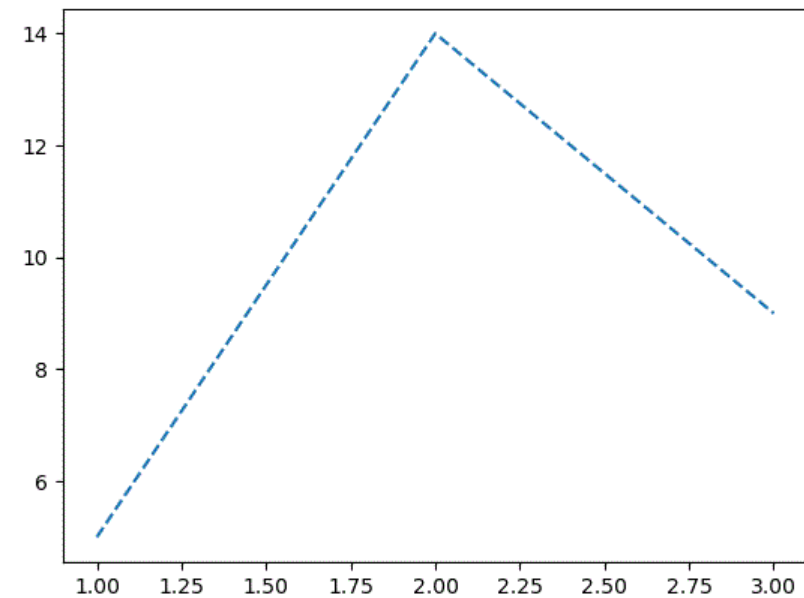
```
plt.plot([1,2,3],[1,4,9],"-")
# linje mellan (1,1), (2,4), (3,9)
plt.show()
```



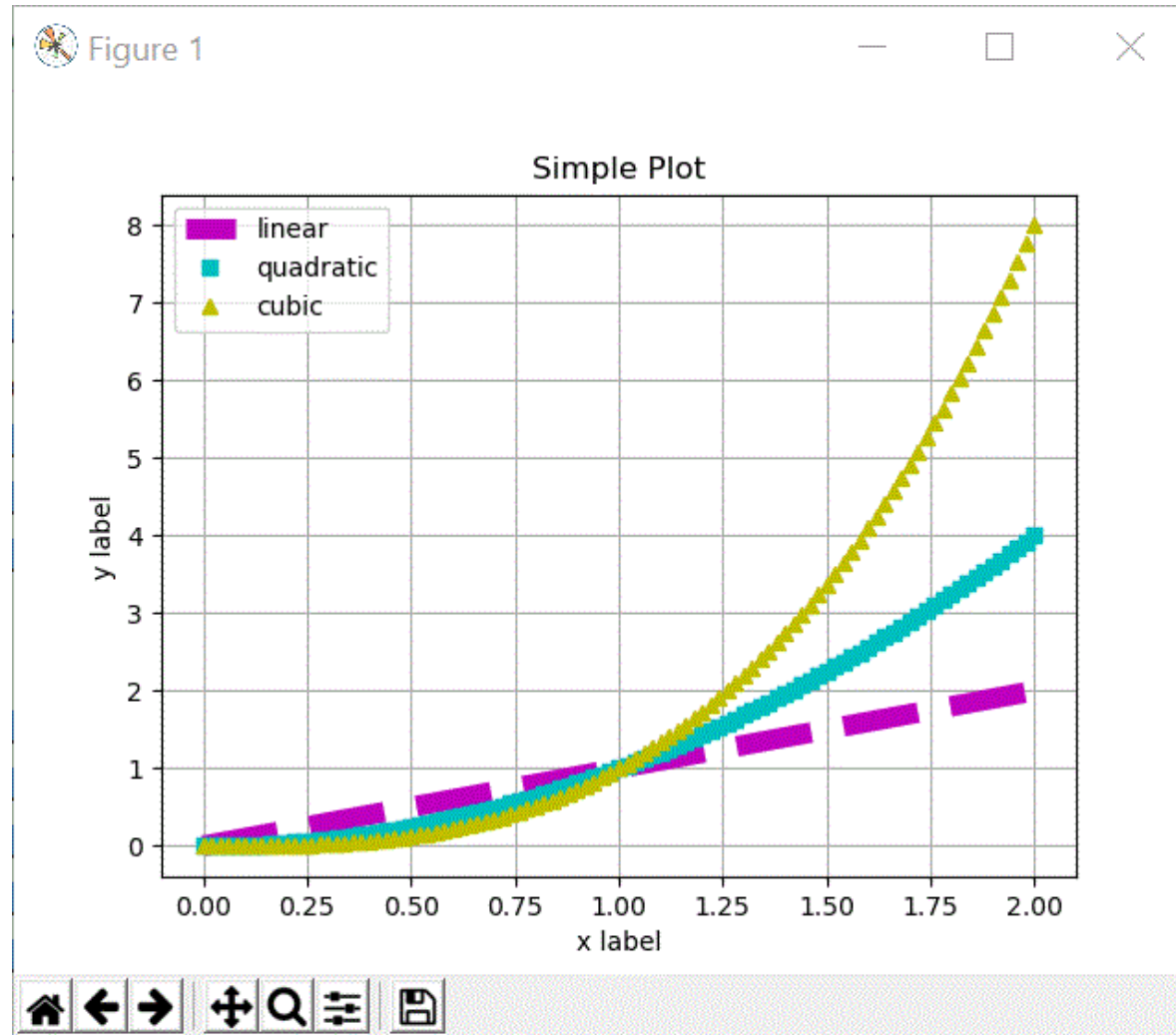
```
plt.plot([1,2,3],[1,4,9], ".")
# ritar punkterna
plt.show()
```



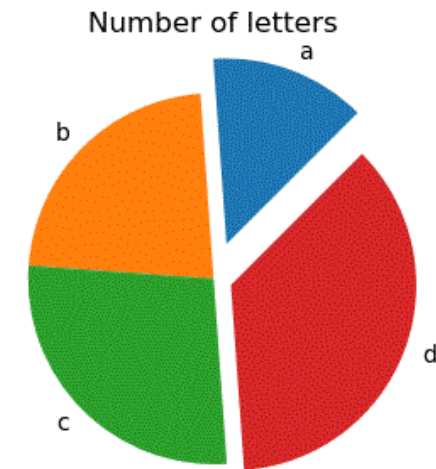
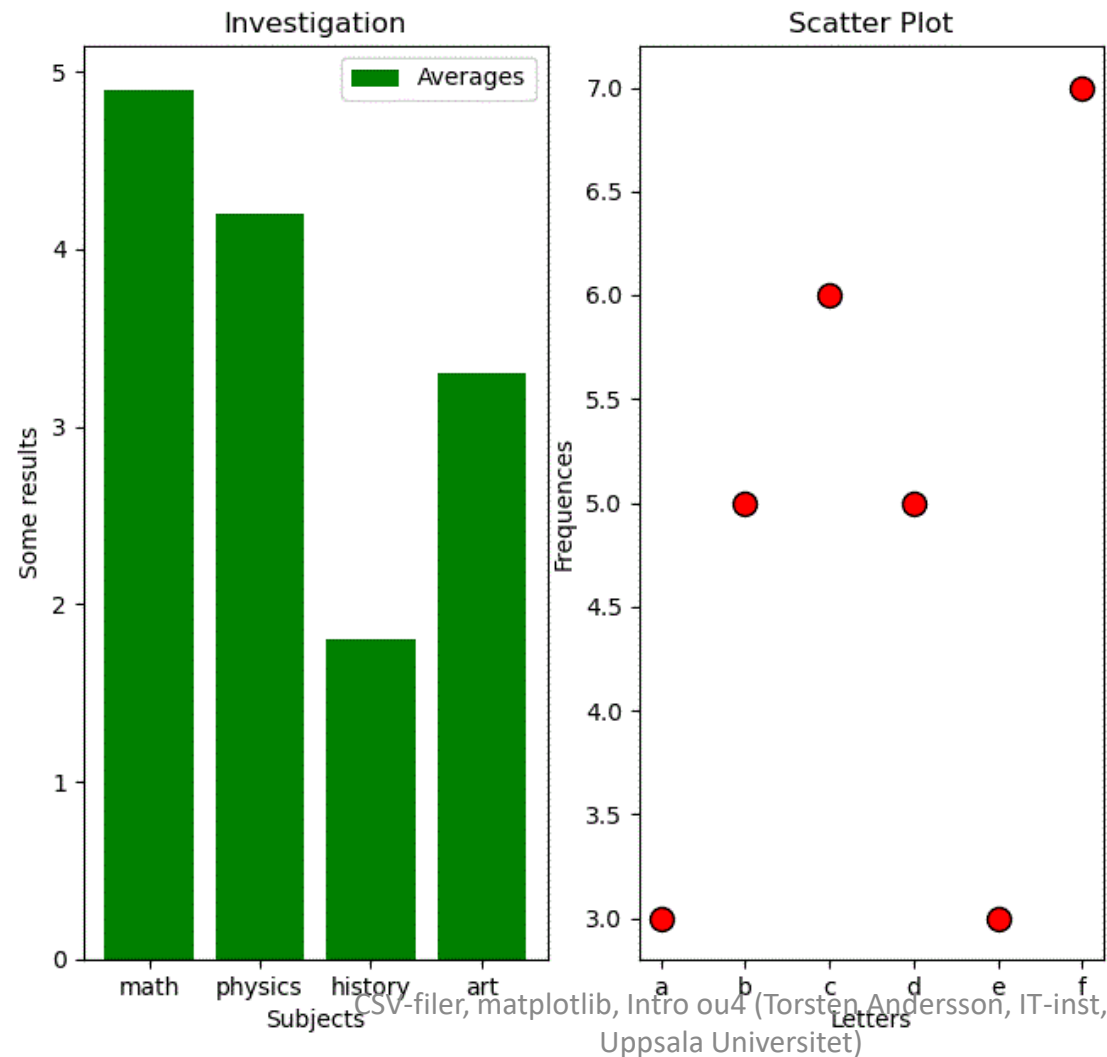
Ex1.py som skapar följande.



[Ex2.py](#) som skapar följande.



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



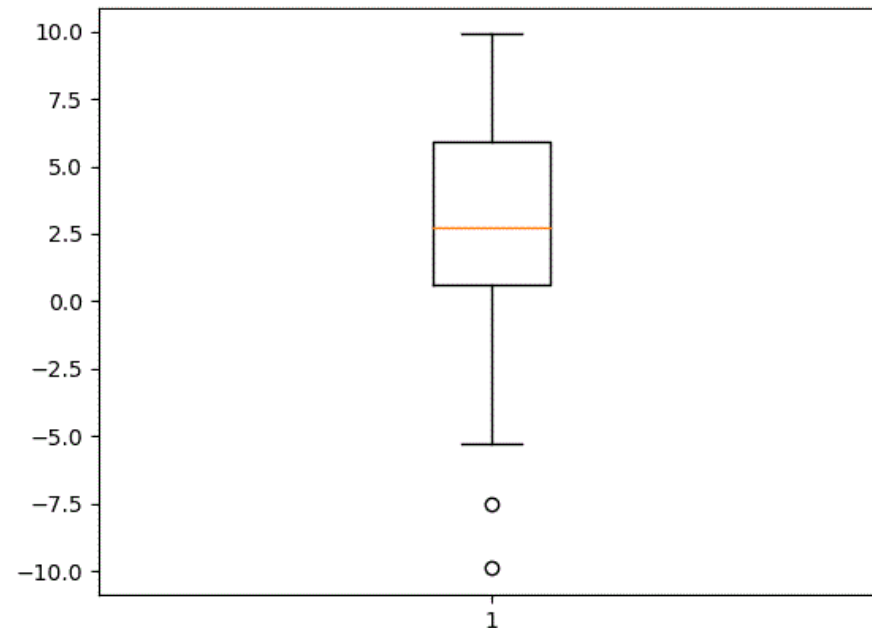
Boxplot (lådadiagram)

```
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()
```

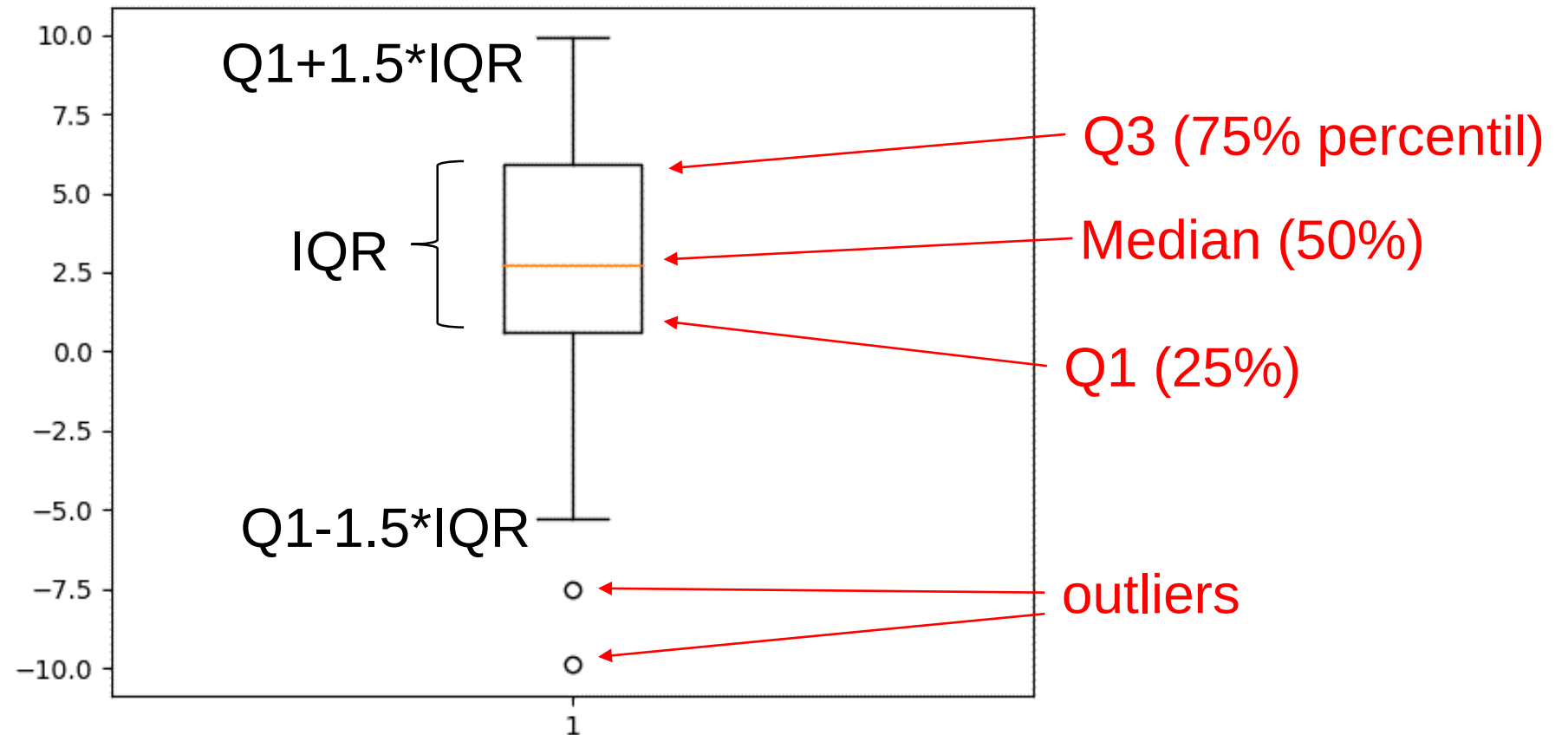
Figure 1



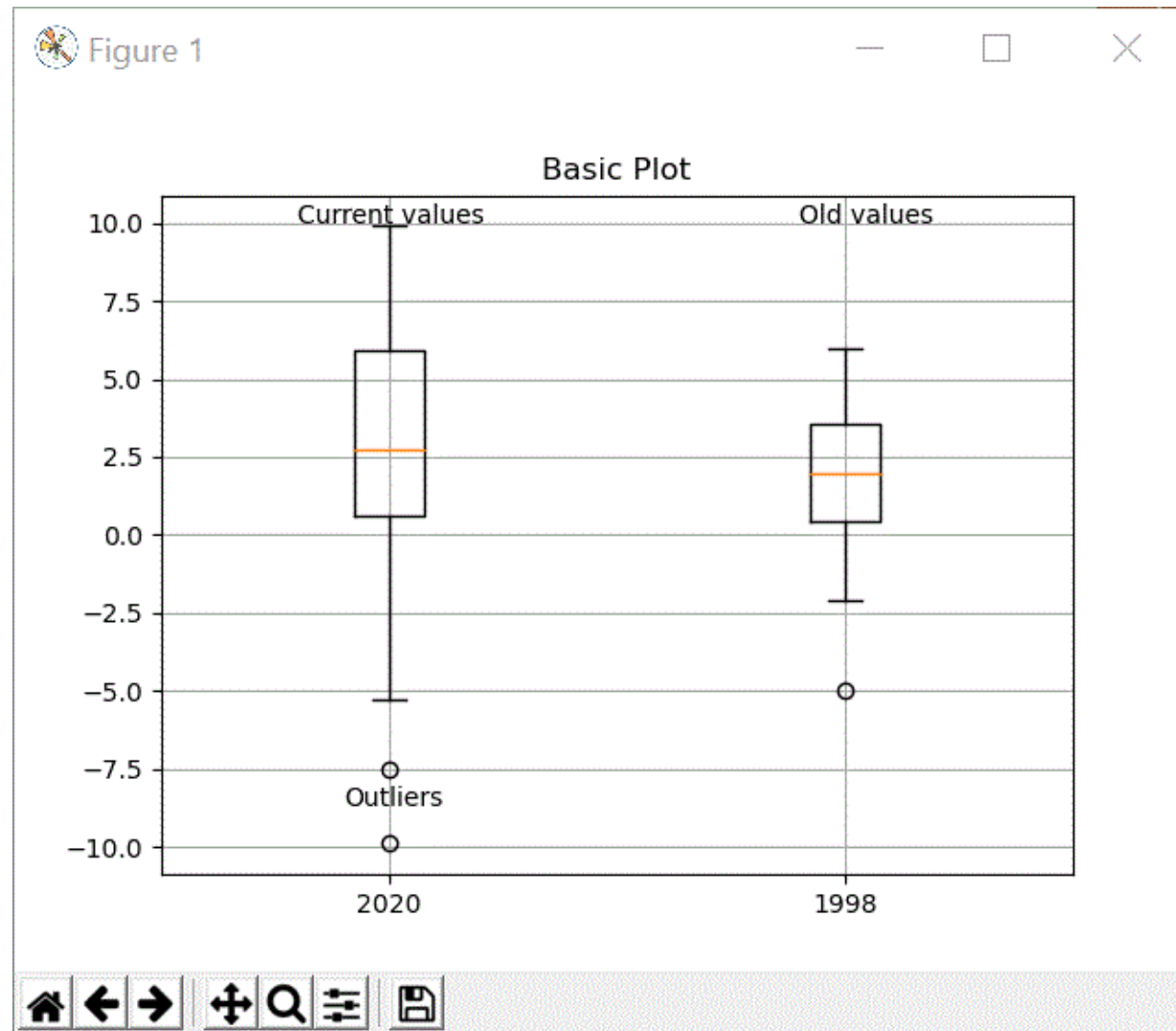
Förklaring ges på nästa sida →

```
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:

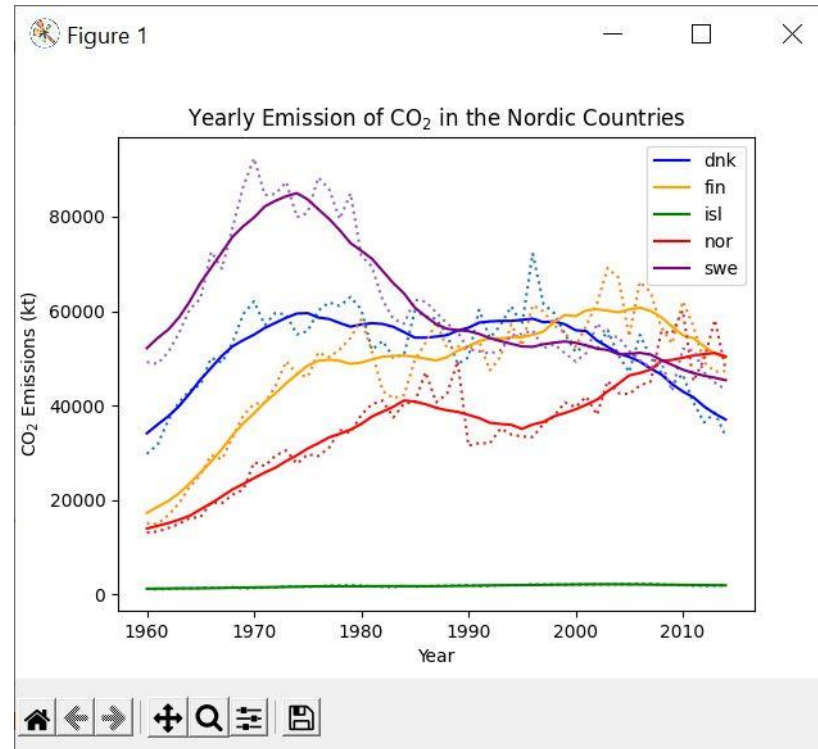
- `plot()` : Används för att plotta punkter eller 'linje'
- `label` : Används för att ge namn till graferna
- `xlabel()` / `ylabel()` : Används för att namnge axlarna (x och y) i grafen.
- `title()` : Sätter rubrik på grafen.
- `annotate()` : Lägg in text på angivna koordinater
- `legend()` : Används 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. Resultatet blir som den prickade kurvorna nedan, vilka visar sig vara ganska brusiga.
2. Minska bruset genom att (åter)använda funktionen `smooth_a` från Lektion 6. Illustrera resultaten från `smooth_a` med heldragen kurva och ursprungsdata med prickad kurva, för alla fem länderna.



Tio olika dataset, som
illustreras med varsin kurva

OU4, uppgift B

FÖRENKLAD ALGORITM, förslag (det fattas detaljer):

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

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

Anropa **load_csv** → returnerar 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 nytt lexikon

```
time = list(range(1960, 2015)) # Skapa en lista [1960, ...]
```

```
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**).

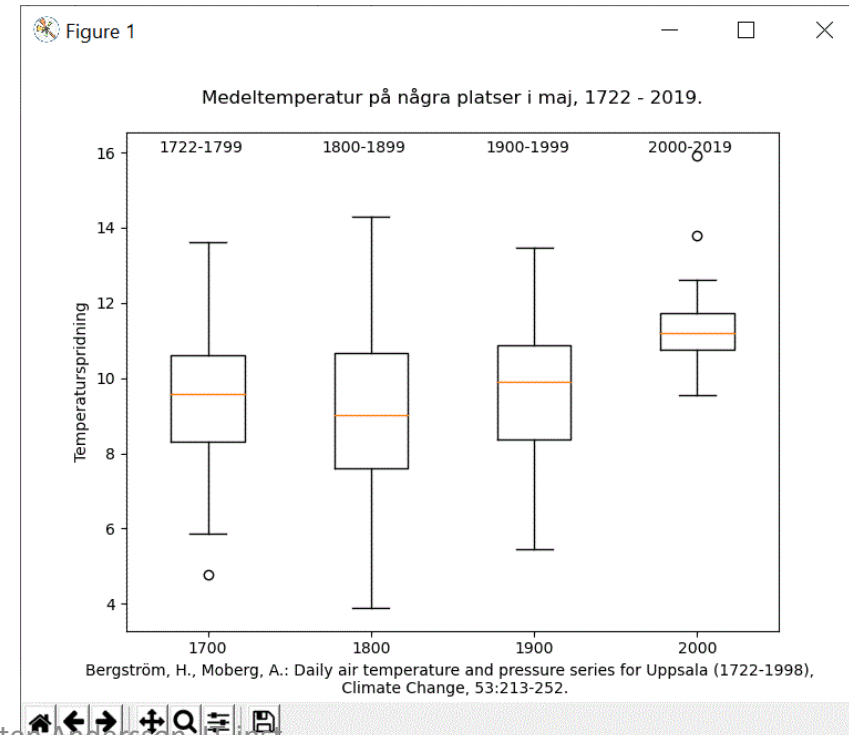
Förslag: Beräkna, spara i listor, dvs

[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_1722-2021.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....



Obs! Nu finns data till och med år 2021 i filen

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.

