Föreläsning 6

CSV-filer, matplotlib.
Genomgång och tips till Lektion 8 (obligatorisk uppgift 4)

Vi spelar in och lägger filmerna på Studium

FÖRELÄSNINGSANTECKNINGARNA och Python-exemplen finns på kurshemsidan

csv-data

- Python har modulen csv för att läsa och skriva s k 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:

```
import csv

with open('people.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    next(reader) # läser bort första raden
    for row in reader:
        print(row)
```

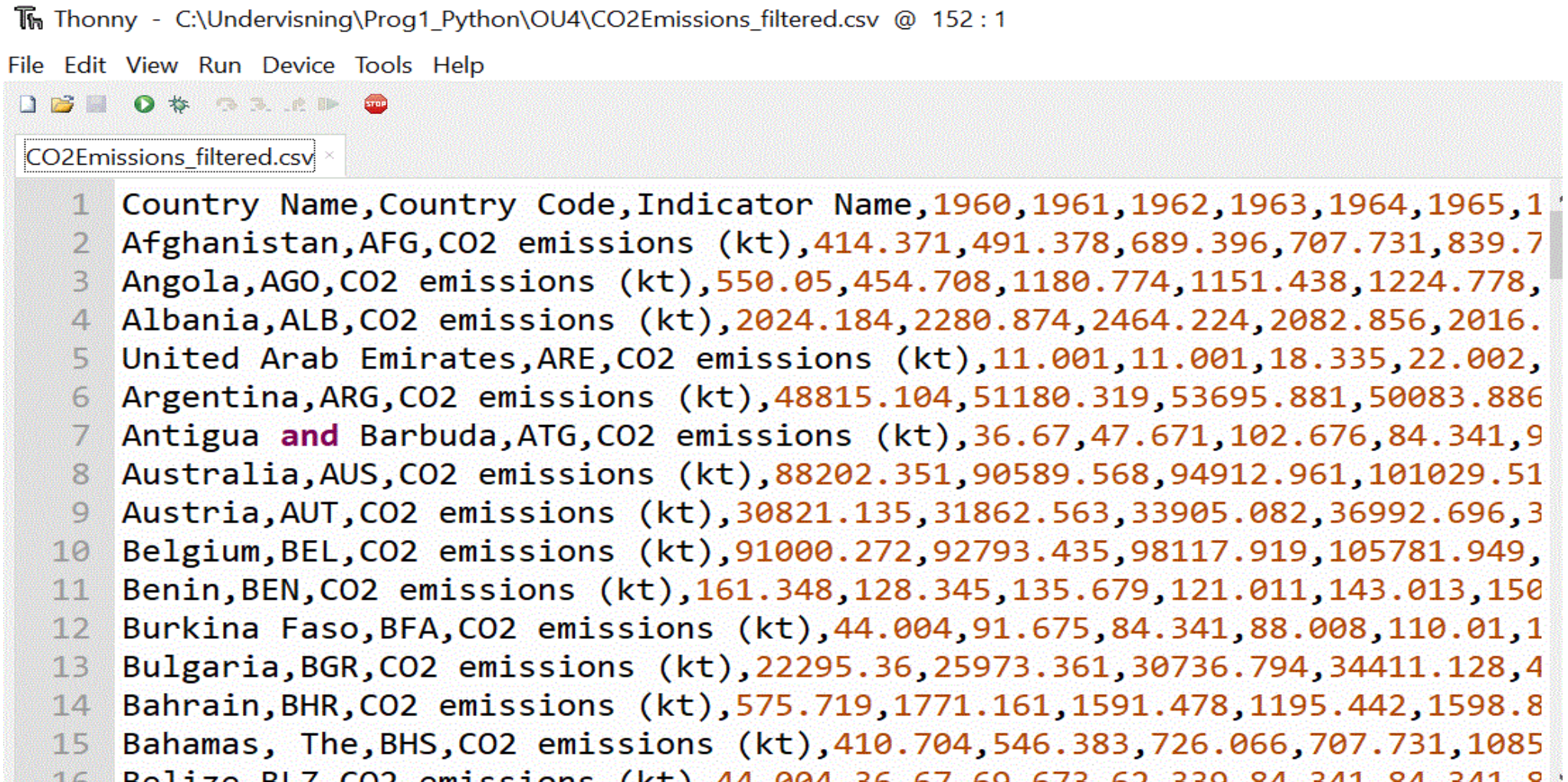
=>

```
['1', ' Alex', ' USA']
['2', ' Erik', ' Sweden']
['3', ' Cheng', ' China']
```

läs filen som en csv-fil.
reader-objektet itererar över varje
rad i filen, och returnerar ***listor***,
en per rad, där elementen är ***strängar***.
Det är ***komma*** som separerar elementen

cvs-data, lektion 8, uppgift A

A. Skriv en funktion load_csv(filename) som skapar och returnerar ett lexikon. Nycklarna är landskoder och värdena är listor med CO2-utsläppen (flyttal) för åren 1960 – 2014: {AFG: [414.371,491.378, ……..],AGO:[550.05,454.708,…],ALB:[….}



FÖRENKLAD ALGORITM (det fattas detaljer):

```
import csv


def load_csv(filename): # CO2Emissions_filtered.csv
    lex = {}
    with open(filename, 'r') as csvFile:
        reader = csv.reader(csvFile)
        for row in reader: # row blir en lista med strängar
            county_code = #det andra elementet på raden. Alltid?
            data = #gör en lista med all data på raden. Start-kolumn? Alltid? Tips: skivning
            #lägg in det nya key-value-paret landskod och data i lex
    return lex
```

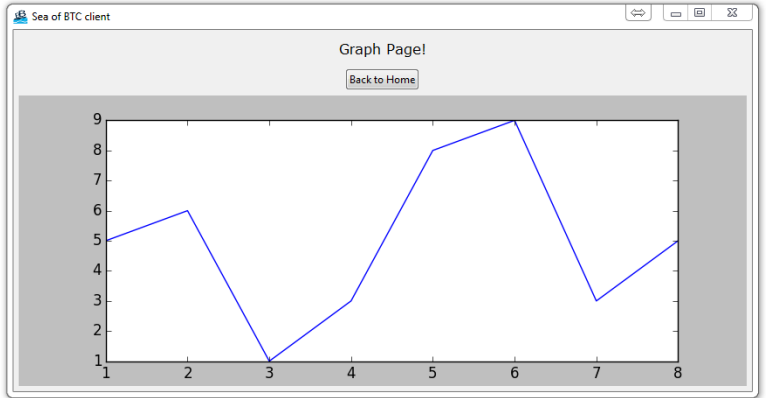
BREAK-OUT ROOM-uppgift:

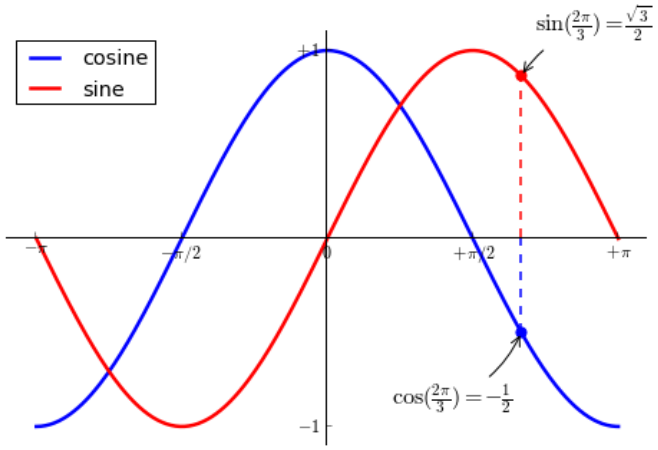
- Titta på filen CO2Emissions_filtered.csv. Finns det “skräp” i datat? (t ex rad 15, 147)
- Hur kan man hantera att raderna ser olika ut? (Ni får inte ändra i csv-filen)
- Vilken typ har datat?
- Fattas det något steg i algoritmen? (Kan läsa en enstaka rad med next(reader))
- Hur ska man felsöka om programmet krashar?

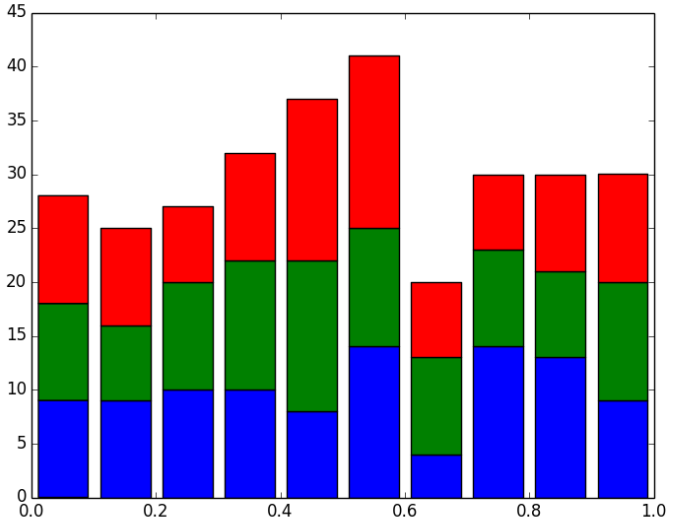
RAST 5 MINUTER

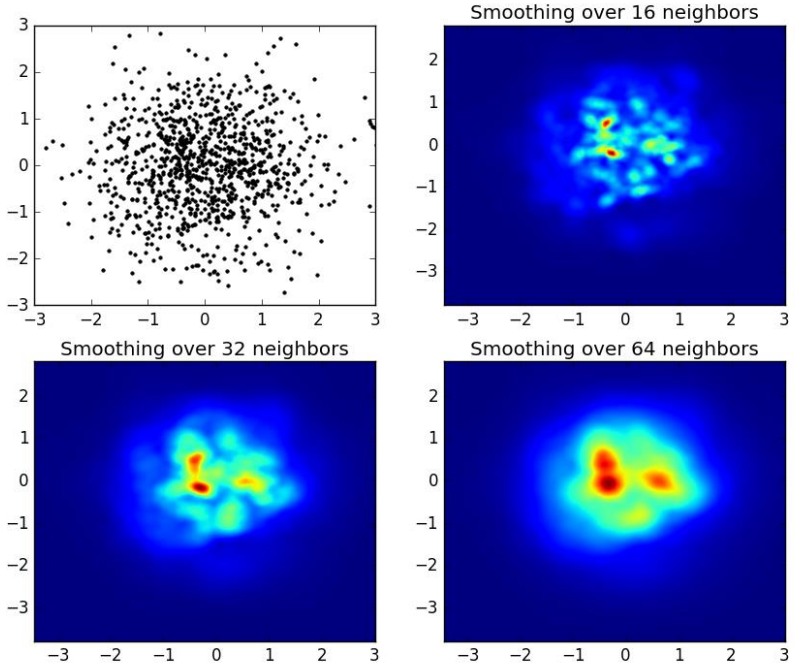
Datavisualisering med Python: matplotlib

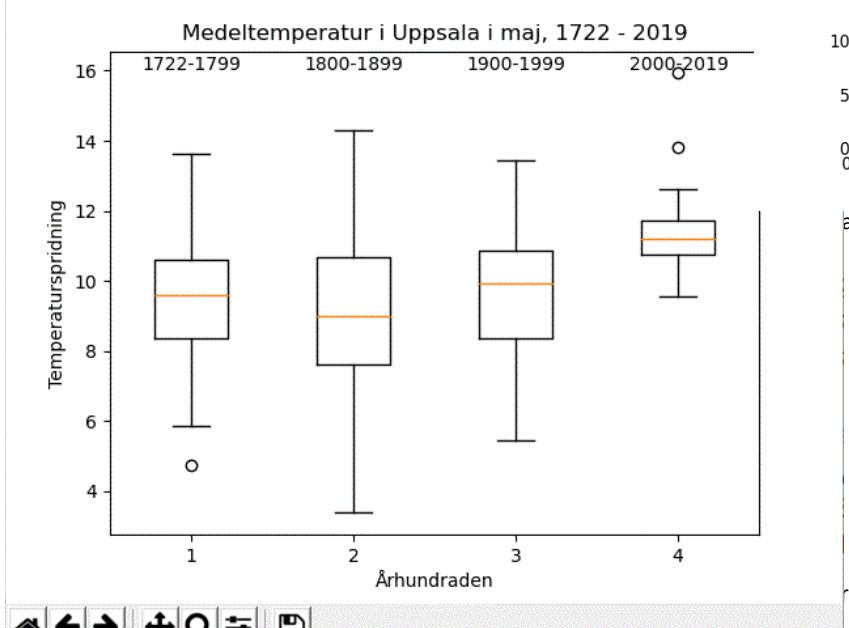
"Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python." **https://matplotlib.org/stable/tutorials/**











Datavisualisering med matplotlib (ex F6_ex1)

import matplotlib.pyplot as plt # pyplot är en modul i paketet matplotlib

#Rita linje resp. punkter:

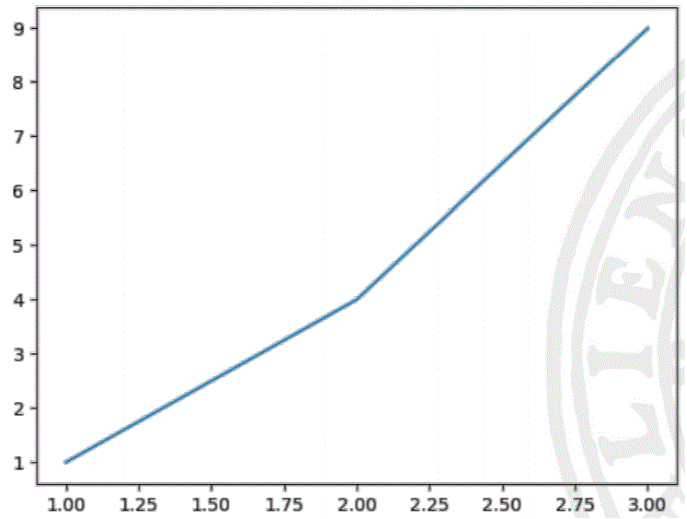
a) matplotlib.pyplot.plot([1,2,3],[1,4,9],"-") Onödigt långt!

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

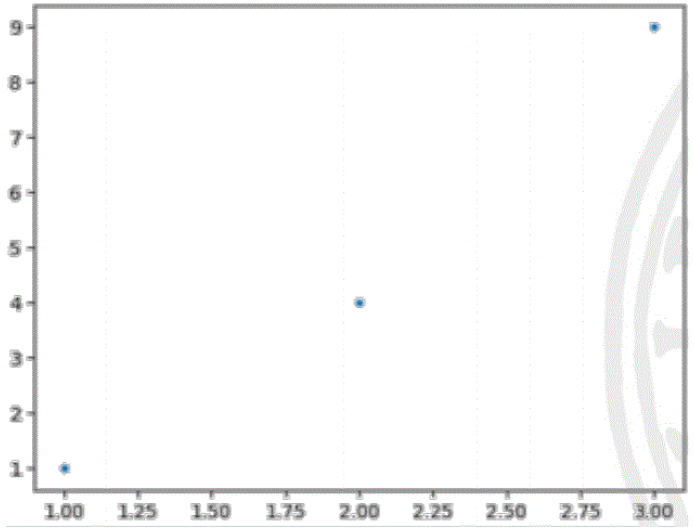
plt.plot([1,2,3],[1,4,9],”.") # b) ritar punkterna

plt.show()

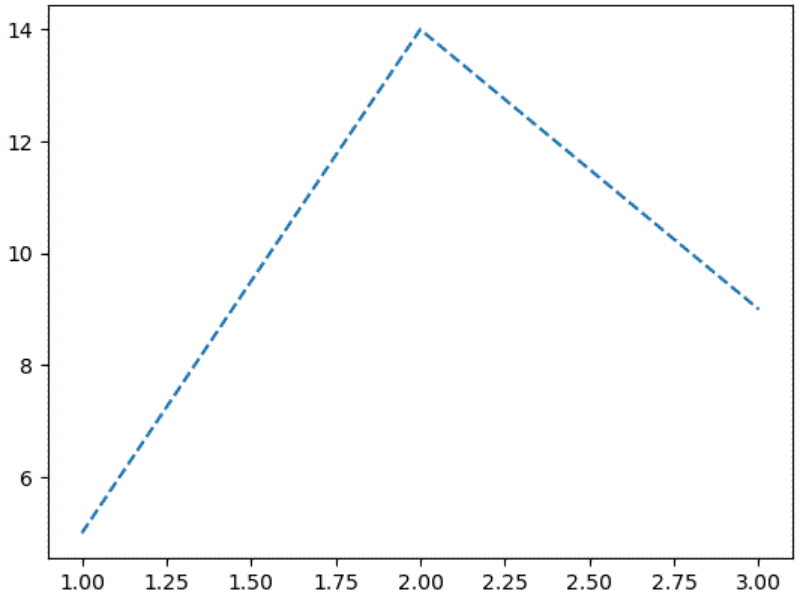
a)

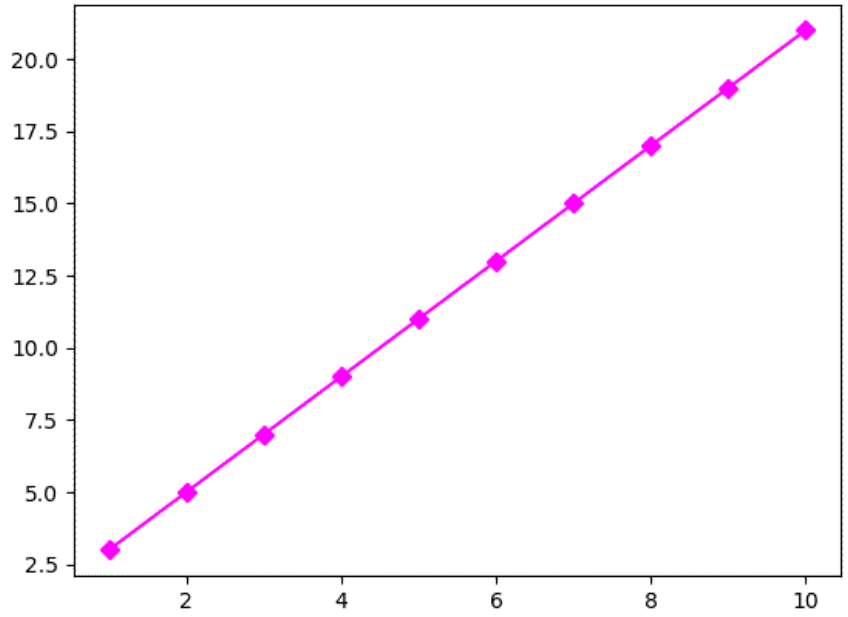


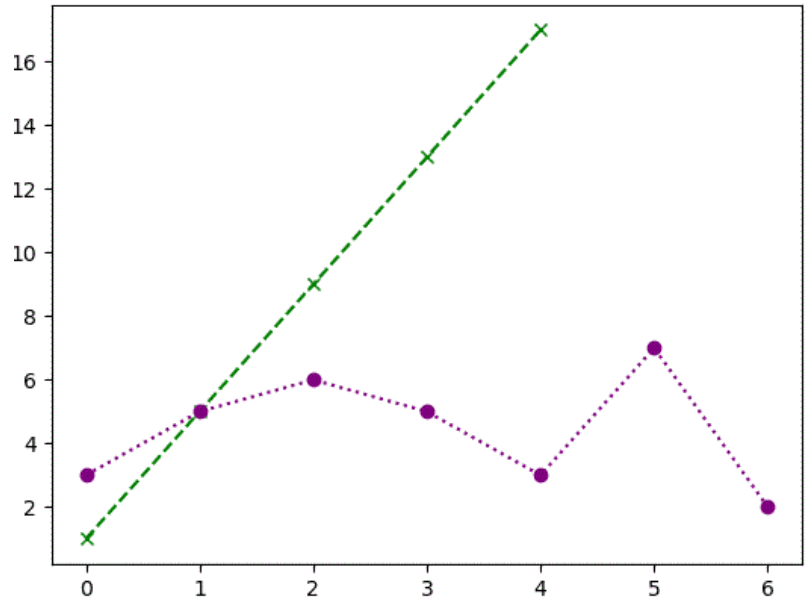
b)



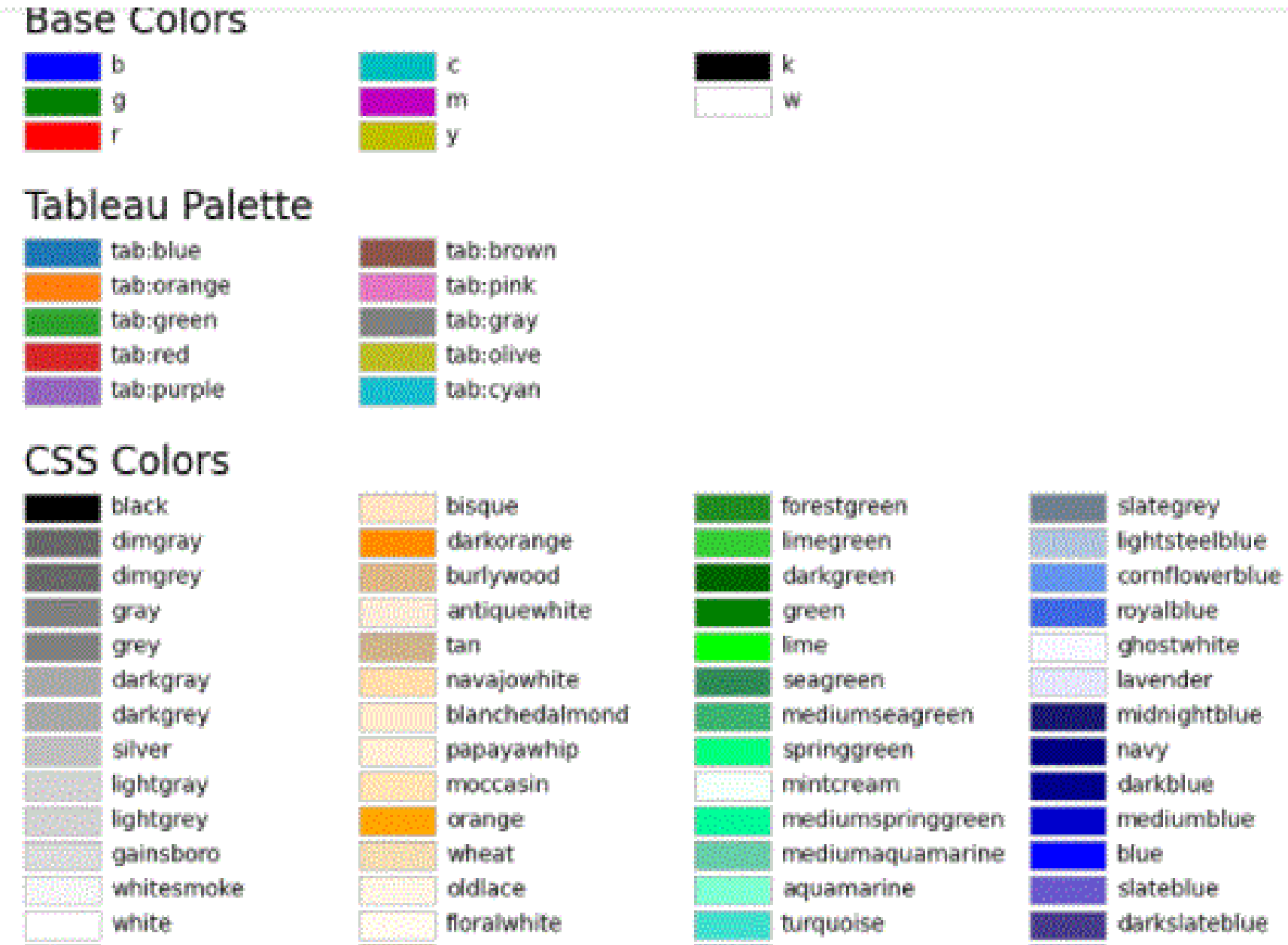
Datavisualisering med matplotlib (ex F6_ex1)



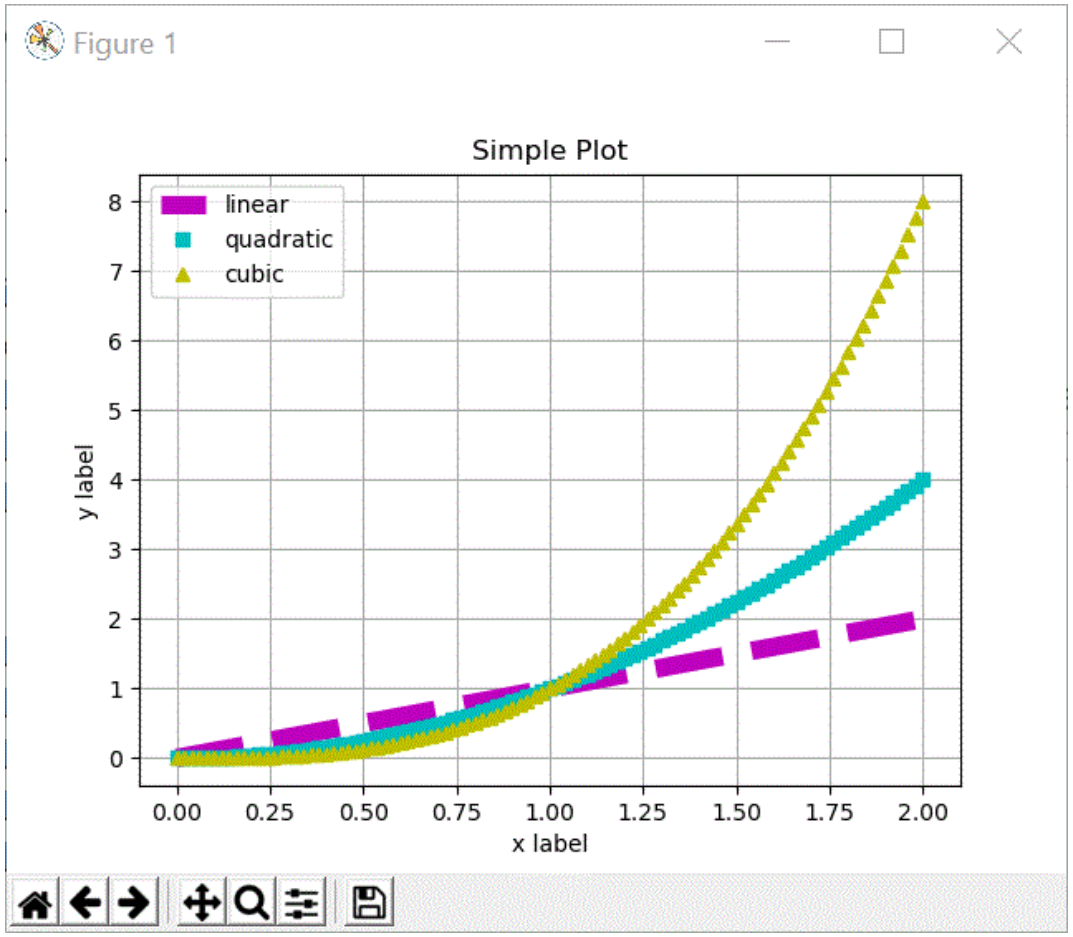




https://matplotlib.org/stable/gallery/color/named_colors.html



Datavisualisering med matplotlib (F6_ex2)



Se https://matplotlib.org/3.1.0/tutorials/

Datavisualisering med matplotlib

import matplotlib.pyplot as plt

Plotta linje:

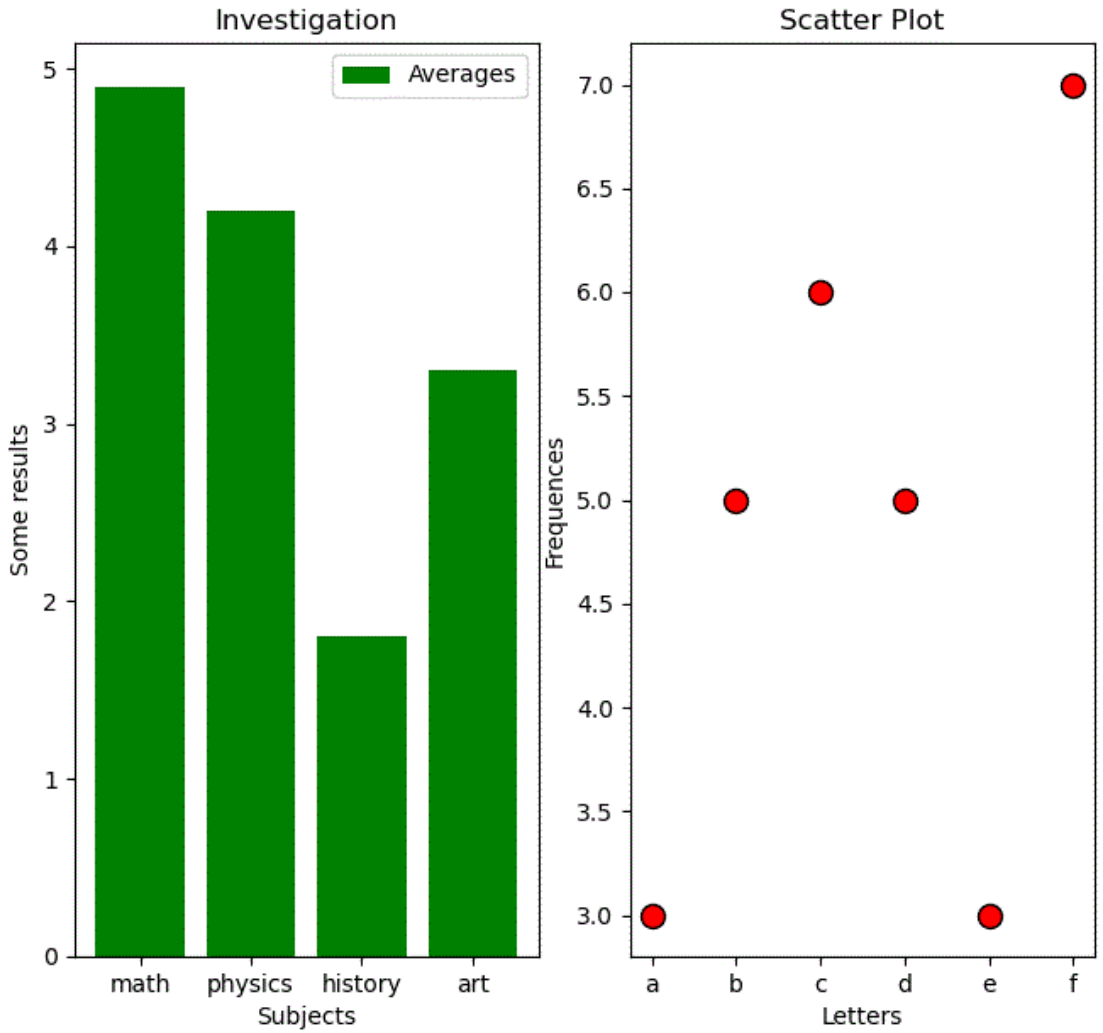
- antal x-värden == antal y-värde
- Linestyle (ls) t ex '-', '--', '-.', ':', 'None', 'solid', 'dashed', 'dashdot', 'dotted', …
- Marker t ex ^(triangle), o (circle), x, s (square), p (pentagon)
- Color t ex 'y','m','c','r','g','b','w' and 'k'
- Linewidth
- …

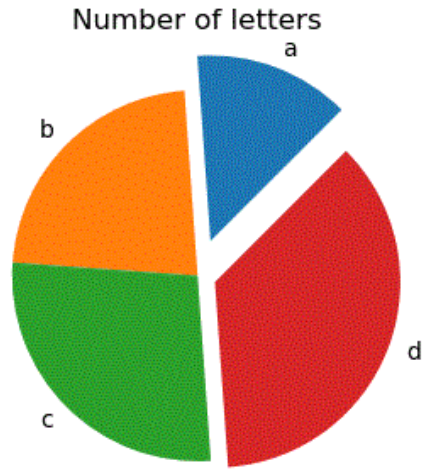
Exempel Ex3 visar hur man kan plotta flera diagram i samma fönster

Se https://matplotlib.org/

Datavisualisering med matplotlib

(F6_ex3_enskilda.py, F6_ex3)





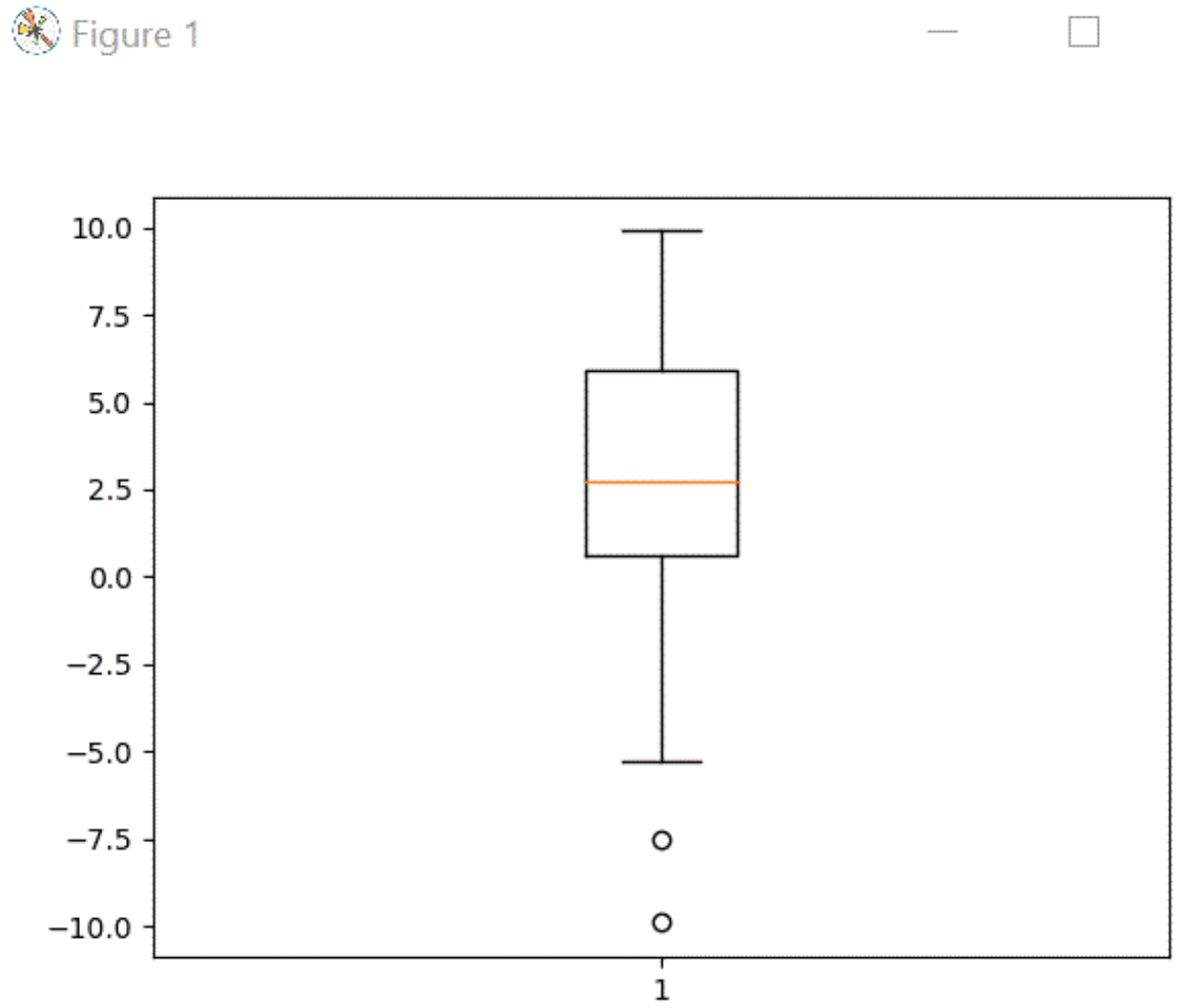
RAST 5 MINUTER

Datavisualisering med matplotlib

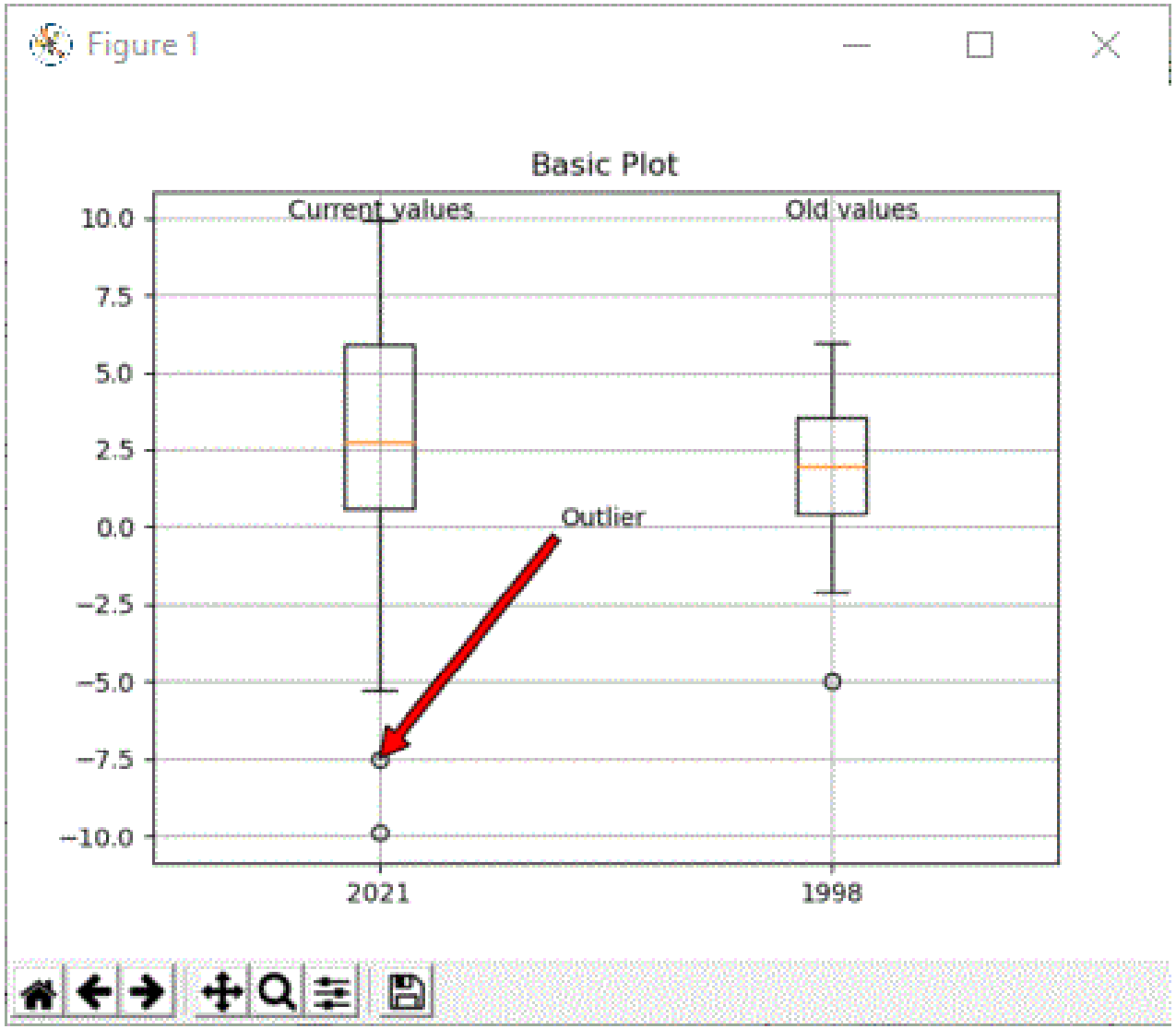
```
measures = [-1.2, 1.2, -5.3, -9.9, 0.7, -7.5, 0.4, 2.8, 2.7, 4.8,
            1.3, 1.1, 6.6, 4.2, 5.8, 6.5, 2.4, 3.4, 5.9, 2.4, -0.9,
            -1.5, 4.1, 8.3, 7.9, 9.2, 9.9, 6.0]
plt.boxplot(measures) # Lådagram
plt.show()
```

FRÅGOR:

1. Hur bestäms de horisontella strecken i ett lådagram (boxplot)?
2. Vad betyder de runda ringarna?



Datavisualisering med matplotlib (F6_ex4)



Datavisualisering med matplotlib

Några viktiga funktioner:

- plot(): Används för att plotta punkter eller ‘linje’
  - Parametern ‘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.

Datavisualisering med matplotlib

Exempel på olika diagram:

plt.plot(…) linjediagram

plt.bar(…) stapeldiagram

plt.scatter(…) punktdiagram/spridningsdiagram

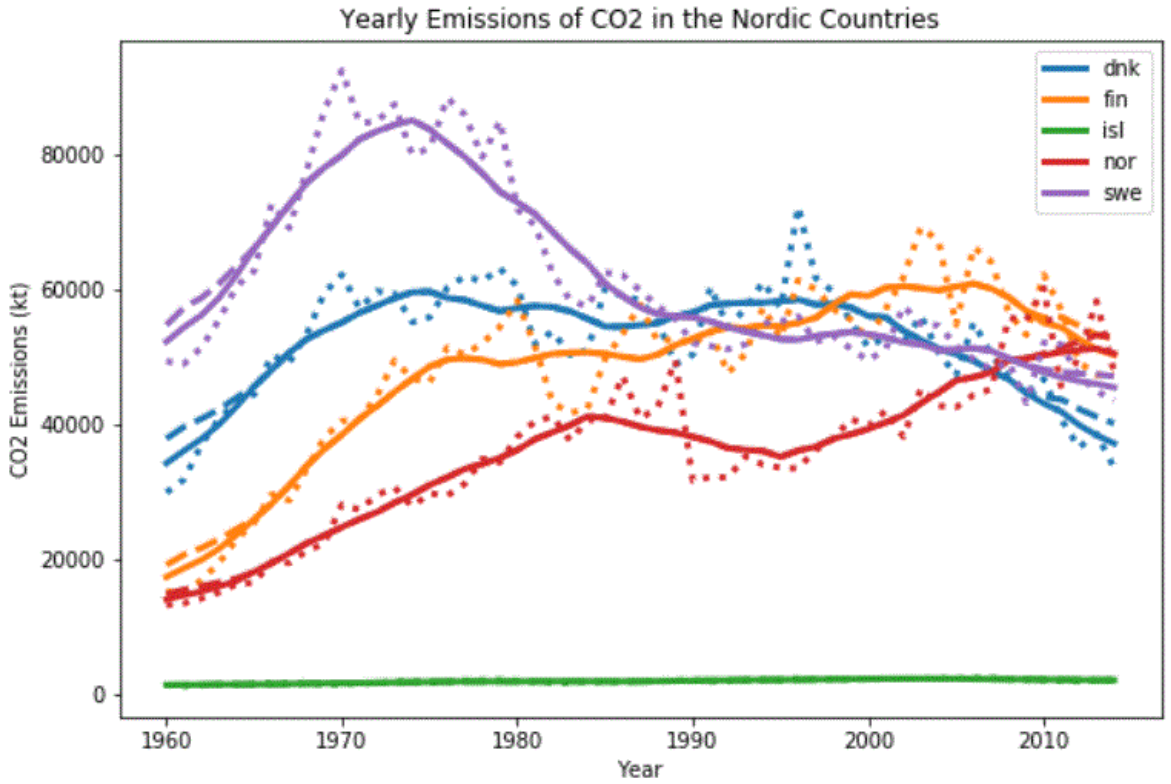
plt.pie(…) cirkeldiagram

plt.hist(…) histogram

plt.boxplot() lådagram: max, min, median, 1:a och 3:a kvartilen

matplotlib, lektion 9, 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 använda medelvärdesfunktionerna smooth_a från Lektion 6, och smooth_b om du gjort den.



matplotlib, lektion 9, uppgift B

FÖRENKLAD ALGORITM, förslag (det fattas detaljer):
diverse kod, t ex from *filnamnet* import load_csv
Gör lista med landskoderna 'dnk', 'fin', 'isl', 'nor', 'swe'
Anrop av load_csv ger {AFG: [414.371,491.378, ……..],AGO:[550.05,454.708,…],ALB:[….}
Loopa igenom lexikonet från funktionen load_csv:
    är det landskoden för ett nordiskt land?
        spara i så fall datat i ett 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 figuren
    ax.plot(# använd smooth_a för y-värdena
    …

RAST 5 MINUTER

matplotlib, lektion 9, uppgift C

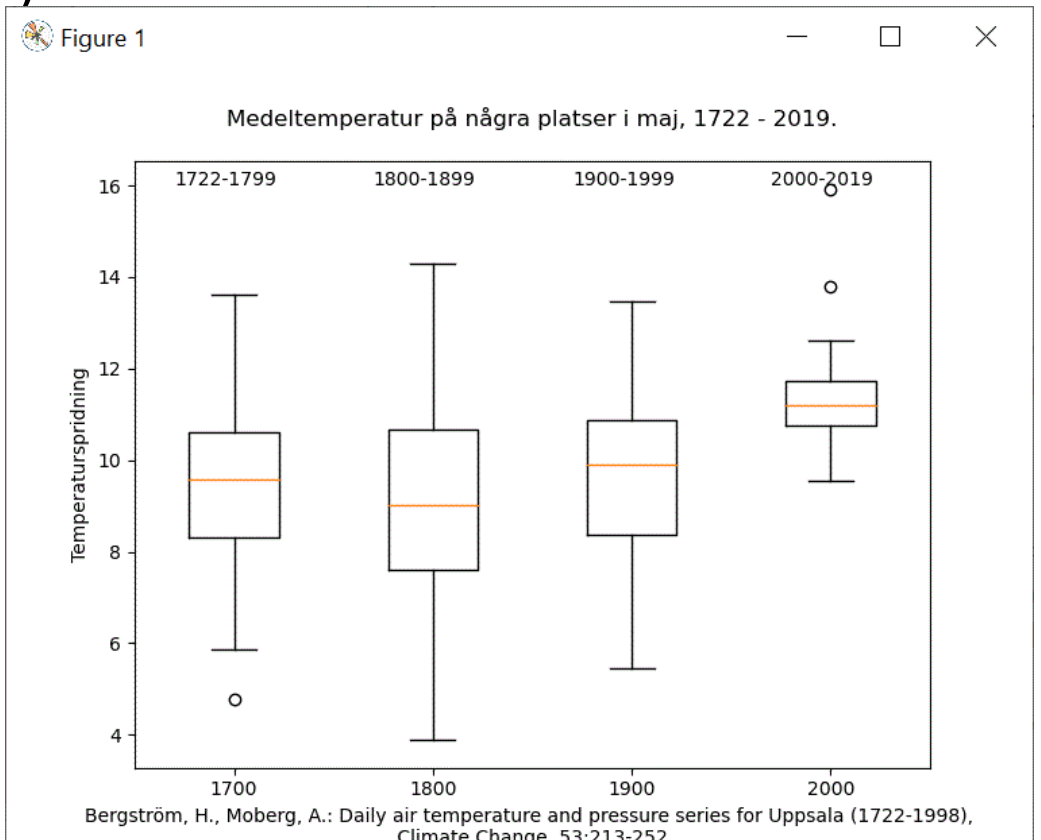
Rita upp lådagram (boxplot) för maj månads medeltemperaturer för Uppsala uppdelat på 1700-, 1800-, 1900- och 2000-talen. Hämta data från textfilen uppsala_tm_1722-2019.dat från SMHI. (***OBS! Inte csv-fil***). Ex:

Beräkna och spara i listor: [*medeltemp maj 1722, medeltemp maj 1723, ….,medeltemp maj 1799*]:

list_avarage_per_year_1700=[*9.8, 8.7, … , 10.1*] # för hela 1700-talet
list_century.append(list_ avarage_per_year_1700)

Filen uppsala_tm_1722-2019.dat:

År	mån	dag	medel	…	se readme-fil
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



matplotlib, lektion 9, uppgift C

Öppna filen uppsala_tm_1722-2019.dat för läsning (inte en csv-fil)

För varje rad i filen:

    ta bort mellanslag - ta bara med decimaltal. Reguljära uttryck eller
        split()-funktionen.

    lägg raden i en lista

För varje rad i listan med all data:

    summera temperaturerna i kolumn 4 för maj månad

Beräkna medeltemperaturen.

Spara i en lista

Rita upp

Det fattas saker i algoritmen….

matplotlib, lektion 9, uppgift C

NB! Users of the file 'uppsala_tm_1722_2019.dat are asked to refer to:
Bergström, H., Moberg, A. :
Daily air temperature and pressure series for Uppsala (1722-1998), Climate Change, 53: 213-252.

```
# FÖRENKLAD ’ALGORITM’, ett förslag (det fattas detaljer):
import matplotlib.pyplot as plt, re

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

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
for row in all_text:
    if int(row[0])==this_year and … #om det är sama år, this_year, och maj:
        sum_temp += float(row[3]) #summera temperaturen i maj det året
    elif int(row[0]) > this_year #Nytt år?
            this_year = …# byt 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
```