Grafieken met Python deel 1.

Overal om ons heen wordt data gegenereerd en verzameld.
Python speelt een steeds belangrijkere rol in het verzamelen
en analyseren van data, denk aan temperatuur, luchtvochtigheid,
gewicht, snelheid, kracht, dichtheid, frequentie, verkeersdrukte enz.

Vaak wil je data analyseren en visualiseren om inzicht in de
materie te krijgen en om verbanden en trends te ontdekken.

Python heeft daar uitstekende gereedschappen voor.

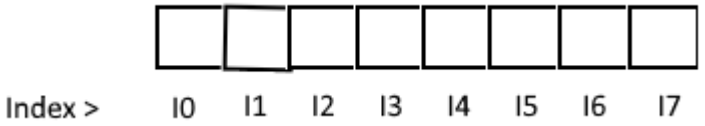
Op de eerste avond van deze workshops houden we ons bezig
met het opslaan en manipuleren van data en de middelen die
we tot onze beschikking hebben.

We kijken naar lists als een container van data.
Een list is een data array waarin je een reeks gegevens kunt opslaan.
Een list geef je een naam en behandel je als een variabele.

Grafieken met Python deel 1.

Wat is een list in Python?

Een list is op het eerste gezicht een eenvoudige data structuur waarbij elk data element een volgnummer heeft. Het eerste data element heeft de index 0, en bij n elementen in de list heeft het laatste element index n-1. Een list is een enkelvoudig array Van bijna onbeperkte lengte.



In Python wordt een list aangeduid met vierkante haken.
De data staat tussen de haken gescheiden door komma's.
Voorbeeld:

a = [1, 2, 3, 4, 5] een list met getallen
b = [“Jan”, “Piet”, “Kees”] een list met strings

print (a) levert als output: [1,2,3,4,5]
print (b) geeft als output [“Jan”, “Piet”, “Kees”]

Grafieken met Python deel 1.

In een list kun je allerlei soorten data opslaan.
Bijvoorbeeld integers, floats en strings.

Opdracht: start je IDE op en maak een list met de naam sport.
Vul deze list met tenminste 5 sporten.

De sporten in de list worden onderling gescheiden door een komma. Een string moet tussen aanhalingstekens staan.

Print de inhoud van de list sport met : *print (sport)*

Sorteer de list oplopend met: *sorted (sport)*
De volgorde van de originele list wordt niet gewijzigd.

Print de inhoud met: *print (sorted (sport))*. Wil je de de gesorteerde list bewaren dan moet je hem opslaan onder de zelfde of een andere naam bijv: *sport = sorted (sport)*

Grafieken met Python deel 1.

sport = [“schaatsen”, “skïen”, “voetbal”, “volleybal”, “hardlopen”]
sport.sort ()) # sorteerd de elementen in oplopende volgorde.

print(sport) geeft ['hardlopen', 'schaatsen', 'skien', 'voetbal', 'volleybal']

Je kunt sport ook in een aflopende volgorde sorteren met:
sport.sort (reverse = True)

Met *print(sorted(sport))* kunnen we sorteren zonder wijziging van de list

We kunnen op verschillende manieren data toevoegen aan een list.
Voorbeeld: *sport.insert(0, “wielrennen”)*
De nieuw toegevoegde data wordt op index 0 dus vooraan geplaatst.
Met het index getal geven we aan waar de data geplaatst moet worden.

De insert functie is traag omdat alle index getallen na de insert index in de list moeten worden aangepast.

Grafieken met Python deel 1.

Een andere methode om data aan een bestaande list toe te voegen doen we met *sport.append (“fierljeppen”)*.

Onze list ziet er dan a.h.v. uit:
sport = [“schaatsen”, “skïen”, “voetbal”, “volleybal”, “hardlopen”, “fierljeppen”]

De append functie voegt nieuwe data altijd aan het eind van de list toe.

Je kunt append heel goed toepassen als je de uitkomsten van een periodieke meting op wilt slaan in een list.

Data uit een bestaande list verwijderen kan op twee manieren.
sport.del(3), data met index 3 hier is dat “volleybal “ wordt uit de list verwijderd.
Met *sport.remove(“skïen”)* wordt skïen uit de list verwijderd

Wil je alleen het laatste element verwijderen, gebruik dan *sport.pop ()*.
laatste_element = sport.pop () je gebruikt het index getal om een specifiek element te verwijderen.
print (laatste_element)

Grafieken met Python deel 1.

Je kunt bij een list ook hoogste en laagste waarde opvragen.
Dat kan een voordeel zijn bij het bepalen van de schaal van een grafiek.
Bijvoorbeeld een list met gemeten temperaturen.

Voorbeeld: *temp = [21, 21, 22, 22, 23, 24, 24, 25, 27, 29, 20, 19, 15]*
laagste = min(temp)
print (laagste) levert als uitkomst 15.

hoogste = max(temp)
print (hoogste) levert als uikomst 29.

Wil je de gemiddelde temperatuur weten dan moet je de som van alle elementen delen door het aantal elementen

totaal = sum(temp) *print (totaal)* geeft als uitkomst is 292
aantal = len(temp) *print (aantal)* geeft als uitkomst is 13

gemiddeld = (totaal/aantal) *print (gemiddeld)* geeft als uitkomst is 22.46

Grafieken met Python deel 1.

Je kunt ook kijken hoe vaak een bepaald element in de list voorkomt.
temp = [21, 21, 22, 22, 23, 24, 24, 25, 27, 29, 20, 19, 15]

temp_22 = temp.count(22)
print(temp_22) het resultaat is 2.

Wil je weten of bijvoorbeeld 28 in de list voorkomt gebruik dan:

```
if 28 in temp:
    print ("Waarde komt voor in lijst")
else:
    print ("Waarde komt niet voor in lijst")
```

Je kunt ook lists samenvoegen bijvoorbeeld temp + temp2

temp = temp.extend (temp2)
De elementen van list temp2 worden achteraan toegevoegd aan temp

Grafieken met Python deel 1.

Soms heb je voor een grafiek een lineair oplopende
reeks getallen nodig, bij voorbeeld de dagen in een maand.
Zo een reeks kunnen we genereren met de range functie:

```
reeks = [ ]
for getal in range (16):
    reeks.append(getal)
print (reeks)
```

Dat levert de volgende lijst op:
[0 ,1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Het kan ook korter.
Een lijst genereren met het list statement.

```
getallen = list (range ( 1, 11))
print ( getallen)
```

Levert de volgende list:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Grafieken met Python deel 1.

Een list genereren met tijdstippen waarop bijvoorbeeld een actie wordt uitgevoerd zoals een temperatuur, windsnelheid of luchtdruk meting.

```
#tijd_lijst.py
import time
tijdlijst = [ ]
a = 0
while a < 10:
    tijd = (time.strftime( "%H.%M.%S"))
    tijdlijst.append(tijd)
    time.sleep (5)
    print (tijdlijst)
    a = a + 1
```

```
['14.51.28', '14.51.33', '14.51.38', '14.51.43', '14.51.48', '14.51.53', '14.51.58', '14.52.03', '14.52.08', '14.52.13']
```

Er worden 10 tijdstippen vastgelegd met tussenpozen van 5 seconden.

Grafieken met Python deel 1.

Een deel van een list uitknippen (list slicing).

```
# list_slice
getallen = [1, 2, 3, 4, 5, 6, 7 , 8]
eerste_drie = getallen [:3]
print ( eerste_drie) # levert [1, 2, 3]
print (getallen [0:3]) # verkorte schrijfwijze

Middelste = getallen [3:5]
print (middelste) # levert [4, 5]
print (getallen [3:5]) # verkorte schrijfwijze

laatste_drie = getallen [-3:]
print (laatste_drie) # levert [6, 7, 8]
print (getallen [-3:]) # verkorte schrijfwijze
```

Grafieken met Python deel 1.

Lists zijn lokale variabelen, als je in het programma een lijst creëert zal deze bij het afsluiten van het programma verdwijnen.

Wil je de inhoud van een list bewaren, dan moet je hem opslaan als tekstbestand. Hiervoor gebruiken we het csv formaat (comma separated values).

Python heeft hiervoor een uitgebreide csv module.

Wij gaan hier alleen kijken naar het schrijven en lezen van csv bestanden.

Om met deze bestanden te kunnen werken, moeten we de csv module van Python in ons programma importeren

import csv

Grafieken met Python deel 1.

Het programma creëert een list met daarin 10 tijdstippen.
De tijden in de list liggen 1 minuut uit elkaar. Als de list voltooid is wordt met de functie csv.writer, tijdlijst.csv gemaakt.

```
# tijdlijst.csv
import time # importeer de time module
import csv # importeer de csv module
tijdlijst = [] #creëer een lege lijst
a = 0 # creëer een variabele als teller
while a < 10: # while doorloopt 10 X de statements
    tijd = (time.strftime("%H.%M")) # lokale tijd opslaan in tijd
    tijdlijst.append (tijd) # voeg de waarde in tijd toe aan tijdlijst
    time.sleep(60) # wacht 1 minuut
    print(tijdlijst) print de waarden van tijdlijst
    a = a + 1 # verhoog de teller a met 1
with open ("tijdlijst.csv", "w") as csvFile: # open een bestand tijdlijst.csv
    writer = csv.writer(csvFile) # start de csv writer functie als writer
    writer.writerow(tijdlijst) # schrijf de list tijdlijst weg als csv bestand
    csvFile.close()  # sluit de tijdelijke csvFile

# met een tekstverwerker kun je het bestand tijdlijst.csv bekijken.

# met een spreadsheet programma kun je tijdlijst.csv importeren en bewerken.
```

Grafieken met Python deel 1.

Data uit een spreadsheet importeren.

	A	B	C
1	Datum	Regenval	Temp
2	01-03-18	0	7
3	02-03-18	0	6
4	03-03-18	2	4
5	04-03-18	7	7
6	05-03-18	10	8
7	07-03-18	2	9
8	08-03-18	0	10
9	09-03-18	0	10
10	10-03-18	12	8
11	11-03-18	5	9
12	12-03-18	1	10
13	13-03-18	8	11
14	14-03-18	7	11
15	15-03-18	0	13
16	16-03-18	0	14
17	17-03-18	0	15
18	18-03-18	10	9
19	19-03-18	3	10
20	20-03-18	0	11
21	21-03-18	0	13
22	22-03-18	0	15
23	23-03-18	0	14
24	24-03-18	5	11
25	25-03-18	7	11
26	26-03-18	3	10
27	27-03-18	0	12
28	28-03-18	0	14
29	29-03-18	2	15
30	30-03-18	1	15
31	31-03-18	6	12

Links de afbeelding van een spreadsheet dat data bevat over de regenval per dag in maart en de bijbehorende temperaturen.
We gaan de data importeren in lists.
Daarvoor moeten we het spreadsheet bestand exporteren als csv bestand.
Bijvoorbeeld als: regenval_maart.csv.
Het csv bestand moet in zelfde map worden opgeslagen als het programma waarmee we het gaan importeren in lists.
We gebruiken voor de import de csv.reader functie. We kunnen de data importeren in rijen of kolommen.
Op de volgende dia een programma dat in kolommen importeert.

Grafieken met Python deel 1.

Programma om een csv bestand koloms gewijs te importeren in lists.

csv_import.py | csv_import_row.py *

```
#csv_import.py
# een van een spreadsheat afgeleid csv bestand
# met data over regenval en temperatuur wordt
# geimporteerd in lijsten voor verder verwerking
# in matplotlib om grafiek te maken.
datum = []
regenval = []
temperatuur = []
import csv
with open("regenval_maart2.csv") as csvfile:
    regen = csv.reader(csvfile, delimiter = ",")
    for column in regen:
        datum.append(column [0])
        regenval.append(column [1])
        temperatuur.append(column [2])

print (datum)
print()
print(regenval)
print()
print(temperatuur)
```

Grafieken met Python deel 1.

CSV import rij voor rij.

csv_import.py × csv_import_row.py * ×

```
1   #csv_import.py
2   # een van een spreadsheat afgeleid csv bestand
3   # met data over regenval en temperatuur wordt
4   # geimporteerd en rij gewijs in lijst weer geplaatst.
5
6   weer = [] # creëer een lege lijst weer.
7   import csv # importeer de python csv module.
8   with open("regenval_maart2.csv") as csvfile:
9       regen = csv.reader(csvfile, delimiter = ",")
10      for row in regen:
11          weer.append(row)
12
13  print (weer)
```

```
[['Datum', 'Regenval', 'Temp'], ['01-03-2018', '0', '7'], ['02-03-2018', '0', '6'], ['03-03-2018', '2', '4'], ['04-03-2018
', '7', '7'], ['05-03-2018', '10', '8'], ['07-03-2018', '2', '9'], ['08-03-2018', '0', '10'], ['09-03-2018', '0', '10'], [
'10-03-2018', '12', '8'], ['11-03-2018', '5', '9'], ['12-03-2018', '1', '10'], ['13-03-2018', '8', '11'], ['14-03-2018', '
7', '11'], ['15-03-2018', '0', '13'], ['16-03-2018', '0', '14'], ['17-03-2018', '0', '15'], ['18-03-2018', '10', '9'], ['1
9-03-2018', '3', '10'], ['20-03-2018', '0', '11'], ['21-03-2018', '0', '13'], ['22-03-2018', '0', '15'], ['23-03-2018', '0
', '14'], ['24-03-2018', '5', '11'], ['25-03-2018', '7', '11'], ['26-03-2018', '3', '10'], ['27-03-2018', '0', '12'], ['28
-03-2018', '0', '14'], ['29-03-2018', '2', '15'], ['30-03-2018', '1', '15'], ['31-03-2018', '6', '12']]
```

Grafieken met Python deel 1.

Soms moet de data in een list gecorrigeerd worden voor bijzondere omstandigheden.
Er kan sprake zijn van een afwijking in een sensor of er moet geconverteerd worden naar een andere eenheid dan de sensor geeft.

Met data in een list is dat niet zo eenvoudig te doen.
Zie voorbeeld.

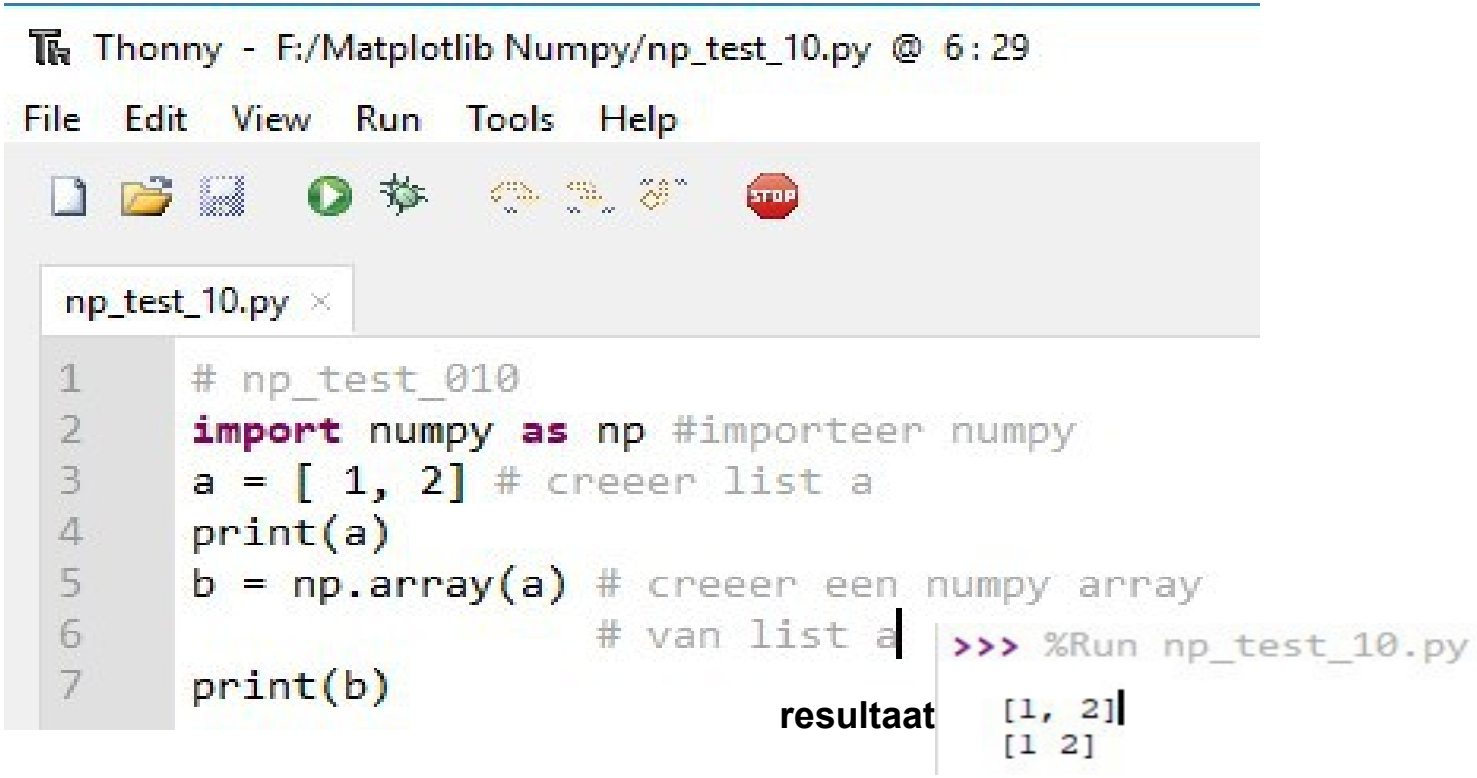
```
# convert_list
a = [1, 2, 3, 4, 5,]
b = []
for x in a:
    x = x * 2
    b.append(x)
print (a)
print (b)
```

De uitkomst is: [1, 2, 3, 4, 5]
[2, 4, 6, 8, 10]

Met numpy gaat dat veel eenvoudiger.

Grafieken met Python deel 1.

Numpy is een mathematische module voor python en het werkt met arrays en matrices.
Een array kan uit enkele maar ook vele duizenden data elementen bestaan.
Een voorbeeld:



Grafieken met Python deel 1.

Vermenigvuldiging van een numpy array.

```
1   # vermenigvuldiging van elementen in
2   # een numpy array
3   import numpy as np
4   a = [1, 2, 3, 4] # list a
5   b = np.array(a) # creeer een numpy aaray van list a
6   print( a)
7   print()
8   print (b)
9   print()
10  b = b * 3 # vermenigvuldig numpy array met drie
11  print( b) # print resultaat
```

```
>>> %Run numpy_product.py
    [1, 2, 3, 4]

    [1 2 3 4]

    [ 3  6  9 12]
```

Grafieken met Python deel 1.

Een list met maand dagen maken.

maand_dagen.py

```
1  # maand_dagen.py
2  maart = list(range(1,32)) # genereert een lijst met dagnummers
3  print (maart)             # van de maand maart
```

Je kunt ook de stap grootte aanpassen.

maand_dagen_2.py

```
1  #maand_dagen_2.py
2  maart = list(range(1,32,2)) # stap grootte is 2
3  print (maart)
```

Grafieken met Python deel 1.

Een numpy array genereren en bewerken.

numpy_test_01.py *

```
1  #test numpy
2  import numpy as np
3  x = np.arange(1, 51)
4  print(x)
5
```

De inhoud van het array bewerken.

numpy_test_03.py

```
1  #test numpy
2  import numpy as np
3  y = [] # creeer een lege list
4  x = np.arange(1., 51.) # genereer een enkelvoudig array
5  y.append (x**2) # kwadrateer het array en plaats het in y
6  print(x) # print het numpy array
7  print(y) # print de lijst met kwadraten
8
```

Grafieken met Python deel 1.

Mathematische bewerkingen met Numpy.

np.sqrt(array_naam) # trekt de wortel uit de elementen in het array.

np.exp(array_naam) # machts verheffen de elementen in het array.

np.sin(array_naam) # berekent de sinus van de elementen in array.

np.cos(array_naam) # berekent de cosinus van de elementen.

np.sum(array_naam) # berekent de som van alle elementen.

np.std(array_naam) # berekent de standaard deviatie van alle elementen

Gedetailleerde informatie over Numpy kun je vinden in de Numpy Reference

https://docs.scipy.org/doc/numpy-1.15.1/reference/

Grafieken met Python deel 1.

'''
Met lineare algebra kunnen we een stelsel van vergelijkingen met
een x aantal onbekenden oplossen.
We proberen daarbij één voor én de onbekenden te elimineren.
Het oplossen van een vergelijking met 4 onbekenden kan al gauw
5 tot 15 minuten in beslag nemen. Numpy doet dat in milliseconden.

Gegeven het volgende stelsel van vergelijkingen:

2w + x +y + 2z =16

3w - x + y + z = 9

5w - 2x – y + 2z =27

4w - 3x + y + z = 8

Als je de vergelijking op de klassieke manier oplost zul je de
volgende waarden voor respectievelijk w, x, y, en z vinden 3, 2, -4, en 6.

De coefficienten van de vergelijkingen plaatsen we in numpy array a.
De constanten komen in numpy array b.

'''

Grafieken met Python deel 1.

Met python numpy kun je het stelsel van vergelijkingen oplossen zoals in onderstaand script is aangegeven. De coëfficienten van resp. w, x, y en z zijn weergegeven in np array a. De constanten achter het = teken vind je array b.

```
import numpy as np
a = np.array ([[2, 1, 1, 2], [3, -1, 1, 1], [5, -2, -1, 2] ,[4, -3, 1, 1]])
b = np.array ([[16], [9], [27],[8]])
print(" Array a bevat de coefficienten: ", a)
print()
print(" Array b bevat de constanten: \n",  b)
s = np.linalg.solve(a,b) # numpy berekent de uitkomsten
print(" Array s bevat de oplossing: ", s)
print()
s.astype(np.int64) # convert floats naar int64 datatype
print()
print(" De oplossing voor w is: ", s[0])
print(" De oplossing voor x is: ", s [1])
print(" De oplossing voor y is: ", s[2])
print(" De oplossing voor z is: ", s[3])
```

Op de volgende dia de oplossing.

Grafieken met Python deel 1.

```
> %Run np_linalg_02.py
 Array a bevat de coefficienten:  [[ 2  1  1  2]
 [ 3 -1  1  1]
 [ 5 -2 -1  2]
 [ 4 -3  1  1]]

 Array b bevat de constanten:
 [[16]
 [ 9]
 [27]
 [ 8]]
 Array s bevat de oplossing:  [[ 3.]
 [ 2.]
 [-4.]
 [ 6.]]


 De oplossing voor w is:  [3.]
 De oplossing voor x is:  [2.]
 De oplossing voor y is:  [-4.]
 De oplossing voor z is:  [6.]
```

Grafieken met Python deel 1.

In de volgende workshop gaan we de kennis die we vandaag hebben opgedaan toepassen.

Met behulp van de module matplotlib gaan we grafieken maken van verzamelde data.

Kijk de workshop over het meten van temperaturen nog eens na.

Zorg er voor dat matplotlib en numpy geïnstalleerd zijn op je Raspberry Pi.

De workshop kan ook gevolgd worden op je PC.
Je moet dan op een andere manier voor data zorgen.
Ook hier moet matplotlib en numpy geinstalleerd zijn.

Vragen?