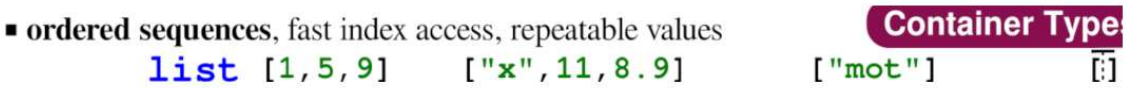
Labo 5

Computationeel Denken

Inleiding: lijsten versus NumPy arrays

Voor het werken met vectoren en data zijn de standaard Python arrays of lijsten niet altijd het meest aangewezen datatype.

- Lijsten zijn dynamisch: het aantal elementen ligt niet vast.
- Ze kunnen ook verschillende types waarden bevatten,
  zie ook figuur 1 (Bron: Python Cheatsheet).



Figuur 1

Zo stemmen operaties op lijsten niet overeen met hoe berekeningen met vectoren gebeuren, zie daarvoor onderstaand voorbeeld :

[4,5] * 3 = [4,5,4,5,4,5]

[4,5] + 3 = error

In deze tekst verwijzen we met 'primitieve matrices' naar matrices die bestaan uit lijsten.

Een NumPy array daarentegen is een geordende rij van variabelen van hetzelfde type. Dit maakt deze datastructuur geschikt voor vectoren en matrices.

Numerical Python, beter bekend als NumPy, is een package met uitgebreide functionaliteiten voor het werken met (onder andere) vectoren en matrices.

Veel takken van de wetenschap maken er dankbaar gebruik van voor numeriek rekenwerk.

Labo 5 Computationeel Denken

Oefeningen Lijsten

1. Bereken het gemiddelde van een lijst van een willekeurig aantal getallen.
   Je geeft getallen in en stopt met 'Enter'.

   Zie Labo 2 – Oefening 6

   Maak gebruik van volgende functies :

   inlezen_lijst ( ), bepaal_gemiddelde (lijst) en toon_lijst (lijst).

2. Maak een lijst aan voor x-waarden horende bij een open interval
   [x_0, …, x_n [bijvoorbeeld: x_0=2 , x_n=6

   - Vul de lijst op met equidistante punten, d.w.z. dat de afstand tussen twee punten x_i en x_{i+1} steeds gelijk dient te zijn.

   Bereken de afstand tussen twee punten door het verschil van de boven- en ondergrens van het interval te delen door een variabele opdeling. Voorbeeld opdeling = 10

   - Rond de waarden af tot op één cijfer na de komma.

   - Maak ook gebruik van de functie toon_lijst ( ) .

   Voorbeeld van uitvoer met ondergrens = 2, bovengrens = 6 en opdeling = 10 :

   2 2.4 2.8 3.2 3.6 4.0 4.4 4.8 5.2 5.6

3. Uitbreiding vorige oefening.

   - Vul de code aan met een methode bereken_functiewaarde (x) die voor een ontvangen waarde een berekende waarde teruggeeft op basis van het voorschrift :

   $f(x) = \sqrt{x} - 1$

- Maak een lus waarin je de lijst met x-waarden doorloopt en ieder element ervan meegeeft aan de methode.

- Sla alle terugkeerwaarden (afgerond op één cijfer na de komma) op in een lijst met y-waarden.

- Toon vervolgens de lijst met x-waarden op een eerste lijn en de lijst met y-waarden netjes geschikt op de lijn eronder. Zorg dus dat de x-waarden net boven de bijhorende y-waarden komen te staan.

Maak gebruik van functie toon_lijst ( ).

Voorbeeld van uitvoer :

2	2.4	2.8	3.2	3.6	4.0	4.4	4.8	5.2	5.6
0.4	0.5	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4

4. Plot van een sinusfunctie.

- Met behulp van de matplotlib-bibliotheek kunnen we functies plotten. De plotfunctie ontvangt arrays voor x- en bijhorende y-waarden.

- De x-array bevat alle elementen uit het domein waarvan we een beeldwaarde willen berekenen. Deze array dient gevuld te worden met een partitie van het interval. Dit houdt in dat de afstand tussen twee opeenvolgende elementen op de x-as steeds gelijk dient te zijn, zoals in de voorgaande oefeningen.

- Inspireer je op de voorgaande oefeningen om de arrays op te vullen met de geschikte waarden.

Plot de sinusfunctie over het interval $[-4\pi , 4\pi]$ met waarde van de opdeling = 100.

- De grafiek kan worden getekend door je code te plaatsen tussen de import-statements en de commando's voor de plotbibliotheek op de laatste twee regels:

```
import matplotlib.pyplot as plt
import math
…
…

plt.plot( x_array , y_array )
plt.show()
```

Oefeningen Numpy arrays voor matrices

5. Schrijf een programma met een functie genereer_matrix ( ) om een Numpy matrix met random gehele getallen uit het gesloten interval [0,9] te genereren.

   Maak hiervoor gebruik van de functie randint ( ) uit de bibliotheek random.

   Vraag aan de gebruiker het gewenst aantal rijen en kolommen. Schrijf een functie toon_matrix ( ) om de matrix op het scherm te tonen.

6. Vul het programma aan met een functie transponeer_matrix ( ) die de matrix transponeert.

   Doe dit zonder gebruik van het ingebouwde attribuut ‘T’ van een Numpy-array.

   Toon de getransponeerde matrix op het scherm.

   Controleer nu eens je resultaat aan de hand van het attribuut ‘T’.

7. Vul het programma aan met een menu :

   Vraag aan de gebruiker of hij de matrix, nadat die automatisch gegenereerd is,
   (1) wilt transponeren
   (2) horizontaal wil spiegelen.

   Maak gebruik van een functie spiegel_matrix_horizontaal ( ).
   Toon de gespiegelde matrix op het scherm.

   Het menu kan er als volgt uitzien :

```
def toon_opties () :

    keuze = ""
    geldige_keuzes = [ "T" , "H" , "S" ]

    while keuze not in geldige_keuzes :

        keuze = input (  "maak uw keuze : \n"
                         "T \t ( transponeren ) \n"
                         "H \t ( voor horizontaal spiegelen ) \n"
                         "S \t ( stop ) "
                         )
        if keuze not in geldige_keuzes :
            print("ongeldige invoer")

    return keuze
```

   Voorbeeld van uitvoer :

```
3       4       6       8       9
6       6       6       3       4
3       7       9       0       2

9       8       6       4       3
4       3       6       6       6
2       0       9       7       3
```

8. Vul het programma aan met een derde optie (3) voor het 'roteren' van een matrix, met als 'rotatiespil' het element in de eerste kolom, op de laatste rij.

   Maak gebruik van een functie roteer_matrix ( ).

   Toon de geroteerde matrix op het scherm.

   Voorbeeld van uitvoer :

```
3    9    0    5    2
3    7    8    0    8
2    8    3    8    5
```

```
2    3    3
8    7    9
3    8    0
8    0    5
5    8    2
```