Opdrachten Sessie 14 (individueel)

Deze opdrachten worden individueel gemaakt en ingeleverd op Canvas. Bij het maken van de opdrachten is het toegestaan de documentatie behorende bij het vak en eerder gemaakte opdrachten te gebruiken. **Het is NIET toegestaan internetbronnen te raadplegen**. Elke opdracht levert één `.py` file op die ingeleverd wordt in Canvas. Er zijn in totaal 7 opdrachten. Het totaal aantal te behalen punten is 100. Je moet minimaal 60 punten scoren voor een ~~VOLDAAN.~~ voldoende.

---

1 (5p) lever in als: `opdracht1.py`

Zoek het meest geschikte Python type bij de data/info. Gebruik elk type slechts één keer. Type je antwoord als comment in de .py file.

	data/info		type
A	filenaam	1	boolean
B	jaartal	2	string
C	gemeten druk	3	integer
D	meetserie van 10 afstanden	4	float
E	wel of geen valide meting	5	numpy array

2 (10p) lever in als: `opdracht2.py`

a) Definieer twee 1D numpy arrays met 100 uniform verdeelde random getallen tussen -1 en 1. Gebruik de namen `x` en `y` voor deze array's.

b) Combineer beide arrays tot één 2-dimensionale numpy array `p` waarbij de waarden van array `x` de eerste kolom is en de waarden van `y` de tweede kolom.

Elke rij in array `p` bestaat nu uit twee getallen. We beschouwen dit als een punt in het x-y vlak. De afstand van het punt tot aan de oorsprong is gegeven door:

$$s = \sqrt{x^2 + y^2}$$

c) Creeër een array `cirkel` die alle rijen van array `p` bevatten waarvoor geldt dat de afstand s van het punt tot de oorsprong groter dan of gelijk aan 0.5 en kleiner dan of gelijk aan 1.

3 (10p) lever in als: `opdracht3.py`

a) Gebruik list comprehension om een list te definieren die de eerste 10 positieve integers (1, 2, 3, ..., 10) bevat.

b) Gebruik list comprehension om een list te definieren die bestaat uit de eerste 10 positieve integers die deelbaar zijn door 3 dus (3, 6, 9, etc).

c) Gegeven een lijst met filenamen `['jan21_meting1.csv', 'jan23_meting1.csv', 'test1.csv', 'jan23_meting2.dat', 'feb23_meting4.csv', 'test4.csv', 'jan23_meting.dat', 'meting3.csv']` , gebruik een list comprehension om een list te definieren die alleen de filenamen bevat die beginnen met `jan23`. Op canvas vind je een .py file met daarin de definitie van de list met filenamen.

4 (15p) lever in als: `opdracht4.py`

Op canvas staat een csv bestand `minmaxtemperatuur.csv` met daarin de minimale en maximale temperatuur per dag over het jaar 2020 zoals gemeten in het meetstation Twente. De file bestaat uit twee kolommen met getallen. De eerste rij is dag 1 van het jaar, de tweede rij is dag 2, etc. De eerste kolom is de minimale temperatuur die gemeten is op die dag en de tweede kolom is de maximale temperatuur die op die dag gemeten is. De temperaturen zijn in graden Celsius. In deze opdracht maak je een plot van de gegevens in de file.
Schrijf een script die:

a) de data van de file inleest in twee numpy arrays `mintemp` en `maxtemp`.

b) een lijnplot maakt met op de x-as de tijd (in dagen) en op de y-as de minimale (blauw) en maximale (rood) temperatuur.

c) een geschikte legend aan de plot toevoegt.

d) geschikte as-labels aan de figuur toevoegt.

5 (20p) lever in als: `opdracht5.py`

We gebruiken hetzelfde csv bestand `minmaxtemperatuur.csv` zoals gebruikt in opdracht 4. Schrijf een script die bepaalt op welke dagen er sprake was van een warmtegolf. Er is sprake van een warmtegolf indien de maximale temperatuur minimaal 5 dagen achtereen 25 graden Celsius of hoger is [8]. Tussen twee verschillende warmtegolven zit altijd minimaal één dag waarbij de maximale temperatuur lager is dan 25 graden Celsius. Het script moet alle! warmtegolven van 2020 printen naar de console waarbij zowel de eerste dag, de laatste dag en de duur van de warmtegolf wordt weergegeven.

6 (20p) lever in als: `opdracht6.py`

Gegeven zijn de positie (r), massa (m) en snelheid (v) van een aantal puntmassa's. Gevraagd wordt een script te schrijven die het impulsmoment L van het systeem bepaalt (ten opzichte van de oorsprong). Het impulsmoment is gegeven door:

$$\vec{L} = \sum_i \vec{r}_i \times \vec{p}_i$$

, met $\vec{p}$ de impuls ($= m\vec{v}$) en waarbij de som over alle puntmassa's is. Ter info, het kruisproduct tussen twee vectoren $\vec{a}$ en $\vec{b}$ is gelijk aan

$$\vec{a} \times \vec{b} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}$$

Op canvas staat een begin van dit script die de data (r, m en v) van de puntmassa's bevat.

[8] ter info, dit is een wat simpelere definitie dan die in werkelijkheid gehanteerd wordt.

7 (20p) lever in als: `opdracht7.py`

Schijf een *functie* `bubblesort(x)` die de list `x` sorteert van klein naar groot door middel van het bubblesort algoritme. Het bubblesort algoritme werkt als volgt:

1. neem de eertse twee elementen van de lijst `x`
2. ruil deze om indien ze niet op juiste volgorde staan
3. neem vervolgens het tweede en derde element van de lijst
4. ruil die om als ze niet op volgorde staan
5. herhaal dit voor het derde en vierde element, vierde en vijfde element etc. tot het einde van de lijst.
6. Na het uitvoeren van stap 1 t/m 5 controleer je of je één of meerdere keren twee elementen hebt moeten omruilen. Indien dat het geval is begin je weer bij stap 1. Zoniet, dan is de lijst gesorteerd en ben je klaar.

Ter verduidelijking, hieronder de procedure aan de hand van een voorbeeld (van wikipedia):
Neem een lijst met getallen [5, 1, 4, 2, 8], en sorteer van laag naar hoog met behulp van het bubblesort algoritme. In elke stap, elementen die vergeleken worden zijn vet (bold) gedrukt. In dit voorbeeld worden drie maal de stappen 1 t/m 5 uitgevoerd (in het voorbeeld worden dit "passes" genoemd) voordat de lijst gesorteerd is.

Eerste "Pass"
(**5** **1** 4 2 8) → (**1** **5** 4 2 8), Vergelijk de eerste twee elementen en ruil om omdat $5 > 1$.
(1 **5** **4** 2 8) → (1 **4** **5** 2 8), ruil om omdat $5 > 4$
(1 4 **5** **2** 8) → (1 4 **2** **5** 8), ruil om omdat $5 > 2$
(1 4 2 **5** **8**) → (1 4 2 **5** **8**), ruil niet om, omdat $5 < 8$

Tweede "Pass"
(**1** **4** 2 5 8) → (**1** **4** 2 5 8)
(1 **4** **2** 5 8) → (1 **2** **4** 5 8), ruil om omdat $4 > 2$
(1 2 **4** **5** 8) → (1 2 **4** **5** 8)
(1 2 4 **5** **8**) → (1 2 4 **5** **8**)

De lijst is nu gesorteerd, maar het algoritme weet dit nog niet. Het algoritme heeft nog één extra "pass" nodig zonder dat er elementen geruild zijn om te weten dat de array gesorteerd is.

Derde "Pass"
(**1** **2** 4 5 8) → (**1** **2** 4 5 8)
(1 **2** **4** 5 8) → (1 **2** **4** 5 8)
(1 2 **4** **5** 8) → (1 2 **4** **5** 8)
(1 2 4 **5** **8**) → (1 2 4 **5** **8**)