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.

---

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	aantal gemeten deeltjes per seconde	1	boolean
B	gemeten spanning	2	string
C	massa's van de planeten in ons zonnestelsel	3	integer
D	naam van planeet	4	float
E	voldoende of onvoldoende	5	numpy array

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

a) Definieer een 2D numpy array `x` (100 rijen en 200 kolommen) met 20000 normaal verdeelde random getallen met gemiddelde 0 en standaard deviatie 1.

b) Bereken een nieuwe numpy array `y` gelijk aan de som van de kolommen van `x` (dus kolom 0 + kolom 1 + kolom 2 + ... + kolom 199). De array `y` heeft dus 100 elementen.

c) Definieer 2 nieuwe numpy arrays `y1` en `y2` waarbij `y1` bestaat uit de eerste 50 elementen van `y` en `y2` bestaat uit de tweede 50 elementen van `y`

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

a) Gebruik list comprehension om een list te definiëren die de kwadraten van de eerste 25 positieve integers (1, 4, 9, 16, ..., 625) bevat.

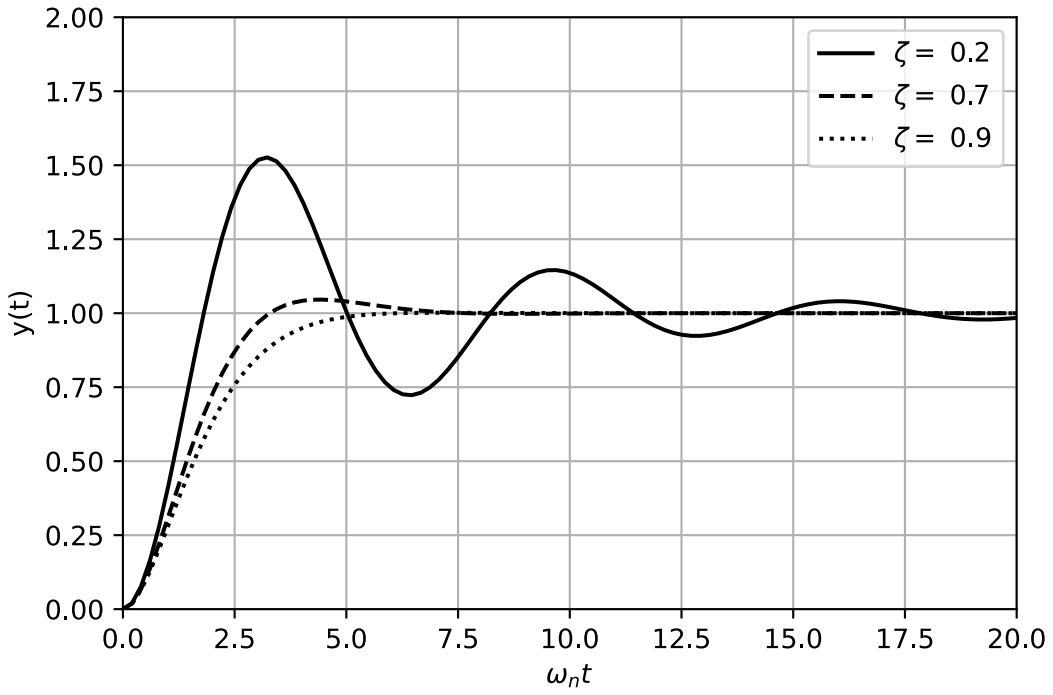
b) Gebruik list comprehension om een list te definiëren die waarden van de list uit a) bevat die groter zijn dan 200 en kleiner dan 400.

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

De step response van een 2-de orde systeem is gegeven door:

$$y(t) = 1 - e^{-\zeta\omega_n t}\left[\cos\left(\omega_n t\sqrt{1-\zeta^2}\right) + \frac{\zeta}{\sqrt{1-\zeta^2}}\sin\left(\omega_n t\sqrt{1-\zeta^2}\right)\right]$$

In onderstaande plot is de step response geplot als functie van $\omega_n t$ voor een aantal verschillende waarden voor de demping ζ. Schrijf een script die de onderstaande plot reproduceert. De plot moet dezelfde kleuren, lijnstijl, legend, as-labels en as-limieten hebben [9].



[9] indien je niet weet hoe je Griekse letters gebruikt mag je ook *gewone* letters gebruiken in de legend en in de as-labels

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

De *taxicab distance* s tussen twee punten $p_1 = (x_1, y_1)$ en $p_2 = (x_2, y_2)$ in een vlak is gegeven door:

$$s = |x_2 - x_1| + |y_2 - y_1|$$

Deze afstand wordt ook wel de Manhatten distance genoemd vanwege de gridlayout van de straten in Manhatten. De afstand die je moet afleggen tussen twee locaties in Manhatten is gelijk aan bovenstaande definitie. Zie ook onderstaande figuur.

Figure 1: Illustratie van de taxicab distance versus de Euclidische afstand. Ref: http://calculus123.com/wiki/Topological_spaces

a) Schrijf een functie `taxicab(p1, p2)` die gegeven twee punten `p1` en `p2` (beide gedefinieerd als 1D numpy arrays van lengte 2) de taxicab distance berekent [10]. Voorbeeld:

IPython Console

```
In [1]: p1 = np.array([-2, 1])
In [2]: p2 = np.array([3, 4])
In [3]: taxicab(p1, p2)
Out[3]: 8
```

Op canvas staat een CSV bestand met een lijst van punten (als twee kolommen, de eerste kolom is de x-coördinaat. de tweede kolom is de y-coördinaat). De lijst van punten beschrijft een route. We willen de taxicab afstand bepalen van de totale route (dus van het eerste punt tot het laatste punt via alle tussen gelegen punten). Schrijf een script die:

b) de data van de file inleest in één 2D numpy array `route`.

c) de totale taxicab afstand van de route berekent. Maak hierbij gebruik van de eerder gemaakte functie.

[10] Het is niet toegestaan een kant-en-klare scipy functie te gebruiken

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

De Madelung constante α van NaCl kan worden berekend met een reeks:

$$\alpha = 6 - \frac{12}{\sqrt{2}} + \frac{8}{\sqrt{3}} - ... + ... = 1.7476...$$

Deze reeks convergeert echter slecht. Een sneller convergerende reeks is de volgende:

$$\alpha = 12\pi \sum_{\substack{m \geq 1 \\ m \text{ odd}}} \sum_{\substack{n \geq 1 \\ n \text{ odd}}} \cosh^{-2}\left[\frac{\pi}{2}\sqrt{(m^2+n^2)}\right]$$

De sommatie over m en n is dus voor alle oneven integer waarden groter gelijk aan 1 Bereken de Madelung constante met behulp van bovenstaande reeks voor m en n beide tot en met 13.

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

Gegeven een list met getallen. Schrijf een functie `lengthlongestincreasing(x)` die bepaalt wat de lengte is van de langste toenemende reeks opeenvolgende getallen in de list `x`.
Bijvoorbeeld:

IPython Console

```
In [1]: x = [23, 2, 3, 4, 88, -1, 2, 3]
In [2]: lengthlongestincreasing(x)
Out[2]: 4
```

Het antwoord is hier 4 omdat de langste toenemende reeks opeenvolgende getallen in de list gelijk is aan `[2, 3, 4, 88]` en de lengte daarvan is 4.