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`

Definieer een 2D numpy array `z` met 51 rijen en 2 kolommen waarbij:

- de eerste kolom bestaat uit alle even getallen tussen -50 en 50 en inclusief 0.
- de tweede kolom bestaat uit de bijbehorende kwadraten.

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

Gegeven is de volgende numpy array:

```
z = np.array([[1, 7, 3, 9, 5], [6, 3, 8, 4, 10]])
```

Schrijf een script die:

a) door middel van slicing van `z` de array `w` definieert:

```
array([[3, 9],
       [8, 4]])
```

b) door middel van 1 statement uit array `z` een array `q` definieert met alle elementen van `z` die groter zijn dan 5:

```
array([ 7,  9,  6,  8, 10])
```

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

The golden ratio is een getal met als waarde 1.618033... . Deze ratio kan worden benaderd met de volgende recursieve relatie:

$$f(n) = 1 \qquad \text{if } n = 0$$
$$f(n) = 1 + 1/f(n-1) \qquad \text{if } n > 0$$

Hoe groter de waarde van n, hoe beter $f(n)$ de golden ratio benaderd.

a) Schrijf een *recursieve* functie `golden_ratio_rec(n)` die de golden ratio benadert gegeven de waarde van n.

b) Schrijf een **NIET** *recursieve* functie `golden_ratio(n)` die de golden ratio benadert gegeven de waarde van n.

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

Gebruik voor deze opdracht de file `meting_134.csv` (zie canvas). De file `meting_134.csv` bestaat uit drie kolommen met getallen. De eerste kolom is de tijd t (eenheid seconden). De tweede kolom is het vermogen P (eenheid Watt) van meting 1; de derde kolom is het vermogen van meting 2. 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 drie numpy arrays `t`, `P1` en `P2`.

b) een lijnplot maakt met op de x-as de tijd en op de y-as het vermogen van de 2 metingen. De kleuren zijn zwart voor $P1$ en rood voor $P2$.

c) een geschikte legend aan de plot toevoegt.

d) geschikte as-labels aan de figuur toevoegt.

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

Schrijf een functie `find_sequence(test_list, sub_list)` die gegeven een list `test_list` bepaalt of die list een bepaalde sublist `sub_list` bevat. Als dat zo is geeft de functie `True` en anders `False`. Dus bijvoorbeeld:

IPython Console

```
In [1]: test_list = [5, 6, 3, 8, 2, 1, 7, 1]
In [2]: sub_list = [8, 2, 3]
In [3]: find_sequence(test_list, sub_list)
Out[3]: False
```

en

IPython Console

```
In [1]: test_list = [5, 6, 3, 8, 2, 1, 7, 1]
In [2]: sub_list = [8, 2, 1]
In [3]: find_sequence(test_list, sub_list)
Out[3]: True
```

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

Twee woorden zijn anagrammen als je het ene woord kan maken met de letters van het andere woord. Dus bijvoorbeeld: *heart* en *earth* of *kneukel* en *kukelen*. Schrijf een functie `anagram(w1, w2)` die van de twee woorden `w1` en `w2` (beide strings) bepalen of ze anagrammen zijn.Dus bijvoorbeeld:

IPython Console

```
In [1]: w1 = 'heart'
In [2]: w2 = 'earth'
In [3]: anagram(w1, w2)
Out[3]: True

In [4]: w1 = 'O Draconian devil'
In [5]: w2 = 'Leonardo Da Vinci'
In [6]: anagram(w1, w2)
Out[6]: True

In [7]: w1 = 'thermo'
In [8]: w2 = 'python'
In [9]: anagram(w1, w2)
Out[9]: False
```

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

Een magisch vierkant is een vierkant waarbij opeenvolgende getallen 1 t/m N zodanig zijn ingevuld dat de som van elke rij, kolom en diagonaal dezelfde uitkomst heeft. In de figuur is een 4×4 magisch vierkant te zien.

Figure 1: Voorbeeld van een magisch vierkant.

In dit geval is de som van elke rij, diagonaal en kolom gelijk aan 34. Schrijf een functie `magic_square(m)` die gegeven een 2D numpy array `m` controleert of de array een magisch vierkant representeert. De functie geeft `True` of `False` afhankelijk van het resultaat. Bijvoorbeeld:

IPython Console

```
In [1]: m = np.array([[2, 7, 6], [9, 5, 1], [4, 3, 8]])
In [2]: magic_square(m)
Out[2]: True

In [1]: m = np.array([[1, 2], [3, 4]])
In [2]: magic_square(m)
Out[2]: False
```