OEFENINGEN PYTHON – REEKS 6

1. A) Schrijf een functie die een getal *x* en een getal *y* meekrijgt. De functie geeft de uitkomst van volgende bewerking als <u>returnwaarde</u> terug:

$$\begin{cases}(x * y) - x & \text{als} \quad x \geq y \\ (x * y) - y & \text{als} \quad x < y\end{cases}$$

   B) Schrijf een nieuwe functie die een getal *a* als parameter meekrijgt. Deze functie gaat de functie uit vraag 1A oproepen voor alle combinaties (*i*,*a*) waarbij parameter *i* alle waarden $\mathbf{1} \leq \boldsymbol{i} \leq \mathbf{2}\boldsymbol{a}$ krijgt. Het resultaat wordt telkens afgeprint.

   C) Voer de functie uit vraag 1B uit voor de waarde *a* = **5**.

   Bijvoorbeeld: *als je de functie uit vraag 1B zou uitvoeren voor a=3 krijg je volgende waarden:*
   0 3 6 8 10 12

2. Matrices
   A) Maak een nieuwe variabele aan waarin je een matrix in de vorm van een geneste lijst gaat opslaan. Schrijf de nodige code om de waarden uit het bestand 'data.txt' automatisch in te lezen en op gepaste wijze op te slaan. <u>Let op:</u> je code moet blijven werken indien de dimensies van de matrix in 'data.txt' zouden worden gewijzigd!

   <u>Tip:</u> als je vraag 2A echt niet zou kunnen dan mag je de waarden manueel ingeven zodat je toch verder kan met de rest van vraag 2. Uiteraard verlies je hierdoor wel punten.

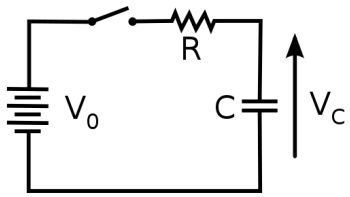
   B) Maak een nieuwe geneste lijst aan die een matrix voorstelt met dezelfde dimensies als de ingelezen matrix uit vraag 2A (schrijf je code zo algemeen mogelijk zodat ze blijft werken indien deze dimensies zouden veranderen). De waarden van deze nieuwe matrix worden berekend door de waarden uit de ingelezen matrix (vraag 2A) over te nemen en te vermenigvuldigen met het rijnummer. De eerste rij heeft als rijnummer 1 (zie voorbeeld).
   Bijvoorbeeld:
   Als de ingelezen matrix in vraag 2A de matrix $\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$ *is dan wordt de nieuwe variabele uit vraag 2B gelijk aan de matrix* $\begin{bmatrix}1 & 2\\ 6 & 8\\ 15 & 18\end{bmatrix}$.

   C) Maak een nieuwe functie waaraan je een matrix (geneste lijst) kunt meegeven. De functie heeft als returnwaarde een lijst die de gemiddelde waarde van elke rij van de meegegeven matrix bevat.

   D) Voer de functie uit vraag 2C uit voor de matrices uit vraag 2A en vraag 2B. Print de verkregen resultaten af.

3. We gaan Python gebruiken om het opladen van een condensator te simuleren. In een eerste stap ga je hiervoor een gepaste functie aanmaken, in een tweede stap ga je deze functie gebruiken om een concreet voorbeeld uit te rekenen.

   Gegeven is volgende stroomkring, met bronvoeding V_0, weerstand R en condensator C. We wensen nu de spanning over de condensator (V_c) en de stroomsterkte in de kring (I) te simuleren.



   De formules hiervoor zijn:

$$V_c = V_0 * \left(1 - e^{\frac{-t}{RC}}\right)$$

$$I = \frac{V_0}{R} * e^{\frac{-t}{RC}}$$

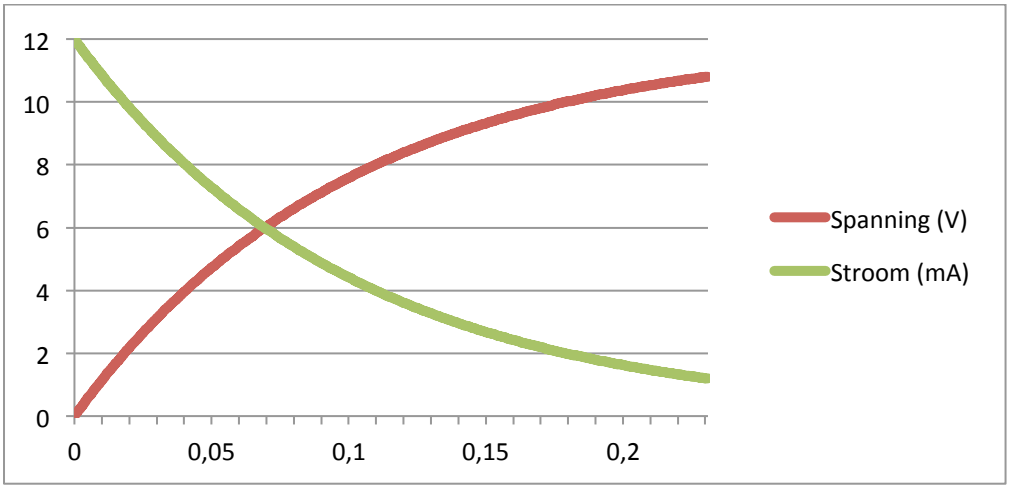
   A) Maak een nieuwe functie die het stroomverloop en het spanningsverloop in deze kring berekent. De functie heeft 5 parameters: V_0, R, C, het stoppercentage en de tijdsstap van de simulatie. De simulatie stopt wanneer V_c groter wordt dan een bepaald percentage van V_0 (dit is dus het stoppercentage dat als parameter wordt meegegeven). Voor elke tijdsstip in de simulatie worden t, I en V_c bijgehouden in een geneste lijst. Deze geneste lijst wordt al resultaat van de functie teruggegeven.

      Tip: om een exponentiele macht te berekenen kan je de exp() functie uit de 'math' bibliotheek gebruiken. Importeer deze bibliotheek dus bovenaan je code en gebruik math.exp(a) om e^a te berekenen.

   B) Gebruik je functie uit vraag 3A nu om V_c en I te simuleren voor een kring met deze parameters: **V_0=12V**, **R=1kΩ** en **C=100µF**. De simulatie stopt als de condensator voor **90%** is opgeladen. Kies als tijdsstap **1ms** of kleiner.

   C) Om de simulatie te kunnen weergeven ga je nu de geneste lijst die werd gegenereerd in vraag 3B wegschrijven naar een CSV bestand. In dit bestand staan op elke lijn de gegevens voor 1 tijdsstip uit de simulatie, gescheiden door een puntkomma.

      Hint: je kan je CSV bestand controleren door het in Excel te importeren (let op het verschil tussen een punt en een komma om decimalen aan te tonen) en een scatterplot te maken (zie hieronder). Let op: dit is NIET verplicht, er staan enkel punten op je code!



4. We programmeren de Zeef van Eratosthenes, een methode om alle priemgetallen tot aan een bepaalde waarde te vinden. De methode werkt als volgt:
    a. Maak een lijst met alle getallen vanaf het getal 2 tot een zelf te kiezen maximum
    b. Kies het kleinste nog niet doorstreepte getal uit de lijst
    c. Streep alle veelvouden van het gekozen getal door (maar niet het getal zelf). Begin het doorstrepen vanaf het kwadraat van het gekozen getal.
    d. Herhaal stappen b en c totdat het kwadraat van het gekozen getal groter is dan het maximum. De getallen die op deze manier overblijven zijn alle priemgetallen tot het maximum.

    Je gaat een functie 'zoekPrimes(*a*)' maken die alle priemgetallen print tussen 2 en meegegeven getal *a*. Deze functie heeft de volgende functionaliteit:

    A) Maak twee nieuwe lijsten aan. De eerste lijst bevat de getallen vanaf 2 tot en met de waarde *a*. De andere lijst is exact even lang en bevat allemaal 1-tjes. Deze tweede lijst gebruiken we om een getal uit de eerste lijst te kunnen 'schrappen' door middel van de 1 die overeenkomt met het desbetreffende getal (dus op dezelfde index in de lijst) in een 0 te veranderen.

    B) Schrijf de nodige code die stappen b en c uit de Zeef van Eratosthenes blijft uitvoeren (zie boven). Laat het programma automatisch stoppen wanneer het kwadraat van het gekozen getal groter is dan de waarde *a*.

    C) Print alle nog niet doorstreepte waarden af

    ⇨ Test je functie door alle priemgetallen kleiner dan of gelijk aan **41** te printen!

5. We gaan een programma schrijven dat de afbetaling van een lening simuleert. Stel dat we **TOT** euro willen lenen. De rente bedraagt **r%** en we kunnen maximaal **MAX** euro per jaar terugbetalen.
   Elk jaar zullen we een intrest (**IN**) moeten betalen: **r%** van het nog terug te betalen bedrag. Het eerste jaar bedraagt de intrest dus **r%** van **TOT**. Aangezien we per jaar **MAX** euro afbetalen, kunnen we jaarlijks buiten de intrest **IN** nog (**MAX**-**IN**) kapitaal afdragen. Na het eerste jaar zijn we de bank dus nog **TOT**-(**MAX**-**IN**) euro verschuldigd. Op dit restbedrag zullen we het volgende jaar dan weer intrest moeten betalen. Aangezien het restbedrag jaarlijks kleiner wordt, wordt de betaalde intrest ook jaarlijks kleiner en wordt het kapitaal dat we jaarlijks afdragen groter.

Voorbeeld: je wil een lening aangaan van 100000. Je kan maximum 7000 euro per jaar terug betalen. De huidige rentevoet is 3%.
Het eerste jaar betalen we 100000*0.03 = 3000 euro intrest. We kunnen dus 7000-3000=4000 euro kapitaal aflossen en in totaal moeten we nog 100000-4000=96000 euro aflossen.
Het tweede jaar betalen we 96000*0.03=2880 euro intrest, 7000-2880=4120 euro kapitaal en rest er ons nog 96000-4210=91880 euro af te lossen

   a. Maak een functie 'mainLening' en definieer in deze functie enkele variabelen waarin je het bedrag van de lening, de rentevoet en het totale jaarlijks bedrag dat je kan terugbetalen kunt bijhouden. Schrijf later alle functie-aanroepen van deze oefening in deze hoofdfunctie!

   b. Maak een nieuwe functie die aan de hand van een opgegeven leenbedrag, een rente en een maximale jaarlijkse afbetaling een afbetalingsplan berekent. Een afbetalingsplan is een geneste lijst, waarvoor elke sublijst één jaar voorstelt met deze informatie:
      [interest kapitaal restbedrag]
      Elke sublijst bevat dus de intrest en het kapitaal die dat jaar werden betaald, en het nog resterende af te betalen bedrag na dat bepaalde jaar.

   c. Maak een functie waaraan je een afbetalingsplan kunt meegeven en die de informatie mooi afdrukt, bvb zo:

```
Geleend bedrag: 100000
Huidige rente: 0.03
Jaar  1 | 4000.00 | 3000.00 | 96000.00
Jaar  2 | 4120.00 | 2880.00 | 91880.00
…
```

      De lijnen kunnen geprint worden met een speciale formatteer expressie:
      print *'Jaar {0:2d} | {1:7.2f} | {2:7.2f} | {3:8.2f}'.format(jaar, kapitaal,interest,restbedrag)*

      Let op! Het onderstreepte deel kunnen jullie exact overnemen. De parameters die de *format* functie neemt zijn variabel naargelang het jaar.

   d. Roep in je hoofdfunctie (zie puntje 3a) de functie uit 3b op voor (TOT=100000, r=3%, MAX=7000) en sla het afbetalingsplan op in een nieuwe variabele. Print deze variabele mooi af door hem mee te geven aan de functie uit 3c.

   e. Indien je slechts 3000 euro had om je afbetaling te doen, wat gebeurt er dan? Print een boodschap af indien dit gebeurt, en geef een lege afbetalingslijst terug.

6. We programmeren de torens van Hanoi, een algoritme waarbij een reeks schijven van één stok naar een andere stok worden verplaatst door gebruik te maken van een derde hulpstok. De schijven hebben een verschillende diameter en dienen steeds van groot naar klein georderdend te zijn. In de onderstaande tabel wordt het principe uitgelegd om 3 schijven van stok A naar stok B te brengen met stok C als hulpstok.

	Stok A	Stok B	Stok C
Begin	3,2,1		
Stap 1	3,2	1	
Stap 2	3	1	2
Stap 3	3		2,1
Stap 4		3	2,1
Stap 5	1	3	2
Stap 6	1	3,2	
Stap 7		3,2,1	

A) Programmeer een recursieve functie **hanoi(bron, bestemming, hulp, aantal)** die 3 lijsten meekrijgt en het aantal te verplaatsen schijven van bronlijst naar bestemmingslijst door gebruik te maken van de hulplijst.

Tip: vanaf stap 1 tot stap 3 worden de 2 bovenste schijven van stok A naar stok C gebracht. Vanaf stap 5 worden diezelfde schijven van stok C naar stok B gebracht. Baseer je hierop voor de recursieve functieoproepen.

Tip: de omgekeerde bewerking van de lijst methode append(waarde) is pop() en haalt de laatste waarde van de lijst af.

B) Test de functie door de torens van hanoi op te lossen voor 5 schijven. Definieer stok A, B en C buiten de recursieve functie en print bij elke stap de inhoud van deze stokken.

C) Breid de functie uit zodat het totale aantal stappen ook wordt bijgehouden en geretourneerd. Bereken hoeveel stappen nodig zijn voor het oplossen met 10 schijven.