Opdracht 1: $\pi \approx \cdots$

Leibniz was de eerste die in 1674 een reeksontwikkelingen als benadering van π formuleerde:

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

a. Definieer een functie die de term van de reeks bepaalt in functie van n.

b. Schrijf een programma dat π benadert d.m.v. de eerste 500000 termen van de reeks.

Opdracht 2: Hoeveel nullen?

De onderstaande functie telt het aantal nullen dat voorkomt in een getal.
Het str()-statement zet een getal om in een string: b.v. str(25) resulteert in "25" en str(2.5) in "2.5".

```
def aantal_nullen(a):
    cijfers=str(a)
    aantal=0
    for c in cijfers:
        if c == "0":
            aantal += 1
    return aantal
```

a. Bereken met deze functie het aantal nullen in 100! .

b. Schrijf een programma dat het kleinste getal n berekent waarvoor geldt dat n! minstens 100 nullen heeft.

Opdracht 3: Palindroom

Een palindroom is een woord waarin de letters symmetrisch gerangschikt zijn, zodanig dat het woord van achter naar voren gelezen hetzelfde is als van voor naar achter.

a. Definieer een functie met als argument een woord (string) die als resultaat True geeft indien het woord een palindroom is en False indien niet.

Een palindroomgetal of numeriek palindroom is een natuurlijk getal dat hetzelfde blijft wanneer zijn cijfers in omgekeerde volgorde worden geschreven, b.v. 13831.

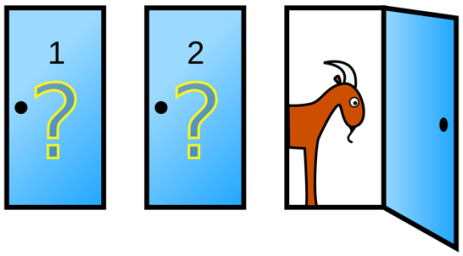
b. Schrijf een programma, gebruikmakend van deze functie, dat alle palindroomgetallen afdrukt tussen 2000 en 3000.

Opdracht 4: Het driedeurenprobleem

In een quiz wordt een deelnemer geconfronteerd met drie gesloten deuren.

Achter één van de deuren staat een auto en achter de twee andere twee deuren een geit. De deelnemer mag een deur aanwijzen en krijgt als prijs datgene wat zich achter die deur bevindt.

Als de deelnemer een deur heeft aangewezen, opent de presentator een van de andere deuren waarachter een geit staat. De presentator geeft de deelnemer daarna de mogelijkheid om te wisselen van gesloten deur, m.a.w. om in plaats van de eerst gekozen deur een andere nog gesloten deur te kiezen.



Wat moet de deelnemer doen? Kan hij beter wisselen van deur, of maakt het niets uit?

Is de kans op het winnen van de auto groter als de deelnemer van deur wisselt?

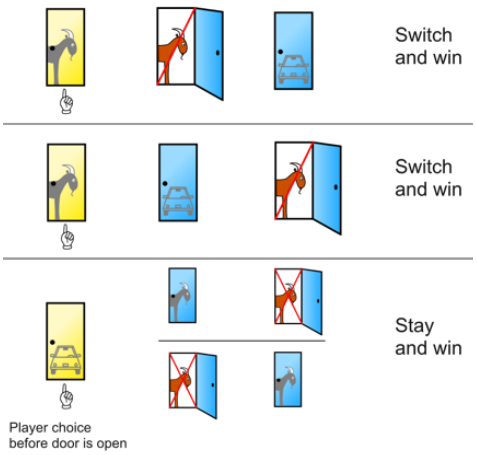
Origineel heet deze brain teaser The Monty Hall problem, gebaseerd op de Amerikaanse TV show Let's make a chance met als gastheer Monty Hall. Het probleem werd voorgelegd aan The America Statistician (een wetenschappelijke academische magazine) in 1975.

Veronderstel dat de deelnemer deur 1 kiest. Dan zijn er de volgende mogelijkheden:

Deur 1	Deur 2	Deur 3	Resultaat bij keuze 1 blijven	Resultaat veranderen van keuze
Geit	Geit	**Auto**	Deelnemer wint een geit	**Deelnemer wint een auto**
Geit	**Auto**	Geit	Deelnemer wint een geit	**Deelnemer wint een auto**
Auto	Geit	Geit	**Deelnemer wint een auto**	Deelnemer wint een geit

Indien de deelnemer bij zijn keuze blijft, is de kans op de auto $\frac{1}{3}$.

En indien de deelnemer zijn keuze wijzigt, is de kans op de auto $\frac{2}{3}$.



In de onderstaande definitie is de waarde van het argument switch gelijk aan True of False.

```
from random import *
def game(switch):
    # index voor prijs
    prijs=randint(0,2)
    # index voor keuze
    keuze=randint(0,2)
    # test resultaat keuze
    resultaat=keuze==prijs
    if switch:
        return not resultaat
    else:
        return resultaat
```

a. Ga na dat de bovenstaande functie het driedeurenprobleem simuleert en dat de keuze van de deelnemer effectief afhankelijk is van de waarde van het argument switch.

b. Gebruik deze functie om de spelsituatie 10000 keer te simuleren en bepaal hiermee de kans op een auto bij het niet wisselen van keuze en bij het wisselen van keuze.

Opdracht 5: Recursie

a. Schrijf een programma dat recursief de n^e (gehele $n \geq 0$) macht van een getal a berekent.

b. Benader $\sqrt{2}$ recursief gebruikmakend van de kettingbreuk:

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{\ldots}}}}$$

Opdracht 6: Iteratie

Schrijf een programma dat $\sqrt{2}$ benadert d.m.v. de Babylonische methode, een veelgebruikte methode die gebruikt wordt door computer en rekenmachines.

Deze methode is gebaseerd op de gelijkheid: $\sqrt{2} = \frac{1}{2}\left(\sqrt{2} + \frac{2}{\sqrt{2}}\right)$.

Vertrekkende van een startwaarde $a_0 = 1$, een ver van nauwkeurige benadering van $\sqrt{2}$, wordt $\sqrt{2}$ stap voor stap iteratief beter benaderd als volgt:

$$a_{n+1} = \frac{1}{2}\left(a_n + \frac{2}{a_n}\right) = \frac{a_n}{2} + \frac{1}{a_n}$$

Voor $a > 0$ de limiet van dit proces geldt: $a = \frac{a}{2} + \frac{1}{a}$, m.a.w $a^2 = 2$.