Inleiding programmeren

1e jaar wis-, natuur- en sterrenkunde

Universiteit van Amsterdam

Tentamen 2020-2021 (blok 1)

23 oktober 2020, 09:00-11:30 uur

Dit tentamen bestaat uit 5 vragen. Er zijn 10 punten te verdelen en je moet minimaal zes punten hebben om te slagen.

Corona-examen: We nemen dit online tentamen af zonder proctoring, maar dit vertrouwen geeft natuurlijk ook een verantwoordelijkheid aan jullie kant. We vragen jullie om dit examen **zelfstandig** te maken. Zonder hulp van buitenaf, dus zonder onderling overleg of hulp via internet. Zoals gebruikelijk is er een plagiaat-check en is er vanmiddag voor een aantal studenten een mondeling. **Jullie ontvangen rond 12:15 uur een mail** als je bent geselecteerd voor een mondeling.

Regels:

- Maak alle opdrachten in de file `tentamen.py` en lever deze in.
- Schrijf je naam en studentnummer bovenaan je file.
- In elke opdracht schrijf je één functie; voor uitvoer van het berekende antwoord gebruik je in elke functie gewoon `print`-opdrachten.
- Je mag in dit tentamen geen functies uit de `numpy` bibliotheek gebruiken, tenzij anders aangegeven in een opdracht.
- Zorg er bij het inleveren van je code voor dat bij het runnen van je programma **alle** functies uitgevoerd worden. Dat helpt bij het nakijken.

Opgave 1: Negenvoudig-deelbare getallen (2 punten)

Het getal 2520 is het kleinste getal dat precies deelbaar is (met rest nul) door de eerste tien natuurlijke getallen: 1, 2, 3, 4, 5, 6, 7, 8, 9 en 10. Er zijn ook getallen die door maar negen van deze getallen deelbaar zijn (in plaats van door alle tien). Deze getallen kunnen kleiner zijn dan 2520. Het kleinste 'negenvoudig-deelbare' getal is 360.

Schrijf een functie `Opgave1()` die alle negenvoudig-deelbare getallen kleiner dan 2520 vindt en zorg dat ze op het scherm geprint worden in het volgende format:

```
negenvoudig-deelbaar getal:  360
negenvoudig-deelbaar getal:  ...
negenvoudig-deelbaar getal:  ...
...
```

Let op: we vragen niet naar getallen die deelbaar zijn door *alle* getallen van 1 t/m 9, maar naar die die deelbaar zijn door precies negen getallen in de reeks 1 t/m 10.

Opgave 2: Ramanujan (2 punten)

Het getal 1729 is een van die getallen met een verhaal. In dit geval is het bekend geworden omdat het het nummer was van de taxi waarmee Hardy (een wiskundige) op bezoek ging in het ziekenhuis bij zijn vriend en collega Ramanujan. Hardy meldde dat 1729 helaas een oninteressant getal was, maar Ramanujan corrigeerde hem. Het was immers het kleinste getal dat te schrijven was als de som van twee derdemachten. Er geldt immers: $1729 = 1^3 + 12^3$ en $9^3 + 10^3$.

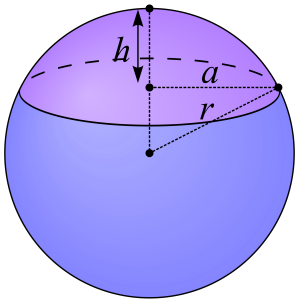
Schrijf een functie Opdracht 2() die het **op een na kleinste** gehele getal vindt dat op twee manieren te schrijven is als de som van twee derdemachten. Zorgt dat je programma de output als volgt geeft:

`Het getal xxx is te schrijven als:` $aaa^3 + bbb^3$ `en ook als` $ccc^3 + ddd^3$

Let op: als je alleen het getal zelf vindt krijg je de helft van de punten.

Opgave 3: Numeriek integreren (2 punten)

Schrijf een functie `Opgave3(h)` die op minimaal 2 decimalen nauwkeurig en met behulp van de Monte Carlo techniek de inhoud berekent van het 'dopje' (lichtpaars in de figuur) van een eenheidsbol ($x^2+y^2+z^2 = 1$), als $h = 0.25$ (hoogte van dopje).



Let op: de hoogte h is gedefiniëerd is ten opzichte van de top van de bol.

De opbouw van het programma volgt dezelfde stappen als die bij het numeriek integreren van functies:

1) Gooi een random punt in een kubus die de bol precies omsluit
   Elk punt wordt dus gegeven door 3 random getallen (x, y, z)
2) Bepaal de fractie van punten die in het 'dopje' terechtkomt: f_{dopje}
   en bereken daarmee de inhoud van het dopje.

Tip: De inhoud van een bol wordt gegeven door $\frac{4}{3}\pi r^3$. Test je programma door $h = 0$ of $h = 1$ te nemen omdat je dan de inhoud van het dopje kent. Zorg ook dat je antwoord stabiel is (in ieder geval de eerste twee decimalen) dus gooi genoeg punten.

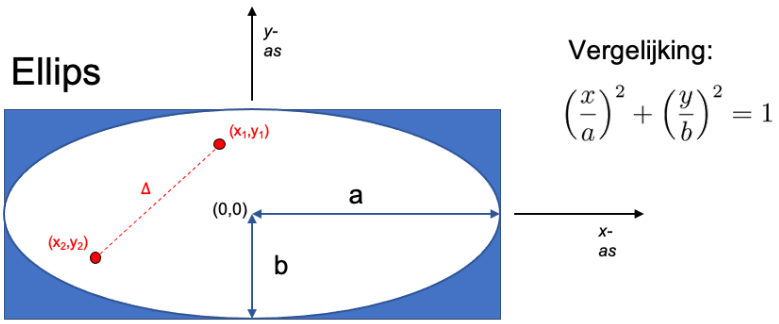
Als output op je scherm verschijnt: `Inhoud dop (h=0.25) is:  ...`

Om random getallen te gebruiken heb je de `random()` functie nodig. Zorg dat je programma dus begint met: `import random`

Opgave 4: Gemiddelde afstand tussen 2 punten in een ellips (2 punten)

Schrijf een functie `Opgave4(a,b)` die de gemiddelde afstand bepaalt tussen twee punten in een ellips met a=2 en b=1. Maakt de functie zo dat hij werkt voor willekeurige a en b.

In de figuur hieronder staat de algemene uitdrukking voor rand van de ellips.



Gebruik om de gemiddelde afstand te bepalen de volgende strategie:

Genereer heel vaak twee random punten (x_1, y_1) en (x_2, y_2) in een rechthoek met breedte 2a ($-2 \leq x \leq 2$ in ons geval) en hoogte 2b ($-1 \leq y \leq 1$ hier). Bepaal voor elk paar punten steeds eerst of **beide punten** binnen de ellips liggen. **Als dat zo is**, doe dan ook de volgende stappen voor deze zogenaamde 'in-de-ellips' paren:

1) Bereken de onderlinge afstand (Δ)
2) Bewaar de totale afstand (som van alle afstanden) voor deze 'in-de-ellips' paren
3) Bereken de gemiddelde afstand door de totale afstand te delen door het totaal aantal 'in-de-ellips' paren

Aan het eind van je programma moet de gemiddelde afstand (met 2 decimalen nauwkeurigheid) op het scherm geprint worden:

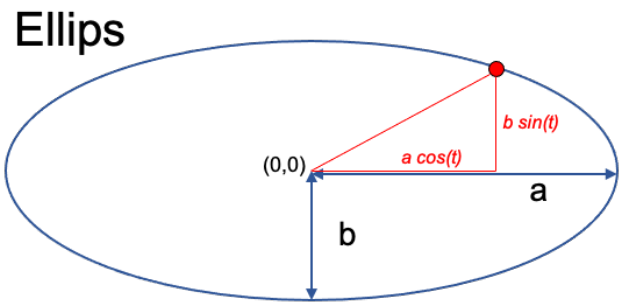
```
De gemiddelde afstand tussen 2 punten in de ellips(2,1) = x.xx
```

Tip: maak een grafiek van de 'in-de-ellips' punten. Liggen ze echt in de ellips?

Om random getallen te gebruiken heb je de `random()` functie nodig. Zorg dat je programma dus begint met: `import random`

Opgave 5: Bereken de omtrek van een ellips (2 punten)

Schrijf een functie `Opgave5(a,b)` die de omtrek berekent van een ellips met a=2 en b=1. Maakt de functie zo dat hij werkt voor willekeurige a en b, maar roep de functie aan met Opgave5(2,1).



Er is geen algemene uitdrukking voor de omtrek van een ellips, maar we gaan in deze opdracht gebruik maken van de volgende parametrisatie, waarbij $0 \leq t \leq 2\pi$.

$$x = a\cos(t) \qquad \text{en} \qquad y = b\sin(t)$$

Het idee is dat we de omtrek van de ellips gaan bepalen door kleine stapjes te nemen in t (stapjes van 0.0001) en steeds de afstand tot het vorige punt berekenen. We doen dit tot we rond zijn ($t = 2\pi$). Ga dus als volgt te werk:

1) Start bij t=0. Dat is het punt (a,0)
2) Neem een kleine stap in t en bereken de nieuwe x-waarde en y-waarde.
3) Bereken de afstand Δ tot het vorige punt en tel die afstand op bij de totale omtrek tot nu toe.
4) Doe dat net zolang tot t = 2π

Aan het eind van je programma moet de omtrek (met 2 decimalen nauwkeurigheid) op het scherm geprint worden:

```
De omtrek van de ellips(2,1) = x.xx
```

Tip:
- Controleer je programma door het aan te roepen met Opgave5(1,1). De oppervlakte van een cirkel weet je immers.
- Je mag in deze opdracht gebruikmaken van de `numpy` functie `arange()`