Opdrachten week 2 les 1 – Beslissen en recursie

Vorige week heb je kennis gemaakt met de programmeertaal Python en de IDE Thonny. Je bent inmiddels in staat om een verzameling functies te ontwerpen die samenwerken om een bepaald doel te bereiken. In deze les ga je een aantal Python-statements leren kennen en gebruiken waarmee je bepaalde code voorwaardelijk uit kunt laten voeren. De code wordt dan alleen uitgevoerd als aan een bepaalde voorwaarde wordt voldaan. Op deze manier is het Python-programma in staat om beslissingen te nemen.

Ook leer je dat een functie zichzelf kan aanroepen. Dit wordt recursie genoemd en je kunt het gebruiken om bepaalde code herhaald uit te laten voeren.

Je leert in deze les hoe je:

- de verschillende operatoren die Python heeft om getallen op elkaar te delen kunt gebruiken;
- booleaanse variabelen en expressies in Python kunt gebruiken;
- een programma beslissingen kunt laten nemen afhankelijk van de waarden van bepaalde variabelen;
- recursieve functies kunt schrijven en debuggen;
- input van de gebruiker kunt inlezen in je programma.

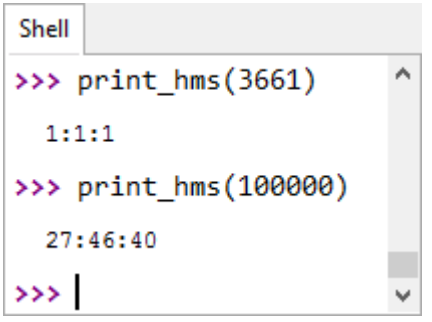
In de volgende les leer je om functies te schrijven die een returnwaarde hebben en leer je andere Python-statements kennen waarmee je bepaalde code herhaald uit kunt laten voeren.

Lees nu hoofdstuk 5 tot en met paragraaf 5.1 van het boek[1].

2.1.1 Schrijf een Python functie die een tijd gegeven in seconden afdrukt in uren, minuten en seconden. Test deze functie door hem aan te roepen in de Python Shell, zie figuur 1. Maak gebruik van de *floor division operator* `//` en de *modulus operator* `%`.

Lees nu paragraaf 5.2 en 5.3 van het boek.

[1] Allen B. Downey. *Think Python: How to Think Like a Computer Scientist*. 2de ed. Green Tea Press, 2016. ISBN: 978-1-4919-3936-9. URL: http://greenteapress.com/wp/think-python-2e/.



Figuur 1: Twee voorbeelden van het gebruik van de functie `print_hms`.

2.1.2 Schrijf een functie die de booleaanse waarde `True` afdrukt als het, als argument meegegeven jaar, een schrikkeljaar is en anders de booleaanse waarde `False` afdrukt. Een jaar is een schrikkeljaar als het:

- deelbaar is door 4 en
- niet deelbaar is door 100 behalve
- als het wel deelbaar is door 100 maar ook door 400.

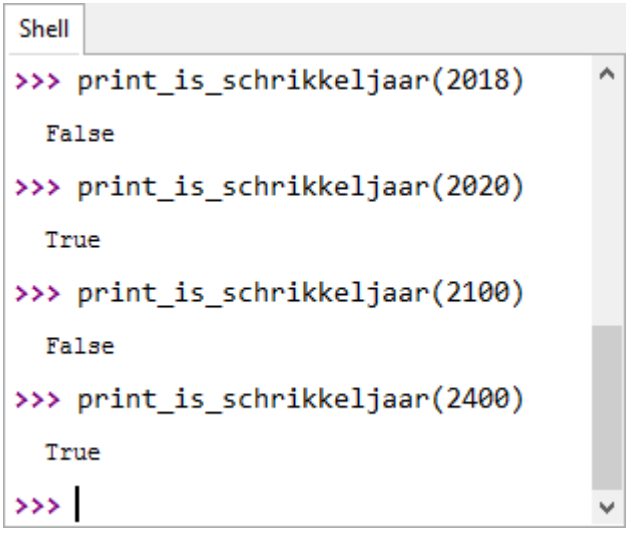
Vul eerst de waarheidstabel uit tabel 1 verder in en leidt hier de booleaanse formule voor de booleaanse variabele `schrikkeljaar` uit af uitgedrukt in de variabelen `deelbaar_door_4`, `deelbaar_door_100` en `deelbaar_door_400`. De waarheidstabel bevat veel don't cares. Bijvoorbeeld: omdat een getal dat niet deelbaar is door 4 nooit deelbaar kan zijn door 400 staat er een don't care op de tweede regel van tabel 1. Deze combinatie van logische waarden van `deelbaar_door_4` en `deelbaar_door_400` kan namelijk nooit voorkomen. De logische waarde van de booleaans variabelen `deelbaar_door_4`, `deelbaar_door_100` en `deelbaar_door_400` kun je bepalen met behulp van de modulo operator `%` en een vergelijkingsoperator `==`. Test de functie door hem aan te roepen in de Python Shell, zie figuur 2.

Lees nu paragraaf 5.4, 5.5, 5.6 en 5.7 van het boek.

2.1.3 Pas de functie die je hebt geschreven in opdracht 2.1.2 nu zo aan dat de uitvoer overeenkomt met figuur 3.

Tabel 1: Waarheidstabel om te bepalen of een jaar een schrikkeljaar is. De d staat voor don't care. Er moeten nog een aantal waarden worden ingevuld.

deelbaar_door_4	deelbaar_door_100	deelbaar_door_400	schrikkeljaar
0	0	0	0
0	0	1	d
0	1	0	.
0	1	1	.
1	0	0	1
1	0	1	d
1	1	0	.
1	1	1	.

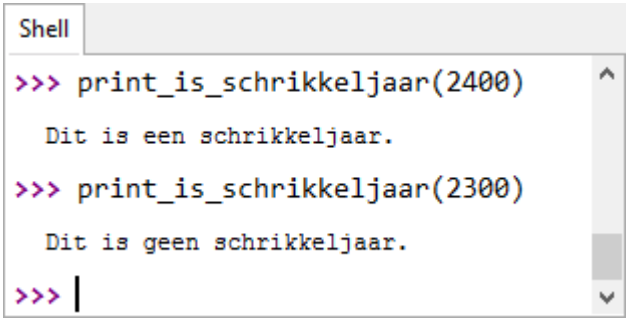


Figuur 2: Vier voorbeelden van het gebruik van de functie `print_is_schrikkeljaar`.

De oplossingen of wortels[2] van een vierkantsvergelijking $a \cdot x^2 + b \cdot x + c = 0$ kun je vinden met de bekende abc-formule:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$$

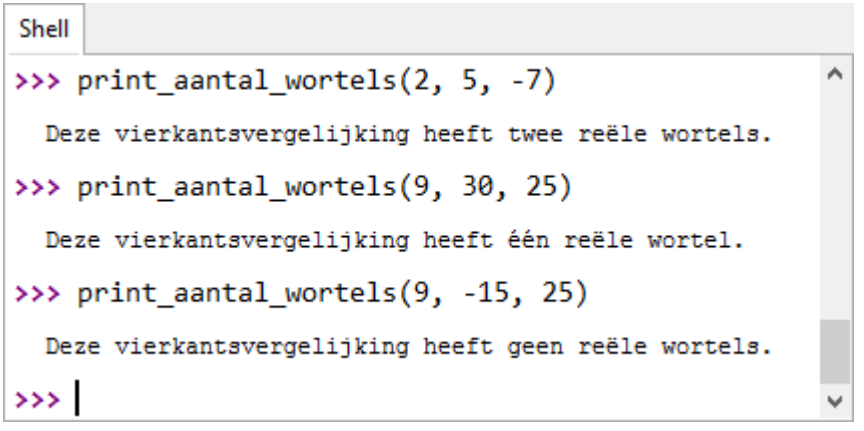
[2] In de wiskunde wordt met de wortel zowel de wortel van een getal als van een vergelijking aangeduid. Zie https://nl.wikipedia.org/wiki/Wortel_(wiskunde)#Wortel_van_een_vergelijking.



Figuur 3: Twee voorbeelden van het gebruik van de aangepaste functie `print_is_schrikkeljaar`.

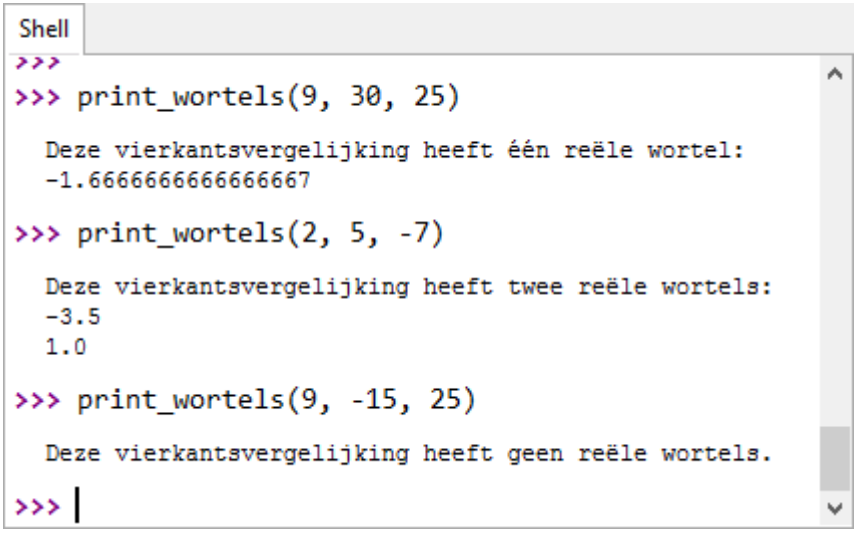
Deze vergelijking heeft echter geen reële oplossing als de zogenoemde discriminant $b^2-4\cdot a\cdot c$ negatief is. Er zijn dan geen reële wortels. Als de discriminant gelijk is aan nul, dan zijn x_1 en x_2 gelijk aan elkaar en is er dus maar één wortel.

2.1.4 Schrijf een functie genaamd `print_aantal_wortels` die drie parameters heeft genaamd a, b en c en die het aantal wortels van de vierkantsvergelijking $a\cdot x^2+b\cdot x+c=0$ afdrukt, zie figuur 4.



Figuur 4: Drie voorbeelden van het gebruik van de aangepaste functie `print_aantal_wortels`.

2.1.5 Pas de functie die je hebt geschreven bij opdracht 2.1.4 nu zo aan dat ook de waarden van eventuele wortels wordt afgedrukt. Noem deze functie `print_wortels`, zie figuur 5.



Figuur 5: Drie voorbeelden van het gebruik van de functie `print_wortels`.

2.1.6 Heb je in de functie die je hebt geschreven bij opdracht 2.1.4 een **`else`**-statement gebruikt? Zo nee, herschrijf de functie zodat je wel gebruik maakt van een **`else`**-statement.

2.1.7 Heb je in de functie die je hebt geschreven bij opdracht 2.1.4 een **`elif`**-statement gebruikt?

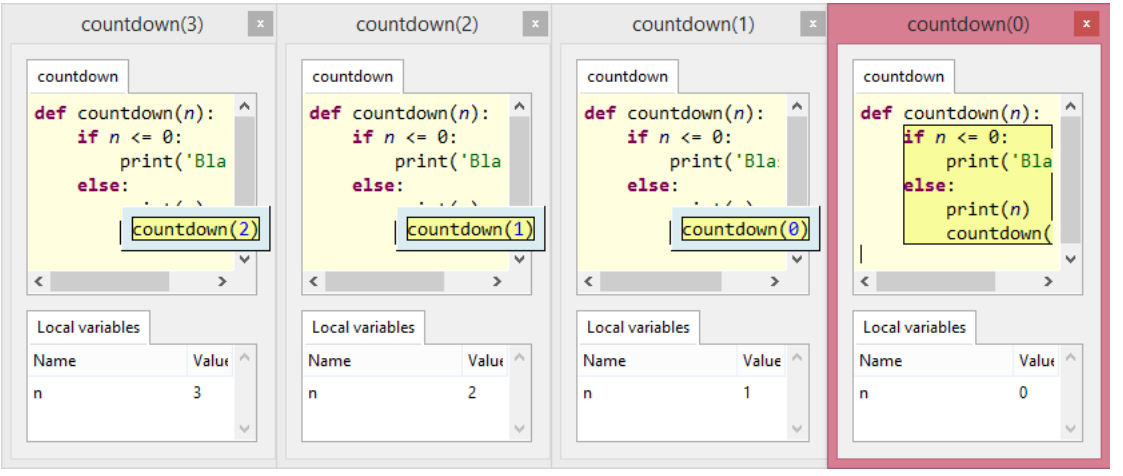
- Zo ja, herschrijf de functie zonder gebruik te maken van een **`elif`**-statement.
- Zo nee, herschrijf de functie zodat je wel gebruik maakt van een **`elif`**-statement.

Lees nu paragraaf 5.8, 5.9 en 5.10 van het boek.

2.1.8 Hoe kun je aan de code zien of een bepaalde functie een recursieve functie is?

2.1.9 Type de functie `countdown` gegeven in paragraaf 5.8 van het boek over in een programma `countdown.py`. Voeg de aanroep `countdown(3)` toe aan het programma. Voer het programma `countdown.py` stap voor stap uit door eerst op het de debug-knop te klikken of door op Ctrl + F5 te drukken en daarna steeds op de step-into-knop te klikken of op F7 te drukken. Elke keer als de functie `countdown` zichzelf aanroept, opent Thonny een nieuw window waarin je de uitvoering van die functie-aanroep kunt

volgen. Ook wordt de waarde van de parameter n van de betreffende functie-aanroep getoond. Je kunt zo'n window verplaatsen onafhankelijk van de overige Thonny windows, zie figuur 6. Zorg dat je goed begrijpt in welke volgorde de verschillende statements van het programma uitgevoerd worden.



Figuur 6: Bij het stap voor stap uitvoeren van de recursieve functie countdown wordt ook de waarde van de parameter n van de betreffende functie-aanroep getoond.

Als je iets afdrukt met de `print`-functie in Python dan wordt na het afdrukken van de betreffende informatie naar het begin van de volgende regel gesprongen. Als je dit niet wilt dan kun je een extra argument, genaamd end, aan de `print`-functie meegeven.

2.1.10 Gegeven is het programma `print_end.py` dat weergegeven wordt in listing 1. Wat is de uitvoer van dit programma? Hoe wordt de manier genoemd waarop het argument end aan print wordt doorgegeven[3]?

2.1.11 Gegeven is het programma `print_iets.py` dat weergegeven wordt in listing 2. Ga na wat het programma exact doet. Maak gebruik van de debugger van Thonny om het programma stap voor stap uit te voeren zodat je de werking kunt doorgronden. Zorg dat je goed begrijpt in welke volgorde de verschillende statements van het programma uitgevoerd worden. Voer het programma uit voor verschillende waarde van het argument (bijvoorbeeld: 2, 3, 9 en 15).

[3] Lees paragraaf 4.5 van het boek nogmaals als je het antwoord op deze vraag niet weet.

Beantwoord de volgende vragen:

- Wat is het verband tussen het argument en het getal dat wordt geprint?
- Wat zou een betere naam zijn voor de functie?
- Verklaar de naam van variabele `lsb`?

```
def tel_tot_en_met(n):
    for i in range(1, n + 1):
        print(i, end=', ')

tel_tot_en_met(4)
for i in range(2):
    print('hoedje van', end=', ')
print()
tel_tot_en_met(4)
print('hoedje van papier')
```

Listing 1: Het programma `print_end.py`.

```
def print_iets(n):
    lsb = n % 2
    if n > 0:
        print_iets(n // 2)
        print(lsb, end='')
    else:
        print()
```

Listing 2: Het programma `print_iets.py`.

2.1.12 Roep de functie `print_iets` aan met een grote waarde als argument bijvoorbeeld 2^{100}. Bedenk hoe je de waarde 2^{100} kunt meegeven als argument[4]. Klopt het antwoord? Hoe vaak wordt de functie `print_iets` aangeroepen?

Sommige problemen kunnen eenvoudiger met recursie dan met een andere methode opgelost worden. Een goed voorbeeld van zo'n probleem is het oplossen van Sudoku puzzels[5]. Een programma waarin recursie wordt gebruikt om Sudoku puzzels op te lossen kun je vinden

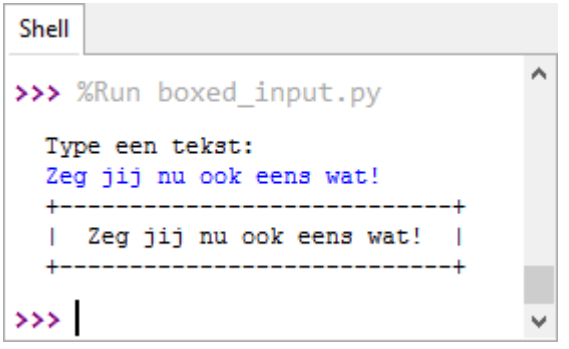
[4] Je kunt als argument ook een expressie (formule) opgeven. Python berekent eerst de waarde van deze expressie en geeft die vervolgens mee als argument aan de functie.

[5] https://nl.wikipedia.org/wiki/Sudoku.

op de EMS10 wiki[6]. Je hebt nu nog onvoldoende kennis van Python om dit programma te kunnen begrijpen, maar na afloop van week 3 zou dit wel moeten lukken!

Lees nu paragraaf 5.11 van het boek.

2.1.13 Bij opdracht 1.2.20 heb je een programma gemaakt dat een string met een rechthoek eromheen afdrukt. Pas dit programma nu zodanig aan dat de gebruiker om input wordt gevraagd en dat deze input met een rechthoek eromheen afgedrukt wordt, zie figuur 7.



Figuur 7: De input van de gebruiker wordt afgedrukt met een rechthoek eromheen.

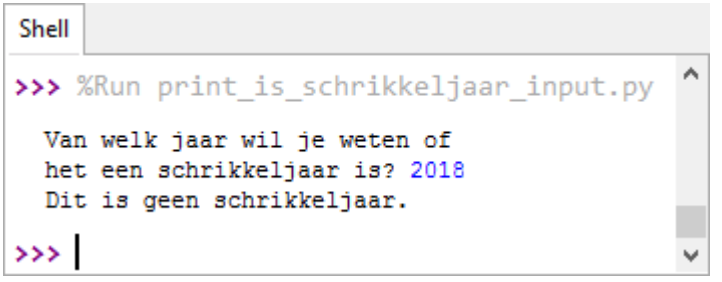
2.1.14 Bij opdracht 2.1.3 heb je een programma gemaakt dat afdrukt of een bepaald jaar een schrikkeljaar is. Pas dit programma nu zodanig aan dat de gebruiker wordt gevraagd om een jaar in te voeren en dat afgedrukt wordt of dit een schrikkeljaar is, zie figuur 8.

Lees nu paragraaf 5.12 van het boek.

Paragraaf 5.13 van het boek definieert de verschillende (vak)termen die in hoofdstuk 5 gebruikt worden. Het kan handig zijn om deze paragraaf te raadplegen als je niet meer weet wat met een bepaalde (vak)term bedoeld wordt.

Hier eindigen de verplichte opdrachten van week 2 les 1. Er volgen nu nog een gedeelte met oefeningen die je helpen om de leerstof van deze les beter te begrijpen en te onthouden.

[6] https://bitbucket.org/HR_ELEKTRO/ems10/wiki/sudoku.md.



Figuur 8: Het programma drukt af of het jaar dat is ingevoerd door de gebruiker een schrikkeljaar is.

Oefening

Alle onderstaande oefeningen (met de bijbehorende *uitwerking*) zijn ook beschikbaar op Anki flashcards zodat je ze regelmatig kunt herhalen. Deze stok kun je hier downloaden: EMS10 Week 2 Les 1.apkg.

2.1.15 Geef de Pythoncode om als de waarde van de variabele `x` niet deelbaar is door drie, de volgende tekst af te drukken:

```
x is niet deelbaar door 3
```

2.1.16 Het resultaat van een toets is (als floating point getal) opgeslagen in de variabele `toets_resultaat`. De toets is behaald als het resultaat groter of gelijk is aan 5,5. Geef de Pythoncode om de tekst `behaald` of `niet behaald` af te drukken afhankelijk van het behaalde resultaat.

2.1.17 Iemands leeftijd is opgeslagen in de variabele `leeftijd`. De onderstaande Pythoncode drukt af of dit de leeftijd van een twintiger is.

```
if leeftijd >= 20:
    if leeftijd < 30:
        print('twintiger')
```

Er zijn twee **if**-statements gebruikt. Maar het kan ook met één **if**-statement. Hoe?

2.1.18 Iemand gooit met een dobbelsteen en slaat de waarde op in de variabele `aantal_ogen`. De onderstaande code drukt even af als het aantal ogen even is en oneven als het aantal ogen oneven is.

```
if aantal_ogen == 2 or aantal_ogen == 4 or aantal_ogen == 6:
    print('even')
else:
    print('oneven')
```

Maar dit kan korter. Hoe?

2.1.19 Iemand gooit met een dobbelsteen en slaat de waarde op in de variabele `aantal_ogen`. De onderstaande code drukt even af als het aantal ogen even is en oneven als het aantal ogen oneven is.

```
if aantal_ogen % 2 == 0:
    print('even')
if aantal_ogen % 2 == 1:
    print('oneven')
```

Maar dit kan korter. Hoe?

2.1.20 Iemand gooit met een dobbelsteen en slaat de waarde op in de variabele `aantal_ogen`. De onderstaande code drukt **`True`** af als het aantal ogen even is en **`False`** als het aantal ogen oneven is.

```
if aantal_ogen % 2 == 0:
    print(True)
else:
    print(False)
```

Maar dit kan korter. Hoe?

2.1.21 Wat is de uitvoer van de volgende Pythoncode?

```
def wtf(n):
    if n > 0:
        print(n)
        wtf(n - 1)

wtf(7)
```

2.1.22 Wat is de uitvoer van de volgende Pythoncode?

```
def wtf(n):
    if n > 0:
        wtf(n - 1)
        print(n)

wtf(7)
```

2.1.23 Wat is de uitvoer van de volgende Pythoncode?

```
def wtf(n):
    if n > 0:
        print(n)
        wtf(n - 1)
        print(n)

wtf(3)
```

2.1.24 Wat is de uitvoer van de volgende Pythoncode?

```
def wtf(n):
    if n > 0:
        wtf(n // 2)
        if n % 2 == 0:
            print('0', end='')
        else:
            print('1', end='')
    else:
        print()

wtf(13)
```

2.1.25 Iemand doet mee aan de olympische race. De behaalde plaats is opgeslagen in de variabele `plaats`. De onderstaande code drukt af welke medaille behaald is.

```
if plaats == 1:
    print('goud')
else:
    if plaats == 2:
        print('zilver')
```

```
    else:
        if plaats == 3:
            print('brons')
        else:
            print('geen')
```

Maar dit kan korter. Hoe?

Paragraaf 5.14 van het boek bevat enkele oefeningen. Oefeningen 5.1 tot en met 5.4 zou je op dit moment zonder problemen moeten kunnen maken. Je kunt ze gebruiken als extra oefenmateriaal. Oefening 5.5 en 5.6 zijn iets pittiger, maar het is wel leuk om te zien hoe je recursieve figuren kunt tekenen met een schildpad. Studenten die een extra uitdaging zoeken kunnen zich uitleven met deze opdrachten, maar het is niet noodzakelijk om ze te maken.