Programming in Python

Typen opdrachten per Les

Dit document bevat per lesLes drie soorten opdrachten: oefeningen, een practicumopdracht en een extra uitdaging.

Oefeningen

De oefeningen worden (soms) als voorbeeld in de klas gebruikt en (soms) in de klas gemaakt. De oefeningen hoef je niet te uploaden op de online leeromgeving.

Een practicumopdracht

Aan de practicumopdrachten werk je zelfstandig of eventueel met een programming-maatje. Ze zijn iets uitdagender dan de oefeningen. De practicumopdrachten moet je uploaden op de online leeromgeving genaamd DLO. Voordat je je practicumopdracht uploadt laat het controleren door je programming-maatje via onderstaande checklist:

Python code conventions checklist	J	N
Voornaam en achternaam van de auteur en het doel van het programma staan als commentaar bovenaan het programma.		
De code wordt voorzien van voldoende commentaar, om de werking van de code te verduidelijken. Commentaar moet het waarom toelichten en niet herhalen wat er al staat. Dus niet: `index+=1 # verhoog index met 1` Maar wel: `index+=1 # verwijs naar volgende student in de rij`		
Functies worden voorafgegaan door commentaar waarin het doel van de functie, de mee te geven argumenten en de return value beschreven staan		
Variabelen, functies en methodes hebben duidelijke namen, waaruit het bedoelde gebruik duidelijk blijkt.		
Voor de naamgeving wordt gebruik gemaakt van lowerCamelCase (beginnen met kleine letter, en elk volgend woord direct daaraan vast met hoofdletter) of van snake_case (allemaal kleine letters, met underscore tussen de woorden).		
Gebruik voor de naamgeving wel consequent lowerCamelCase of snake_case. Gebruik ook consequent de Nederlandse, of de Engelse taal. Dit geldt ook voor je commentaar.		
Gebruik voor de naamgeving wel consequent lowerCamelCase of snake_case. Gebruik ook consequent de Nederlandse, of de Engelse taal. Dit geldt ook voor je commentaar.		
Regels niet langer dan 120 tekens, liefst korter dan 100		
Consequente indentatie (inspringing) van 4 spaties		
Importeer modules bovenin je programma, direct na de regels commentaar met het doel van het programma en je naam. Importeer geen modules, die je niet gebruikt.		
Verwijder statements die weg kunnen en maak je programma “schoon”. Dit geldt ook voor code die “commented-out” is.		
Vermijd "Magic Numbers": getallen die opeens in programmacode voorkomen en waarvan de functie niet duidelijk is. Dus niet: `bedragInEuro = aantalBitcoin * 7432` maar wel: `bedragInEuro = aantalBitcoin * koersBitcoinInEuro`		
Organiseer je programma in blokken programma-code. Een blok is een aantal regels code die functioneel bij elkaar horen. Scheidt de blokken met witregels. (Gebruik geen witregel na elke losse regel code.)		

Een extra uitdaging

Als je niet genoeg kan krijgen van de oefeningen en opdrachten kun je aan de extra uitdagingen beginnen. Ze worden echter niet in de klas besproken en je hoeft ze ook niet te uploaden op de online leeromgeving. Als je er niet (of samen met je programming-maatje) uitkomt of je wilt je werk laten zien aan de docent is dat natuurlijk altijd mogelijk.

Les 1

Les 1. Oefeningen

Oefening 1. Hello World

Deze eerste oefening heb je, als het goed is, al tijdens het hoor- / werkcollege gemaakt. Je moet Python en een Integrated Development Environment (IDE) downloaden en installeren. Een integrated development environment of IDE is software die een softwareontwikkelaar ondersteunt bij het ontwikkelen van software. Wij gebruiken in deze cursus PyCharm (Community)

1. Download en installeer de laatste versie van Python 3.x voor jouw operating system (OS) van: https://www.python.org/downloads/
2. Download en installeer de IDE van: https://www.jetbrains.com/pycharm/download
3. Start PyCharm. (Op Windows in het start menu onder Jetbrains - Jetbrains PyCharm)
4. Maak een nieuw Python project (File / New Project) aan in PyCharm en noem dit "helloworld". In het project verschijnt een folder genaamd 'venv'.
5. Klik op project naam, klik op de rechter muisknop en kies New Python File: Maak een file aan en noem deze 'helloworld' en klik op OK. PyCharm maakt nu een file helloworld.py aan (Conventie bij .py files: kleine letters).
6. Door te dubbelklikken op het 'helloworld.py' bestand wordt het lege bestand songtekst.py geopend.
7. Type op de eerste regel eerst het teken #. Alle tekst hierachter is commentaar: commentaar wordt niet door Python uitgevoerd, maar helpt ons het programma te begrijpen. Zet in de commentaarregel het doel van het programma neer, bijvoorbeeld: "Dit programma zegt hallo tegen de wereld!".
8. Maak nog een commentaarregel met je voornaam en achternaam.
9. Onder het commentaar komt je code. Gebruik hiervoor de functie `print()`.
10. Run je programma. Klik daarvoor met je rechter muisknop in het editorvenster en kies Run 'helloworld'. Als er foutmeldingen zijn, kijk dan goed wat er staat, om het probleem op te lossen.

Oefening 2. Datatype int

Het datatype `int` gebruiken we voor **gehele getallen**. In het werkcollege hebben we ook al enkele *operatoren* gezien. Hiermee kunnen we rekenkundige operaties uitvoeren op bijvoorbeeld variabelen van het type `int`.

1. Zoek op welke operatoren Python nog meer kent. Welke operator symbool gebruikt Python voor vermenigvuldigen, delen en machtsverheffen?
2. Stel we hebben een rechthoek met lengte 5 en breedte 3. Gebruik een *operator* om de oppervlakte te berekenen en druk het resultaat af op het scherm met `print()`.
3. Stel we hebben een pizza met 8 punten en we willen de pizza verdelen onder 5 hardwerkende programmeurs. Hoeveel pizza punten krijgt elke programmeur? Gebruik een *operator* om het aantal pizza punten te berekenen en druk het resultaat af op het scherm

met print.

Het valt je misschien op dat het resultaat van vraag 3 geen heel getal oplevert. Dus de waarde is ook niet van het type `int`. Niet-gehele getallen, dus getallen met cijfers achter de komma zijn van het datatype `float`.

Oefening 3. Datatype string

We hebben net gezien dat Python enkele operatoren kent, waarmee we kunnen rekenen. Sommige daarvan kunnen we ook voor het datatype `string` gebruiken. Zo kunnen we bijvoorbeeld strings optellen / samenvoegen met de `+` operator.

1. Type: `print("Hello"+"World")` Wat gebeurt er?
2. Dat is nog niet ideaal: er ontbreekt een spatie tussen Hello en World. Kun je het voorgaande commando aanpassen om wel een spatie tussen de twee woorden te krijgen, zonder de strings "Hello" en "World" zelf te veranderen?
3. Een leuke truc is, om strings te vermenigvuldigen. Stel: je hebt 100 uitroeptekens nodig. Een echte programmeur is natuurlijk te lui om dat met de hand in te typen, en wil dat automatiseren!

   Type: `print(100 * "!")` Wat gebeurt er?

Oefening 4. Quotes en special characters

Strings beginnen en eindigen met quotes: `"dit is mijn string"`. Dat is een probleem als er quotes in de string zelf voorkomen.

1. Probeer de volgende `string` te printen, inclusief de quotes om "Python is leuk":

```
Ada Lovelace zei: "Python is leuk".
```

2. Door de extra quotes weet Python niet meer waar de string moet eindigen. Dit probleem kunnen we oplossen door de dubbele quotes rondom de string te vervangen door enkele quotes: 'string'
3. Probeer opnieuw de string te printen, maar gebruik nu enkele quotes om de volgende string:

```
Ada Lovelace zei: "Python is leuk".
```

Hiermee is het probleem niet helemaal opgelost. Wat nu, als je een string wil afdrukken met zowel enkele als dubbele quotes er in? Dan kun je gebruik maken van een **escape sequence** door een backslash: `\` voor een speciale karakter te plaatsen. Bijvoorbeeld:

Escape Sequence	Description	Example	Output
\`	Prints single-quote	`print("\'")`	`
\"	Prints double quote	`print("\"")`	"
\b	Backspace removes previous character	`print("ab\bc")`	ac
\\	Prints backslash	`print("\\")`	\

4. Zoek op welke *escape sequences* Python nog meer kent. Welke escape sequence gebruiken we voor een nieuwe regel (line feed)?

Oefening 5. Letters in een string

Een `string` is eigenlijk een serie letters. Soms wil je één bepaalde letter uit een string gebruiken. Dat kan door achter de `string` het nummer van de gewenste letter in blokhaken: [ ] te zetten. De letters worden van links naar rechts geteld vanaf 0.

Probeer het maar: onderstaande code geeft het karakter `h` terug.

```
print("hallo"[0])
```

En onderstaande code geeft het karakter `a` terug.

```
print("hallo"[1])
```

1. Pas het onderstaande commando aan om te zorgen dat er een kleine letter e wordt geprint:

```
 print("Python4Ever"[x])
```

Oefening 6. Casten van datatypen

We kunnen de datatypes `int` en `string` naar elkaar omzetten met de functies: `int()` en `str()`. Dit omzetten van het ene datatype naar het andere heet *casten*.

We hebben al eerder een functie gebruikt, namelijk `print()`. Net als bij print geven we de benodigde informatie mee tussen haakjes: ( ).
Onderstaande functie geeft de `int` waarde `216` terug:

```
int("216")
```

En onderstaande functie geeft de `string` waarde `"216"` terug:

```
str(216)
```

We kunnen in principe van elke `int` een `string` maken, maar kun je ook van elke string een int maken?

1. Bekijk de twee regels code hieronder. Welke uitkomst verwacht je bij regel 1 en bij regel 2? Probeer de codes uit. Klopte je verwachting?

```
print( int("4") + int("2") ) # regel 1
print( str(4) + str(2) )     # regel 2
print( int("hallo ik ben een string")) # regel 3
```

Oefening 7. Datatype boolean

Behalve de datatypen `int`, `string`, `float` (wordt later behandeld) heb je het datatype `boolean`. Dit datatype kan alleen de waardes: `True` en `False` aannemen.

We hebben eerder de rekenkundige operaties behandeld waarbij er een getal het resultaat is. Dit getal kan van een geheel getal (`int`) zijn of een decimaal getal (`float`).
Sommige operaties hebben slechts waar of onwaar als resultaat. Dit soort operaties heten *logische operaties*. Logische operaties hebben ook hun eigen operatoren zoals: `>, >=, <, <=, ==`.

Bijvoorbeeld: `1 > 2` is onwaar want 1 is natuurlijk niet groter dan 2. Dus het resultaat van de operatie levert een waarde `False` op.

1. Wat zijn de resultaten van de volgende regels code?

```
print(5<= 4)
```

```
print("py"+"thon"=="python")
```

```
print(4>=3)
```

```
print(4>=3)
```

Oefening 8. Variabelen en bewerkingen

Een variabele is niks anders dan een naam die verwijst naar een bepaald type informatie. De informatie kan variëren en daarom heet het dus een variabele.

Bij het programmeren kun je zelf waardes aan een variabele toewijzen. Bijvoorbeeld:

```
naam = "George"
achternaam = "Boole"
getrouwd = True
student = False
geboorteJaar = 1815
```

Hoeveel BitCoin kun je kopen voor 1000 euro? Dit rekenen we uit met een klein Python programma.

1. Maak een variabele met de naam `aantalEuro` en geef deze de waarde `1000`.
2. Maak vervolgens een variable genaamd `aantalBitCoin`. Geef deze als waarde: het aantal euro maal de huidige wisselkoers.
3. Druk het resultaat af.

Oefening 9. Assignment operators

Om een waarde aan een variabele toe te kennen, gebruiken we een **assignment.** Bijvoorbeeld: `aantal = 5`. Als we later in het programma de waarde willen verhogen, kan dat weer met een assignment, bijvoorbeeld:

```
aantal = aantal + 1
```

Dit soort code komt zo vaak voor, dat er een shortcut voor is gemaakt. Dat scheelt je weer programmeerwerk! Je gebruikt dan:

```
aantal+=1 # (korte versie van aantal = aantal + 1)
```

Ander voorbeeld:

```
aantal+=6 # (korte versie van aantal = aantal + 6)
```

1. Zoek online op, welke van deze korte *assignment operators* er nog meer zijn. Bijvoorbeeld voor aftrekken, delen en vermenigvuldigen.
2. Welk operator gebruik je om de huidige waarde van de variabele genaamd `aantal` te delen door 5 en het resultaat weer op te slaan in dezelfde variabele?

Oefening 10. Input en output

Je kunt behalve informatie naar de terminal uitprinten via de functie `print`, ook informatie vanuit de terminal in je programma krijgen. Dat doe je door gebruik te maken van de functie `input()`.

Voorbeeld:

```
naam = input("Wat is je naam: ")
print("Wat is "+ naam + " een bijzondere naam zeg!")
```

Schrijf een programma dat de oppervlakte en de omtrek van een rechthoek uitrekent.

1. Gebruik aparte variabelen in je programma die je nodig hebt om de oppervlakte en omtrek te berekenen.
2. Laat de gebruiker de hoogte en de breedte van de rechthoek opgeven. Gebruik hiervoor de functie `input()`.

3. Je programma moet de oppervlakte en omtrek uitprinten.

Les 1. Practicumopdracht

Leeftijdberekening

We zijn begonnen met een aantal opdrachten om de theorie onder de knie te krijgen. Elke Les sluiten we af met een meer ingewikkelde opdracht om te zien of je de stof echt goed beheerst - de practicumopdracht. Je uitwerkingen van de practicumopdrachten upload je steeds op de online leeromgeving. Het doel van onderstaande opdracht is zelfstandig een eerste Python programma te maken met meerdere regels code.

Schrijf een applicatie die het volgende doet:

- Vraag de naam van een persoon en lees deze in in een variabele.
- Vraag vervolgens het geboortejaar van deze persoon en lees deze in.
- Bereken hierna de leeftijd van deze persoon en druk deze af.
- Tot slot, druk de leeftijd af in "Venusjaren". Een jaar op Venus is 0.62 keer de lengte van een jaar op Aarde. Dus Venus draait sneller om de zon dan dat de Aarde om de zon draait.

Hieronder een voorbeeld van input / output van deze applicatie. Om te verduidelijken wat de input van de gebruiker is is de input hieronder ***schuin en vetgedrukt*** afgebeeld.

> Hoe heet je? ***Alan Turing***
>
> Wat is je geboortejaar? ***1912***
>
> Beste Dennis, in 2020 zal je leeftijd op aarde 108 zijn.
>
> En je leeftijd is dan 174.193548387 in Venusjaren.

Het is handig je programma stap voor stap te maken. Controleer tussentijds steeds of je programma werkt, zoals verwacht.

1. Focus je eerst op de 'normale' leeftijd. Hoe kan je deze berekenen?
2. Welke informatie heb je hierbij nodig en welke variabelen ga je gebruiken?
3. Maak in PyCharm een nieuw Python file genaamd `venusjaren`.
4. Gebruik `input()` om de gewenste informatie aan de gebruiker te vragen en op te slaan in variabelen.
5. Bereken de leeftijd en druk het resultaat af met `print()`
6. Test je programma door het te runnen. Los eventuele foutmeldingen op. Als je programma zonder problemen runt: controleer of de leeftijdsberekening klopt.
7. Als alles correct werkt, voeg dan als laatste de berekening van de leeftijd in Venusjaren toe. Een jaar op Venus is 0.62 keer een jaar op aarde. (Op Venus ben je dus ouder of jonger?)
8. Run opnieuw je programma en controleer of de berekening klopt. Komt de uitvoer precies overeen met het voorbeeld hierboven?
9. Voldoet je programma aan de code conventions? Laat je programma via de conventions checklist nalopen door je programming-maatje.

10. Heb je je programma verbeterd? Upload dan de source, dat is het `venusjaren.py` bestand, online.

Les 1. Extra uitdaging

BMI berekenen

De BMI (Body Mass Index) is een maat om te berekenen of iemand een gezond gewicht heeft. De formule voor het berekenen van de BMI is als volgt:

> BMI = Gewicht in kilogram / (Lengte in meter * Lengte in meter)

Schrijf een applicatie die het volgende doet:

- Vraag het gewicht van een persoon in kilogram en lees deze in.
- Vraag vervolgens de lengte van deze persoon in centimeters en lees deze in.
- Bereken hierna de BMI van deze persoon en druk deze af.

Hieronder een voorbeeld van input/output van deze applicatie. Input die de gebruiker invoert is ***schuin en vetgedrukt***.

> Gewicht in kilogram : ***69.6***
>
> Lengte in centimeter: ***175***
>
> BMI: 22.726530612244897

Het is handig je programma stap voor stap te maken. Controleer tussentijds steeds of je programma werkt, zoals verwacht.

1. Gebruik variabelen voor het opslaan van het gewicht en de lengte. Gebruik de juiste datatypes voor de variabelen: gewicht is een decimaal, lengte is een geheel getal. **Hint**: decimalen hebben het datatype `float`. Je kunt een `int` of een `string` *casten* naar een decimaal met de functie `float()`.
2. Vergeet niet om de lengte in centimeter om te zetten naar meters.
3. Je kunt ervoor kiezen om eerst de BMI te berekenen en deze op te slaan in een variabele. Ook de BMI is een decimaal. Je kunt er ook voor kiezen om de berekening direct af te drukken met `print()`.
4. Schrijf je programma.
5. Voldoet je programma aan de code conventions?

Les 2

Les 2. Oefeningen

Oefening 1. if statements

Onderstaande code vraagt een getal aan de gebruiker. Als de gebruiker het getal 10 invoert, wordt de tekst: 'OK' afgedrukt.

```
getal = int(input("Geef een getal:"))
if getal == 10:
  print("OK")
```

1. Pas in bovenstaande code de if-statement aan, zodat de tekst 'OK' alleen wordt afgedrukt als het getal ongelijk is aan 10. Test je code.
2. Pas opnieuw de if-statement aan, zodat de tekst 'OK' wordt afgedrukt als het getal groter dan of gelijk is aan 10. Test je code. Krijg je ook 'OK' voor het getal 10 zelf?
3. Pas opnieuw de if-statement aan, zodat de tekst 'OK' wordt afgedrukt als het getal groter is dan 20, en kleiner dan 30. Test je code. Bij de getallen 20 en 30 zelf, zou je geen 'OK' moeten krijgen
4. Pas weer de if aan. Zorg er dit keer voor dat 'OK' wordt afgedrukt als één van de volgende getallen wordt ingevoerd: 12, 36 of 128.

Oefening 2. if statements en het boolean datatype

1. Wat verwacht je van de code hieronder:

```
a = True
b = True
c = False

if a and b and not c: print("ja")
else: print("nee")
```

Oefening 3. if else statements

1. Schrijf een programma, dat de naam vraagt van de gebruiker. Als de gegeven naam "Willem Alexander" of "Maxima" is, dan geeft het programma de uitvoer: *"Welkom uwe Hoogheid".* Bij elke andere naam geeft het programma de uitvoer Welkom, en de naam van de gebruiker. Dus bijvoorbeeld: *"Welkom Klaas".*

Oefening 4. if elif else statements

1. Schrijf een kort programma, dat allereerst aan de gebruiker een score vraagt tussen de 0 en de 100. Het programma geeft vervolgens de onderstaande uitvoer. Gebruik in je code `if`, `elif` en `else`.
    - voor een score onder de 0 of boven de 100 is de output: *"ongeldige score"*
    - voor een score hoger dan of gelijk aan 80 is de output:*"geweldig"*
    - Voor een score hoger dan of gelijk aan 55 (maar kleiner dan 80) is de output: *"voldoende"*
    - Voor een score onder de 55 is de output: *"onvoldoende"*

Oefening 5. korte if-statement

If-statements komen erg vaak voor in programma's en vaak wordt er maar één regel code uitgevoerd als de voorwaarde al dan niet waar is. Een voorbeeld hiervan is onderstaande code:

```
if a > b:
    print(a)
else:
    print(b)
```

Dit soort if-statements komen zo vaak voor, dat er in Python een korte versie voor is. Deze code kan ook op één regel worden gezet. Onderstaand programma doet precies hetzelfde, maar is veel korter!

```
print(a) if a > b else print(b)
```

1. Schrijf een programma dat een getal vraagt aan de gebruiker. Als het getal groter dan of gelijk is aan 0, dan geeft het programma als output: *" positief getal",* anders de output: *"negatief getal"* Gebruik bij deze opdracht een kort if-statement (op één regel)

Oefening 6. Gebruik van modules Math

Voor wiskundige berekeningen kunnen we gebruik maken van bestaande functies uit de **math module**. Voordat we deze kunnen gebruiken, moet deze module eerst in ons programma geïmporteerd worden! Gebruik hiervoor bovenaan je programma de volgende regel:

```
import math
```

1. Zoek online op, welke functies zoal beschikbaar zijn in de math module.

2. Schrijf een programma dat, met behulp van functies uit de math module, de volgende berekeningen uitvoert:
    - de wortel van 16
    - 2 tot de macht 5
    - de tangens van 10
    - 6.216 naar boven afgerond

Oefening 7. Random

Een andere nuttige module is de **random module**. Hierin zitten allerlei functies om willekeurige getallen te genereren.

1. Zoek online op, welke functies zoal beschikbaar zijn in de random module.
2. Schrijf een programma dat, met behulp van functies uit de random module, een willekeurig getal kiest tussen de 0 en de 5. Print vervolgens dit getal af naar de terminal.
3. Test je code enkele keren. Komen de resultaten overeen met je verwachtingen? Hint: ook de random module moet eerst in je programma geïmporteerd worden.
4. Pas je programma aan. Kies weer een willekeurig getal tussen de 0 en de 5, maar druk dit niet af. Vraag vervolgens een getal aan de gebruiker. Als de twee getallen overeenkomen, druk de tekst *"Goed geraden"* af. Als de getallen verschillen, druk dan de tekst *"Volgende keer beter"* af.
5. Test je code enkele keren. Komen de resultaten overeen met je verwachtingen?

Les 2. Practicumopdracht

Money exchange

Schrijf een programma dat uitrekent hoeveel euro's een klant krijgt voor zijn buitenlandse valuta. In het programma kunnen worden omgewisseld:

- US dollar
- GB pound
- Japanse Yen

Zorg dat de invoer en uitvoer van het programma er als volgt uitziet (de wisselkoers kan verschillen):

> Valuta (1 = US dollar, 2 = GB pounds, 3 = Yen): ***3*** In te wisselen bedrag: ***12000*** Voor 12000 Yen krijgt u 102.26 Euro.

Voor het wisselen moet de klant transactiekosten betalen. Deze bedragen 1,5%, met een minimum van 2 euro per transactie en een maximum van 15 euro per transactie.

Zorg dat de invoer en uitvoer van het programma er als volgt uitziet:

> Valuta (1 = US dollar, 2 = GB pounds, 3 = Yen): ***3*** In te wisselen bedrag: ***12000*** Voor 12000 Yen krijgt u 102.26 Euro. De transactiekosten bedragen 2.0 Euro. U ontvangt 100.26 Euro.

1. Schrijf eerst in pseudo-code op wat je programma moet doen, voordat je gaat programmeren.
2. Maak je programma stap voor stap: zorg eerst dat het verkopen van valuta werkt en dat de bedragen kloppen. Ga pas daarna verder met de transactiekosten.
3. Gebruik in je programma variabelen om de wisselkoersen op te slaan.
4. Rond decimalen af zoals bij bedragen gebruikelijk is: op 2 cijfers achter de komma. Als de laatste decimaal een 0 is zal je deze in de uitvoer echter niet zien.
5. Voldoet je programma aan de code conventions? Laat je programma via de conventions checklist nalopen door je programming-maatje.
6. Heb je je programma verbeterd? Upload dan de source, dat is het `.py` bestand, online.

Les 2. Extra uitdaging

BMI berekenen vervolg

Deze opdracht is een vervolg op de BMI opdracht van vorige Les. Maak de volgende aanpassingen aan je programma:

1. Pas je berekening van de BMI zo aan dat je in plaats van `lengte*lengte` gebruik maakt van lengte in het kwadraat. Gebruik hiervoor `math.pow()` uit de math module.
2. Rond de BMI af op 1 cijfer achter de komma. Hiervoor is in Python standaard een functie beschikbaar, namelijk: round(). Zoek online op, hoe je deze kunt gebruiken.
3. Bepaal op basis van de BMI of iemand een gezond gewicht heeft. Maak hierbij gebruik van de onderstaande tabel. Gebruik in je code `if`, `elif` en `else`.
4. Voldoet je programma aan de code conventions?

BMI	**Categorie**
tot 18,5	Ondergewicht
18,5 - 25,0	Gezond gewicht
25,0 - 30,0	Overgewicht
Meer dan 30,0	Obesitas

Hieronder een voorbeeld van input/output van de applicatie na de wijzigingen. Input die de gebruiker invoert is ***schuin en vetgedrukt***.

> Gewicht in kilogram : ***69.6***
>
> Lengte in centimeter: ***175***
>
> BMI: 22.7
>
> Gezond gewicht

Het is handig je programma stap voor stap te maken. Controleer tussentijds steeds of je programma werkt, zoals verwacht.

Les 3

Les 3. Oefeningen

Oefening 1. list

Gegeven is onderstaande `list`:

```
steden = ["Amsterdam", "Praag", "Lissabon", "Londen", "Parijs", "Milaan"]
```

1. Druk achtereenvolgens af: het eerste element uit de lijst ("Amsterdam"), het derde element ("Lissabon") en het laatste element ("Milaan").
2. Met de functie `len()` kun je de lengte van een lijst bepalen. Gebruik deze functie om het aantal elementen in de lijst met steden te bepalen.
3. Met de functie `len()` kun je ook de lengte van een `string` bepalen.

   Gebruik deze functie om het aantal letters in de tweede string ("Praag") in de lijst met steden te bepalen.
4. Je kunt items toevoegen aan de list met de += operator. Probeer onderstaande code:

   ```
   steden=["Amsterdam","Praag","Lissabon"]
   steden+=["Brussel","Warschau"]
   print(steden)
   ```
5. Je kunt afzonderlijke items in de list een nieuwe waarde geven, bijvoorbeeld:

   ```
   steden[2] = "Kopenhagen"
   ```

   Pas in de list met steden het eerste element aan. Verander "Amsterdam" in "Rotterdam".
6. Controleer het resultaat door de lijst met steden opnieuw af te drukken.

Oefening 2. loops en berekeningen 1

1. Schrijf een programma, dat de tafel print van een door de gebruiker op te geven geheel getal. Gebruik hierbij een loop. De in- en uitvoer van het programma moet er als volgt uitzien:

   > Welke tafel wilt u printen? **3** De tafel van 3: 3 6 9 12 15 18 21 24 27 30
2. Breid het programma zo uit, dat de gebruiker ook kan aangeven hoeveel getallen hij/zij wil printen. De in- en uitvoer ziet er dan als volgt uit:

   > Welke tafel wilt u printen? **12** Hoeveel getallen wilt u printen? 3 De tafel van 12: 12 24 36

3. We breiden ons programma dat tafels afdrukt nogmaals verder uit. Zorg er nu voor, dat de gebruiker meerdere tafels achter elkaar kan printen. Het programma stopt pas wanneer voor de tafel van 0 wordt gekozen. Een voorbeeld van de in- en uitvoer:

   > Welke tafel wilt u printen (0 = stoppen)? ***12*** Hoeveel getallen wilt u printen? ***3*** De tafel van 12: 12 24 36 Welke tafel wilt u printen (0 = stoppen)? ***2*** Hoeveel getallen wilt u printen? ***4*** De tafel van 2: 2 4 6 8 Welke tafel wilt u printen (0 = stoppen)? ***0***

Hints:

- Plaats je eerdere code binnen een nieuwe loop, die net zolang doorgaat tot de gebruiker 0 kiest. Gebruik je daarvoor `for` of `while?`
- Je krijgt te maken met een geneste loop: een loop in een loop. Let goed op welke code binnen welke loop hoort en zorg ervoor dat je zorgvuldig omgaat met het inspringen!

Oefening 3. loops en berekeningen 2

Gegeven is onderstaande lijst met getallen:

```
getallen = [3,7,0,15]
```

1. Schrijf een programma, dat elk getal in deze `list` met twee vermenigvuldigt en het resultaat afdrukt op het scherm. Gebruik hierbij een loop. De uitvoer is dus:

   > 6 14 0 30

2. Pas je programma aan, zodat elk getal in de `list` wordt gedeeld door 3 (in plaats van vermenigvuldigd met 2). Rond de resultaten af op 1 cijfer achter de komma. De uitvoer is dus:

   > 1.0 2.3 0.0 5.0

Oefening 4. Elementen in een list aanpassen

1. We beginnen met onderstaande `list` met getallen:

   ```
   getallen = [12, 2, 1, 17, 5]
   ```

   We gaan nu de waardes in de `list` updaten. Schrijf een kort programma dat elk getal in deze `list` met twee vermenigvuldigt en het resultaat opslaat in de `list`, op de positie van het oude getal.

   Gebruik tenslotte: `print(str(getallen))` om het resultaat te controleren. De uitvoer is dus:

   > [24, 4, 2, 34, 10]

   Deze opdracht is moeilijker dan het lijkt! Een paar tips:

    - Gebruik `getallen[x] = nieuweWaarde` om de waarde op positie x van de `list` te veranderen
    - We hebben dus een teller nodig, die de positie bijhoudt. Dat leent zich tot een oplossing met `for x in range():`
    - Maar wat moet de range dan zijn? Met `len()` kunnen we de lengte van de `list` bepalen...
2. We kunnen ook getallen toevoegen aan `list` via de methode `append()`. Een voorbeeld:

```
getallen = [12, 2, 1, 17, 5]
print("Voor append: "+str(getallen))
getallen.append(42)
print("Na append: "+str(getallen))
```

   Schrijf nu een programma dat de eerste 10 getallen van de Fibonacci reeks via de `.append()` methode toevoegt aan een `list` genaamd `fibonacciReeks`. Print de getallen via een loop naar de output.

Les 3. Practicumopdracht

Getallen raden

Stappenplan

1. Schrijf een programma dat een willekeurig raad-getal van 1 - 100 genereert. Gebruik hiervoor de `random` module. De gebruiker kan vervolgens herhaaldelijk een getal invoeren, net zo lang tot hij het gegenereerde raad-getal heeft geraden. Gebruik hiervoor een loop.
2. Na elke incorrecte gok geeft het programma aan of het getal te hoog of te laag was. Als het getal geraden is, wordt de gebruiker gefeliciteerd.
3. Breid het programma uit zodat de gebruiker van tevoren kan aangeven tussen welk waardes het raad-getal moet zijn.
4. Breid het programma uit zodat de gebruiker kan aangeven hoe vaak er geraden mag worden. Zorg voor een foutmelding als er minder dan 1 of hoger dan 5 wordt ingevoerd en vraag het opnieuw. Gebruik hiervoor een loop.
5. Breid het programma verder uit zodat wanneer het raad-getal is geraden of het aantal keer dat geraden mocht worden is bereikt je dan alle invoer wordt getoond met de informatie of het te hoog of te laag was. Gebruik hiervoor een `list`.
6. Voldoet je programma aan de code conventions? Laat je programma via de conventions checklist nalopen door je programming-maatje.
7. Heb je je programma verbeterd? Upload dan de source, dat is het `.py` bestand, online.

Voorbeeld input en output

Hieronder een voorbeeld van input/output van de applicatie bij geen succes. Input die de gebruiker invoert is ***schuin en vetgedrukt***.

> Voer de minimum waarde in van het raad-getal: ***5*** Voer de maximum waarde in van het raad-getal: ***35*** Hoe vaak mag er geraden worden [1-5]: ***6*** Je mag alleen een getal van 1 tot en met 5 invoeren. Hoe vaak mag er geraden worden [1-5]: ***3*** Raad het raad-getal: ***4*** Fout, te laag! Raad het raad-getal: ***5*** Fout, te laag! Raad het raad-getal: ***6*** Fout, te laag!
>
> Het raad-getal was 34. Hieronder een overzicht van uw invoer: 4, te laag. 5, te laag. 6, te laag.

Hieronder een voorbeeld van input/output van de applicatie bij succes. Input die de gebruiker invoert is ***schuin en vetgedrukt***.

> Voer de minimum waarde in van het raad-getal: ***5*** Voer de maximum waarde in van het raad-getal: ***35*** Hoe vaak mag er geraden worden [1-5]: ***6*** Je mag alleen een getal van 1 tot en met 5 invoeren. Hoe vaak mag er geraden worden [1-5]: ***3*** Raad het raad-getal: ***4*** Fout, te laag! Raad het raad-getal: ***5*** Fout, te laag! Raad het raad-getal: ***34***
>
> Gefeliciteerd! Hieronder een overzicht van uw invoer: 4, te laag. 5, te laag. 34, succes!

Les 3. Extra uitdaging(en)

BSA monitor

Tijdens je opleiding op de Hogeschool van Amsterdam krijg je cijfers voor de vakken en projecten die je volgt. Aan het eind van jaar 1 krijgt iedere student een bindend studieadvies (BSA). Een BSA kan negatief of positief zijn. Is het BSA positief dan is het mogelijk om de opleiding te vervolgen. Bij een negatief BSA is het niet mogelijk om de opleiding te vervolgen aan de Hogeschool van Amsterdam. Een student ontvangt een negatief BSA als hij of zij aan het einde van het eerste jaar van inschrijving minder dan 50 studiepunten heeft behaald uit de propedeuse.

Nu je al een beetje kan programmeren, kan je een applicatie schrijven waarbij je je behaalde cijfers in semester 1 invoert, waarna je een overzicht krijgt van je studievoortgang. Tevens kan de applicatie voorspellen of je een positief of negatief BSA zal krijgen.

Ieder vak/project heeft een naam en een hoeveelheid studiepunten. Deze studiepunten ontvang je wanneer je alle bij het vak behorende toetsen (tentamen, opdrachten etc.) haalt.

De volgende tabel geeft een overzicht van de vakken en projecten die studenten in semester 1 gemeenschappelijk hebben:

Vak	**Studiepunten**
Project Fasten Your Seatbelts	12
Programming	3
Databases	3
Personal Skills	2
Project Skills	2

Daarnaast heeft elke richting nog twee vakken van 3 punten in het semester. Welke vakken dit zijn voor jouw richting kun je vinden in de studiegids . (De Essential Skills vakken laten we voor nu buiten beschouwing).

Het is handig je programma stap voor stap te maken. Controleer tussentijds steeds of je programma werkt, zoals verwacht.

Stappenplan

1. Maak een nieuw Python programma genaamd "BsaMonitor".
2. Gebruik in je programma drie `list`'s, alle drie even lang:
    - Een `list` genaamd `vakNamen` waarin de namen van vakken en projecten komen. Vul de lijst met de namen van de vakken en projecten die jouw richting krijgt in semester 1.
    - Een `list` genaamd `vakPunten` waarin voor ieder vak de studiepunten van de vakken en projecten staan. Vul deze array met de juiste studiepunten, dus:

```
vakNamen[1]="Programming"
vakPunten[1]=3 # doordat het vak Programming index 1 heeft weet je nu dat 3 bij Programming hoort
```

    - Een `list` genaamd `vakCijfers`, waarin je de cijfers die je voor elk vak behaalt zult opslaan.
3. Gebruik een loop om alle vakken af te gaan. Vraag de gebruiker per vak om het behaalde of verwachte cijfer in te voeren en sla de ingevoerde cijfers op in de `list` `vakCijfers`. Zorg ervoor dat de gebruiker alleen geldige cijfers (tussen de 1.0 en de 10.0) kan invoeren door de invoer te controleren en opnieuw om het cijfer te vragen, als de invoer niet geldig is.
4. Nadat de gebruiker de cijfers heeft ingevoerd, kan het programma bepalen hoeveel studiepunten je hebt gehaald voor ieder vak/project, gebaseerd op het aantal studiepunten dat je kan verdienen en het cijfer dat je hebt gehaald (of hoopt te halen). Je krijgt de bij het vak/project behorende studiepunten als je cijfer groter of gelijk is aan 5,5.
5. Gebruik weer een loop om alle vakken af te gaan:
    - Tel de daadwerkelijk behaalde studiepunten voor de vakken met een voldoende cijfer op.
    - Tel de maximaal te behalen studiepunten voor alle vakken op.
    - Toon voor elk vak de resultaten op het scherm. Gebruik `.format()` om deze netjes uit te lijnen. Zie https://www.w3schools.com/python/ref_string_format.asp voor voorbeelden.
6. Toon tenslotte het aantal behaalde studiepunten. Als het aantal behaalde studiepunten kleiner is dan 5/6 van het totaal, dan moet de volgende melding worden afgedrukt: "PAS OP: je ligt op schema voor een negatief BSA!".
7. Voldoet je programma aan de code conventions?

Voorbeeld input en output

Hieronder een voorbeeld van input/output van de applicatie voor de richting CS. Input die de gebruiker invoert is ***schuin en vetgedrukt***.

> Voer behaalde cijfers in: Fasten Your Seatbelts: ***7.5*** Programming: ***10.0*** Databases: ***5.5*** Personal Skills: ***7*** Project Skills: ***5*** Infrastructure: ***8*** Network Engineering 1: ***5.4*** Vak/project: Fasten Your Seatbelts Cijfer: 7.5 Behaalde punten: 12 Vak/project: Programming Cijfer: 10.0 Behaalde punten: 3 Vak/project: Databases Cijfer: 5.5 Behaalde punten: 3 Vak/project: Personal Skills Cijfer: 7.0 Behaalde punten: 2 Vak/project: Project Skills Cijfer: 5.0 Behaalde punten: 0 Vak/project: Infrastructure Cijfer: 8.0 Behaalde punten: 3 Vak/project: Network Engineering 1 Cijfer: 5.4 Behaalde punten: 0
>
> Totaal behaalde studiepunten: 23/28 PAS OP: je ligt op schema voor een negatief BSA!

Geneste list (Geen tentamenstof maar wel leuk!)

We hebben gezien dat de elementen in een lijst verschillende datatypes kunnen hebben: `list`'s van `string`'s', lijsten van `int`'s', etc. We kunnen ook `list`'s van `list`'s maken!

Een boter, kaas en eierenspel kun je weergeven als een speelbord met 3 rijen en 3 kolommen.

X	**O**	**O**
X	**X**	**X**
O	**O**	**X**

We kunnen dit speelveld in Python weergeven door een lijst te maken, met daarin voor elke rij weer een lijst:

```
speelveld = [["X","O","O"],["X", "X", "X"], ["O", "O", "X"]]
```

Zowel de rijen als de kolommen worden vanaf 0 geteld. Willen we nu bijvoorbeeld weten, wat er in het vakje linksboven staat, dan vragen we de waarde van de lijst in rij 0 en kolom 0 op:

```
speelveld[0][0] # geeft de string "X" op
```

1. Vraag, van het bovenstaande speelveld, de waarde van het vakje in de onderste rij en de middelste kolom op. Dit zou een "O" moeten zijn.
2. Controleer je resultaat door dit af te drukken.

Les 4

Oefententamen

Je zult deze Les een oefententamen maken. Dit oefententamen zal in de les worden besproken.

Les 5

Les 5. Oefeningen

Oefening 1. functies

1. Schrijf een functie die het maximum van drie getallen teruggeeft. Test je functie door deze aan te roepen met verschillende argumenten
2. Schrijf een functie genaamd `venusJaar()` die als input je geboortejaar neemt en je leeftijd in Venus-jaren retourneert.

Les 5. Practicumopdracht

Gezondheidscheck

Schrijf een applicatie om een gezondheidstest uit te voeren voor een volwassen gebruiker. De applicatie vraagt de gebruik om drie getallen: de hartslag (in slagen per minuut), de lichaamstemperatuur (in graden Celsius) en de bovendruk (in mm Hg). Voor ieder getal moet de applicatie aangeven of de gebruiker gezond is. Gemiddeld genomen heeft een gezond persoon de volgende waarden:

- hartslag tussen 55 en 90 slagen per minuut
- lichaamstemperatuur tussen 36.3 en 37.5 graden Celsius
- bovendruk tussen 100 en 140 mm Hg

De grenzen doen mee. Dus tussen 55 en 90 betekent: vanaf 55 tot en met 90.

1. Schrijf een functie om te controleren of een bepaalde waarde tussen een minimum en een maximum in zit. De functie moet `True` teruggeven als de waarde tussen het minimum en het maximum zit en `False` als dit niet zo is.
2. Voordat je verdergaat met het schrijven van de rest van de applicatie, test de functie door deze met verschillende argumenten aan te roepen en het resultaat te printen. Dit kan bijvoorbeeld met de volgende code:

   ```
   print("waarde=3, min=3, max=8, isTussen=", isTussen(3, 3, 8))
   print("waarde=1.99, min=2, max=4, isTussen=", isTussen(1.99, 2,4))
   print("waarde=8.21, min=3, max=8.21, isTussen=", isTussen(8.21, 3, 8.21))
   print("waarde=8.22, min=3, max=8.21, isTussen=", isTussen(8.22, 3, 8.21))
   ```

3. Vervang vervolgens bovenstaande testcode door een applicatie die de gebruiker vraagt om drie getallen: de hartslag, de lichaamstemperatuur, en de bovendruk.
4. Gebruik de functie `isTussen` om alle drie de ingevoerde waarden te controleren. Per waarde moet de applicatie feedback geven of de waarde gezond is of niet.

Les 5. Extra uitdaging

Rekenmachine

Deze Les gaan we als practicumopdracht een eenvoudige rekenmachine maken. Het programma vraagt de gebruiker om een som en toont vervolgens het antwoord op het scherm. De som bestaat uit twee positieve getallen met daartussen een operator. (+, -, *, / of %). Bijvoorbeeld:

Som	Antwoord
1+1	2
3*5.2	15.6
6/2	3
8%2	0

Stappenplan

1. Schrijf een functie om te controleren of een bepaalde string een geldige operator bevat. De geldige operatoren zijn: +, -, *, / en %. Deze functie geeft de gevonden operator terug, of 'error' als er geen geldige operator wordt gevonden.
2. Test je functie voordat je verdergaat, door je programma een tekstregel aan de gebruiker te laten vragen, je functie aan te roepen met deze tekstregel als argument, en het resultaat af te drukken.
3. Als alles werkt, ga je verder. Heeft de gebruiker geen geldige operator ingevoerd, toon dan een foutmelding. Als de gebruiker wel een geldige operator heeft ingevoerd, willen we weten op welke positie in de ingevoerde string deze operator staat. Schrijf een tweede functie, die als argumenten een string en de te vinden operator heeft. Deze functie geeft de positie van de operator in de `string` als `int` terug. Je kunt de positie vinden door de `string` van links naar rechts karakter voor karakter af te gaan, totdat de operator is gevonden. Ook nu: test voor je verdergaat.
4. Nu we weten op welke positie van de ingevoerde `string` de operator staat, kunnen we de twee getallen uit de som halen door de ingevoerde `string` te slicen. Alles voor de operator is getal 1 en alles na de operator is getal 2. Vergeet niet de getallen om te zetten naar `floats`.
5. Maak een derde en laatste functie die de som uitrekent. Argumenten zijn de twee getallen en de operator. Return value is de uitkomst van de som.
6. Druk het resultaat van de som af.
7. Het is natuurlijk niet zo handig als de gebruiker de applicatie voor elke som opnieuw moet starten. Pas daarom je programma zo aan, dat de gebruiker herhaaldelijk een som kan invoeren, totdat hij/zij als som de tekst 'stop' invoert.
8. Voldoet je programma aan de code conventions?

N.B.: naarmate je meer leert over Python, zul je merken dat er manieren zijn om deze opdracht efficiënter uit te voeren. Maar door de opdracht op deze manier te maken, leren we meer over het werken met functies en strings!

Les 6

Les 6. Oefeningen

Oefening 1. String methods

Bij de onderstaande opdrachten zullen we strings manipuleren met bestaande Python methods. Bekijk eerst in de online literatuur, welke `string` methods er zoal beschikbaar zijn.

1. Gegeven is de string:

   ```
   mijnString = "Ik kan al aardig programmeren!"
   ```

   Druk een versie van deze string af, waarbij elk woord begint met een hoofdletter.
2. Tel hoe vaak de letter 'a' voorkomt in bovenstaande string en druk het resultaat af.
3. Kijk op welke positie de letter 'a' voor het eerst voorkomt in bovenstaande string en druk deze positie af.
4. Controleer of de string alleen alfabetische tekens bevat en druk het resultaat af.

Oefening 2. list methods

Bij de onderstaande oefeningen zullen we `list`s manipuleren met bestaande Python methods. Bekijk eerst in de online literatuur, welke list methods er zoal beschikbaar zijn.

1. Gegeven is de list:

```
mijnLijst=["Amsterdam", "Rotterdam", "Utrecht", "Groningen"]
```

2. Sorteer de aangevulde lijst en druk wederom de lijst af..
3. Draai de volgorde van de lijst om. Druk wederom de lijst af ter controle.
4. Verwijder Rotterdam uit de lijst en druk de lijst af.

Les 6. Practicumopdracht

Tel het zoekgetal

Het doel van deze opdracht is om een applicatie te schrijven die een rij getallen genereert en vervolgens telt hoe vaak een bepaald getal voorkomt. Dit aantal moet worden afgedrukt, samen met het percentage dat dit aantal is van het totale aantal getallen. Bij deze opdracht kun je goed gebruik maken van een aantal bestaande methods!

Stappenplan

Schrijf een applicatie die aan het doel van de opdracht voldoet. Doe dit als volgt:

1. Maak eerst een lege `list`.
2. Vraag de gebruiker om het aantal te genereren getallen. Dit moet een getal van 5 tot en met 25 zijn.
3. Voeg het gewenste aantal random getallen van 0 tot en met 9 toe aan de lijst.
4. Druk de getallen uit de list op 1 regel af, gescheiden door komma's. Hint: je kunt aan de print functie een argument genaamd `end` meegeven. Let op: achter het laatste getal staat geen komma.
5. Vraag de gebruiker om het te zoeken getal. Dit is een geheel getal van 0 tot en met 9. (Bij een perfecte uitwerking van deze opdracht test je programma op foutieve invoer en vraagt indien nodig herhaaldelijk opnieuw om de gevraagde invoer.)
6. Tel hoe vaak het te zoeken getal voorkomt.
7. Druk dit aantal af.
8. Bereken welk percentage dit aantal van het totale aantal is en druk dit af, afgerond op 1 decimaal achter de komma.
9. Voldoet je programma aan de code conventions? Laat je programma via de conventions checklist nalopen door je programming-maatje.
10. Heb je je programma verbeterd? Upload dan de source, dat is het `.py` bestand, online.

Voorbeeld input en output

Hieronder een voorbeeld van input/output van de applicatie bij geen succes. Input die de gebruiker invoert is ***schuin en vetgedrukt***.

> Hoe groot moet de lijst getallen zijn [5 - 25]? ***4*** De grootte moet tussen 5 en 25 liggen! Doe nog een poging. Hoe groot moet de lijst getallen zijn [5 - 25]? ***20*** 8, 1, 8, 2, 7, 6, 9, 5, 4, 3, 0, 6, 0, 2, 9, 2, 2, 2, 0, 6
>
> Welk getal moet ik zoeken (0..9)? ***-1*** Het zoekgetal moet tussen 0 en 9 liggen! Doe nog een poging.
>
> Welk getal moet ik zoeken (0..9)? ***6*** Het getal 6 komt 3 keer voor in de lijst. Dat betekent dat 15.0% van de getallen in de lijst gelijk is aan 6.

Les 6. Extra uitdaging

Verjaardag

Als laatste meesterproef maken we een programma dat onze komende verjaardagen berekent. Je zult hierbij gebruik maken van een module die we nog niet eerder behandeld hebben, namelijk de `datetime` module.

Met de `datetime` module kun je verschillende soorten objecten gebruiken, bijvoorbeeld objecten van het type: `datetime.time` voor het bijhouden van een tijdstip `datetime.date` voor het bijhouden van een datum `datetime.datetime` voor het bijhouden van een tijdstip op een bepaalde datum.

Voor het bijhouden van een verjaardag kunnen we goed gebruik maken van een `datetime.date` object. Je zult je voor het uitvoeren van deze opdracht moeten verdiepen in de werking van dit object en de bijbehorende methods. Zoek daarom eerst online naar de documentatie!

Het te schrijven programma vraagt allereerst je geboortedatum en vraagt daarna, hoeveel komende verjaardagen je wilt berekenen.

Het programma geeft vervolgens voor het huidige jaar, en de komende jaren, aan hoe oud je wordt en op welke dag van de Les je verjaardag valt.

Stappenplan

1. Schrijf een functie getDate. Aan deze functie wordt een string als argument meegegeven. Bij aanroepen van de functie toont de functie de meegegeven string als prompt en vervolgens moet de gebruiker een string invoeren.
2. Ervan uitgaande dat de ingevoerde string een geldige datum is, bevat deze drie cijfers, gescheiden door een liggend streepje, -. Het eerste cijfer is de dag, dan de maand en dan het jaar. Je kunt dus een method gebruiken om de string op te splitsen in een lijst met drie elementen: dag, maand en jaar.
3. Met de elementen van je lijst als argumenten, kun je nu een nieuw datetime.date object aanmaken, dat de geboortedatum bevat.
4. De functie geeft dit datetime.date object terug als return value.
5. Test je functie voordat je verdergaat, door deze aan te roepen en het resultaat af te drukken.
6. Als alles werkt, ga je verder. Vraag de gebruiker hoeveel verjaardagen deze wil berekenen.
7. Maak nu een nieuw datetime.date object met de huidige (systeem) datum. Je kunt vervolgens verschillende eigenschappen van je nieuwe object opvragen, waaronder het jaar.
8. Bereken, op basis van het huidige jaar en het geboortejaar, hoe oud de gebruiker dit jaar wordt.
9. Bepaal op welke Lesdag dit jaar de verjaardag valt. Hint: maak eerst een nieuw datetime.date object met als datum de verjaardag in het betreffende jaar. Deze datum kun je vervolgens precies zo afdrukken als gewenst met de methode `.strftime()`.
10. Als dit allemaal goed gaat, herhaal dit dan voor de komende jaren.
11. Als laatste kun je je functie `getDate` uitbreiden met een aantal extra controles:
    - bevat de ingevoerde string inderdaad precies twee liggende streepjes (-)?
    - bestaan de drie substrings met datum, maand en jaar uit alleen maar decimale cijfers?
    - ligt de dag wel tussen 1 en 31, de maand tussen 1 en 12 en het jaar tussen (zeg) 1900 en 2100?
    - andere controles die je nog kunt bedenken?

    Zo niet, geef dan een foutmelding en vraag de gebruiker opnieuw om input.
12. Voldoet je programma aan de code conventions?

Voorbeeld input en output

Hieronder een voorbeeld van input/output van de applicatie bij geen succes. Input die de gebruiker invoert is ***schuin en vetgedrukt***.

> Enter your date of birth (dd-mm-yyyy): ***1-13-1990*** Incorrect input, please try again
>
> Enter your date of birth (dd-mm-yyyy): ***1-12-1990*** Calculate how many birthdays? **3**
>
> This year you will be 29 years old. Your birthday will be on a Sunday .
>
> In 2020 you will be 30 years old. Your birthday will be on a Tuesday .
>
> In 2021 you will be 31 years old. Your birthday will be on a Wednesday .

Les 7

Oefententamen

Je zult deze Les een oefententamen maken. Dit oefententamen zal in de les worden besproken.