

Input en output

Een computerprogramma is maar nuttig als het met de buitenwereld in verbinding staat. Denk maar aan een webbrowser (Firefox, Chrome), een game (Minecraft, The Sims), maar ook computerprogramma's die hoofdzakelijk berekeningen doen (het getal π tot miljoenen cijfers na de komma berekenen) moeten de resultaten van deze berekeningen uiteindelijk aan de wereld kenbaar maken.

Input zijn die gegevens die binnenkomen in een programma, output is hetgeen dat weggaat uit een programma. Als afkorting gebruiken we vaak 'I/O' of 'IO'.

 Hoeveel voorbeelden van IO kan je bedenken? Welke zijn input, welke zijn output?

De terminal

Een van de oudste, en daarom best ondersteunde en misschien wel eenvoudigst te programmeren vorm van IO is de interactieve terminal of console. De enige interactiemogelijkheden zijn via tekst: invoer via het toetsenbord en output via het scherm, in een sober venster.

Het eenvoudigste Python-programma doet dan ook niet veel meer dan een boodschap tonen via de terminal:


```
print("Hello world!")
```


We krijgen input van de gebruiker met de input-functie:


```
naam = input("Wat is jouw naam? ")
print(f"Dag {naam}!")
```

Het woord 'naam' in de code hierboven is een voorbeeld van een variabele. We gebruiken variabelen om waarden op te slaan om later te gebruiken in code.

 Wat gebeurt er als je de twee lijnen code van volgorde wisselt?

 Het gelijkheidsteken (=) betekent in Python iets anders dan in rekenen. Wat is het verschil?

 Waarom staat er een spatie na het vraagteken in de eerste lijn code? Hoe voelt het programma aan als je deze spatie weglaat? Het verschil tussen een werkend en een goed programma zit vaak in aandacht voor kleine details!

 Waarom staat de 'f' in het print-commando? Wat gebeurt er als je die er niet zet?

De stukken tekst tussen quotes (") noemen we strings. Je kan strings zowel met dubbele als enkele quotes schrijven. Dat is vooral handig als je de andere quote nodig hebt binnen je string.

```
print("'t Is mooi weer vandaag!")
print('Goedemorgen', zei de man.')
```

Escapes

Er is echter een oplossing als je ze allebei zou nodig hebben:

```
print('Goedemorgen,' zei de man, '\t is mooi weer vandaag.')
```

Door een backslash (\) voor een quote te zetten escaperen we deze, waardoor Python de string niet afsluit. Zoals je ziet wordt dat snel nogal ingewikkeld.

Rekenen met Python

Computers zijn in de eerste plaats uitstekende rekenmachines! Laat ons eens kijken. Tip: doe dit in een REPL of worksheet, dan hoef je niet telkens 'print' te typen!

```
print(1+1)
print(8 - 10)
print(7*8)
print(2**10)
```

Getallen die zeer groot of klein zijn (klein = kommagetal zeer dicht bij 0) kan je met de exponent-notatie voorstellen:

```
print(1e10)
print(15e-5)
print(1.5e-4)
print(2e20*2e-19)
```

 Wat denk je dat volgende berekening teruggeeft?


```
print(5+2*3)
```

 Wat is het verband tussen volgende drie operatoren?

```
print(7/3)
print(7//3)
print(7%3)
```

Een aantal functies die je minder vaak nodig hebt, kan je in de math-module vinden. Modules zijn een manier van Python om uitbreidingen op de programmeertaal te voorzien. Daar hebben we het later nog wel over, het belangrijkste nu is dat je de identifiers uit modules moet importeren voor je ze kan gebruiken, zoals in het voorbeeld hieronder.

```
from math import sqrt, pi
print(sqrt(2))
print(pi)
```

 Wat gebeurt er als je het importeren niet doet? Wat gebeurt er als je importeert nadat je de sqrt-functie gebruikt?

Gehele getallen, en floating-point getallen

Computers werken vaak met benaderingen! Python zal gehele getallen exact voorstellen zolang het kan, maar komma-getallen zijn erg vaak incorrect, en beperkt in waarde-bereik. Het voordeel van deze keuze is dat computers zeer snel kunnen rekenen met deze kommagetallen. Voorbeelden waar het mis gaat:

```
print(0.1-0.1)
print(0.1+0.1-0.1-0.1)
print(0.1+0.1+0.1-0.1-0.1-0.1)

print(2**10000)
print(2.0**10000)
print(2**10000/2)
```

Meestal hoef je hier niet te veel rekening mee te houden, maar tijdens het programmeren van games zie je deze effecten soms optreden. Bijvoorbeeld: een game waarbij Mario, als je op het pijltje naar rechts drukt, een snelheid van 0.1 km/u verkrijgt, en als je op het pijltje naar links drukt, een snelheid van 0.1 km/u verliest. Het wordt snel onmogelijk om Mario nog exact stil te laten staan!

 Kan je een manier bedenken om dit probleem op te lossen?

Datatypes

In Python kan je stukken tekst aan elkaar plakken:

```
print("Stuk" + "jes")
```

en je kan getallen optellen:

```
print(4 + 3)
```

Wat gebeurt er als je tekst en getallen optelt?

```
pagina = 1
aantal_paginas = 5
percentage = pagina/aantal_paginas * 100
print("Dit is pagina " + pagina + " van " + aantal_paginas + " (" + percentage + "%")
```

Of gaat rekenen met tekst?

```
geboortejaar = input("In welk jaar ben je geboren? ")
leeftijd = 2019 - geboortejaar
print(f"Je bent nu ongeveer {leeftijd} jaar oud!")
```

Elke waarde in Python heeft een type. De types die je al kent zijn str (van 'string', voor tekst), int (van 'integer', voor gehele getallen) en float (voor floating-point getallen). Je kan het type van een waarde opvragen met de type-functie:

```
print(type("Tekst"))
print(type(2019))
```

Programmeertalen gaan soms heel anders om met datatypes. In Javascript kan je bijvoorbeeld wel strings en getallen optellen, en in C# moet je op voorhand aangeven welk datatype een variabele heeft.

Gelukkig kan je waardes omzetten naar andere datatypes!


```
print(int("10") + int("-1"))
print(str(10) + str(-1))
```


 Kan je de vorige programma's werkende krijgen met datatype-omzettingen?

Nog een weetje.. Je kan strings en getallen niet optellen, maar je kan ze wel vermenigvuldigen!


```
print(10 * "-")
print(f"De bel doet tr{'i' * 20}ng")
```

Oefeningen


 Schrijf een programma om de oppervlakte van een cirkel te berekenen, gegeven de straal. Ken je nog andere formule's die je in programmavorm zou kunnen gieten?


 Schrijf een programma dat 5 getallen vraagt, en een histogram van deze getallen maakt. Dat ziet er dan bijvoorbeeld zo uit:


```
Getal 1? 1
Getal 2? 3
Getal 3? 8
Getal 4? 6
Getal 5? 4
1 #
3 ###
8 #####
6 #####
4 ####
```


 Wat gebeurt er als een van de getallen groter is dan 10? Kan je dit oplossen? Misschien kan je inspiratie halen uit volgend voorbeeld met extra functionaliteit van format-strings:


```
aantal_visjes = 4
print(f"{aantal_visjes:3} kleine visjes, die zwommen naar de zee")
aantal_visjes = 37
print(f"{aantal_visjes:3} kleine visjes, die zwommen naar de zee")
aantal_visjes = 189
print(f"{aantal_visjes:3} kleine visjes, die zwommen naar de zee")
```

 ⚡ Kan je dit oplossen voor nog grotere getallen? Wat als je niet op voorhand weet wat het grootste getal is dat een gebruiker gaat ingeven? Hoe ziet de grafiek er uit als die niet meer op het scherm past? Hoe zou je dat kunnen oplossen?

 Schrijf een programma dat de geboortedatum van de gebruiker vraagt en de leeftijd van de gebruiker in dagen berekent. Het is voorlopig het eenvoudigst als je de geboortedatum met drie input-stappen opvraagt: de dag, de maand en het jaar. Het is ook het makkelijkst als je de datum van vandaag 'hardcode', dus rechtstreeks in het programma stopt.

 Kan je de leeftijd in seconden teruggeven?

 ⚡ Vergelijk de antwoorden die je programma teruggeeft met een website met een soortgelijke berekening. Zijn er verschillen? Zo ja, vanwaar komen de verschillen? Kan je deze oplossen?

 ⚡ Wat is het nadeel van hardcoden? Wat is het voordeel van hardcoden? Je computer weet wat de datum vandaag is, en je kan deze ophalen en gebruiken voor berekeningen:

```
from datetime import datetime
print(datetime.now())
print(datetime.now().day)
```