Lesbrief 1: Print, variabelen en errors

De print functie

Met Python is het mogelijk om tekst en getallen op het scherm te tonen. Dat doe je met de functie `print`. Deze functie "print" gewoon de informatie die je wilt op het scherm. Probeer het maar uit met de volgende codes, en voel je altijd vrij om een beetje rond te neuzen en spelen met de functies. Hier leer je namelijk het meeste van!

```
print ("Dit komt dus op het scherm terecht.")
print ("We gaan nu een paar sommetjes doen:")
print (230-21)
print (78-2+6)
print (78-2*6)
```

Je kunt de bovenstaande code ook hier uitproberen: Print sommetjes

Zoals je ziet kun je ook berekeningen tussen de haakjes van de `print()` functie zetten. Let er wel op dat de voorrangsregels van de wiskunde ook gelden voor programmeertalen: de * en de / gaan dus vóór de + en de -.

Opgave 1
Schrijf een programma dat met de `print()` functie “Hello, world” op het scherm zet via trinket.

Oké, nu je iets kan printen op het scherm gaan we hier een beetje mee spelen! Er bestaat iets als een string. Een string in python is een rij van tekens (characters). Een teken is een letter (zoals ‘a’, ‘B’), nummer (‘0’, ‘2’), symbool (‘#’, ‘^’, ‘}’), enz. en kan gebruikt worden om allerlei soorten informatie in op te slaan. Vaak is dit dus gewoon een stukje tekst wat iemand op het scherm wil printen.

‘string’ betekent in het Engels touw of draad. Het is dus een touw met karakters er aan!

Bij de vorige opdracht heb je steeds een aparte string geprint op het scherm. Nu bestaat er ook iets als string concatenation, waarin je twee strings samen kan voegen. Concatenation is een Engels woord voor “aaneenschakeling”, en in python is dit ontzettend makkelijk: Je plakt twee string aan elkaar vast met een +! Probeer maar:

```
print ("Hallo, dit is deel een!" + " En dit is deel twee!")
print ("Zo kun je dus " + "meerdere verschillende " + "strings aan
elkaar plakken!")
```

Deze code kan je natuurlijk ook weer hier uitproberen:
[Strings aan elkaar]

Opgave 2
Schrijf een programma dat met de print functie door middel van concatenation de komende drie zinnen op dezelfde manier als het voorgaande voorbeeld aan elkaar plakt:

Lotte heeft
pannenkoeken gegeten!

De kat van Kees
heeft een muis gevangen.

De parkiet van Bas
heeft hem de hele
nacht wakker gehouden.

Opgave 3
Print bij deze opdracht de komende drie woorden in alle volgordes die er maar zijn dus bijvoorbeeld: ‘Niezen Knikkers’. Hint, er zijn 6 verschillende volgordes.

Niezen

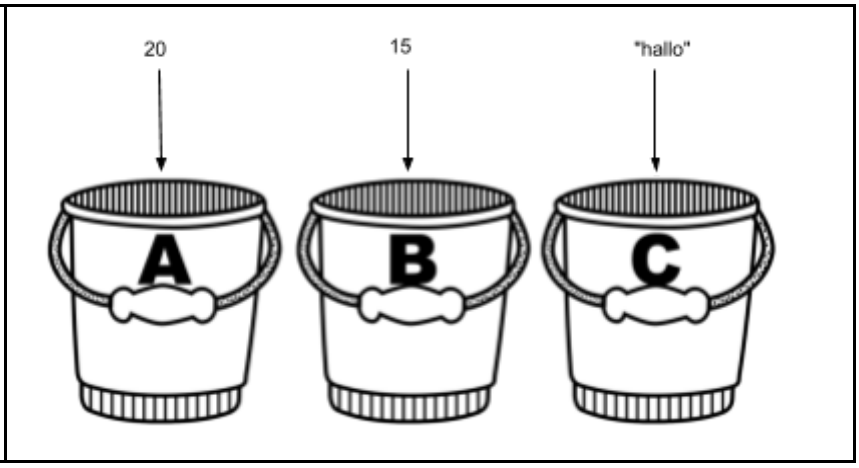
Kikkers

Blij

Variabelen

Vaak wil je in je programma tijdelijk gegevens bewaren. Dit doe je met *variabelen*. Een variabele kun je zien als een emmer met een label erop. Je kunt er iets in zetten en je kunt er iets uithalen. Hieronder zie je een stukje code met ernaast een plaatje waarop te zien is wat er gebeurt:

```
a = 20
b = 15
c = "hallo"
```



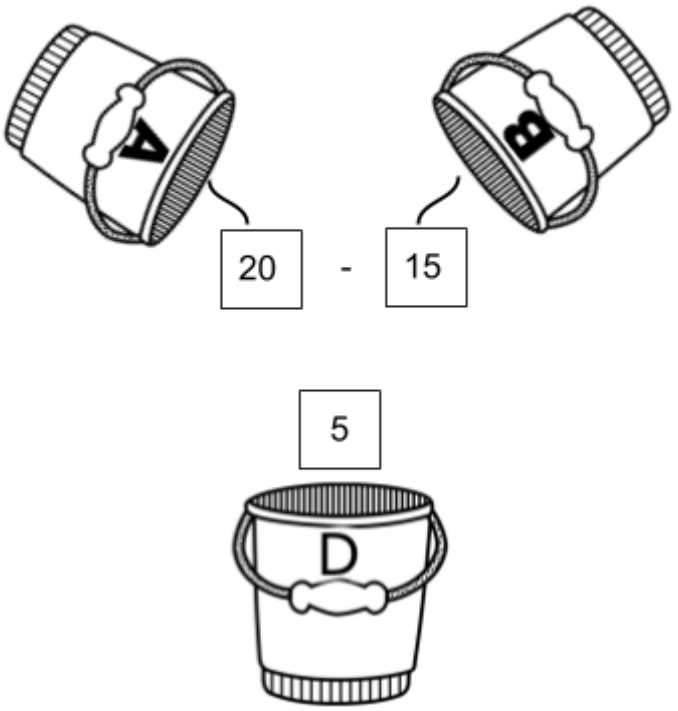
Je ziet dus dat er nu drie emmers zijn. Elke emmer heeft een naam/label en een inhoud. In het begin zijn alle emmers leeg. Maar na de drie regels van hierboven verandert de zaak:

- In de emmer met als naam *a* komt namelijk het getal 20 te staan.
- In emmer b komt 15 te staan.
- En in c komt de tekst "hallo" te staan.

We kunnen nu ook een vierde emmer genaamd d in het leven roepen. In die variabele kunnen we het verschil (min) tussen a en b plaatsen. Dat gaat zo:

```
a = 20
b = 15
c = "hallo"

d = a - b
```



We halen dus het getal 20 uit emmer A en we halen het getal 15 uit emmer b. Het verschil wordt vervolgens berekend en het antwoord komt in emmer d te staan.

Uiteindelijk zouden we ook de inhoud van de variabelen op het scherm kunnen tonen. Dat doen we natuurlijk met de functie `print`. Dit gaat zo:

```
print (a)
print (b)
print (c)
print (d)
```

Opgave 4

Schrijf een programma waarbij je door middel van variabelen de komende berekeningen doet:

8 - 5

23 * 34

21 + 8

Opgave 5

Stop de antwoorden van de vorige opdracht in drie verschillende variabelen, vermenigvuldig deze met elkaar en print het uiteindelijke antwoord.

De naamgeving van een variabele kan je altijd zelf bepalen. Dus als je berekeningen wilt doen met je favoriete getal, kan je deze in een variabele stoppen met als naam bijvoorbeeld `favoriete_getal`. Dit is handig omdat je dan makkelijk kan bijhouden wat er in de variabelen zit. Noem je variabelen dus altijd op een logische manier! Kijk maar naar het volgende voorbeeld:

```
favoriete_getal = 7
print (favoriete_getal + 1)
print (favoriete_getal)
print (favoriete_getal * 2)
favoriete_getal = 4 + 5
print (favoriete_getal)
print ("favoriete_getal")
```

Zoals je kan zien zijn de eerste drie antwoorden van het programma (dit worden ook wel outputs genoemd) gewoon een wiskunde som die de inhoud van onze variabelen gebruikt. Daarna wordt er een verandering gemaakt in de variabele `favoriete_getal`. Er wordt een andere waarde aan de variabele gegeven, namelijk 4 + 5, dus 9. Je kan na het toewijzen van de waarde van een variabelen deze dus altijd ook nog later veranderen.

Bij de laatste output zien we een beetje een instinkertje. Namelijk doordat er " " om `favoriete_getal` stonden, wordt deze gezien als wat? Ja! Een string! Het is dus heel erg belangrijk om in je achterhoofd te houden of je iets wilt printen als een string of als een variabele. Als je de " " namelijk vergeet, gaat het programma zoeken naar een variabele met die naam. En als er dus geen variabele toegewezen is kan je hierdoor een foutmelding (oftewel een *error*) krijgen.

Inleiding voor errors

Een 'error' is een foutmelding. Op errors gaan we later dieper in, maar voor nu is het handig om te weten dat je bij bijvoorbeeld de volgende code:

```
getel = 4
print (getal + 1)
```

iets kan krijgen als:

```
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
NameError: name 'getal' is not defined
```

Als je kijkt naar line 2 (regel 2) dan wordt daar een variabele met de naam getal gebruikt, terwijl deze dus niet bestaat of geen waarde toegewezen heeft gekregen. Je ziet dat er een spelfout is gemaakt in regel 1. Er staat getel in plaats van getal. Als je dat verandert verdwijnt de foutmelding.

Een error kun je bijvoorbeeld krijgen als:

- je de " " om een string ergens vergeten bent,
- je variabelenaam verkeerd hebt gespeld
- je variabele nog niet een waarde hebt gegeven

Het kan ook zijn dat je de toewijzing andersom doet, zoals:

```
4 = x
```

Dan krijg je een error die eruit ziet als:

```
Traceback (most recent call last):
  In line 1 of the code you submitted:
    4 = x
    ^
SyntaxError: can't assign to literal
```

Deze foutmelding houdt dan dus in dat je altijd eerst je variabelenaam moet noemen, en dan pas de inhoud ervan moet geven. In de geval verbeter je het dus door er van te maken:

```
x = 4
```

In de toekomst gaan we dieper in op errors en hoe je problemen kunt oplossen. Maar het is handig dat je hier alvast wat over hebt gelezen, in het geval dat je straks wat schrijft en opeens een error krijgt. Ze zien er altijd heel ingewikkeld uit, maar als je hier eenmaal handig in wordt valt het best mee! Neem gewoon rustig je tijd, en bekijk wat er staat en neem je code dan nog even door.

Opgave 6

Run de volgende codes, en los de errors op.

Errors opgave 1.1

Errors opgave 1.2

Errors opgave 1.3

Errors opgave 1.4