Start to Program: Python, Les 2

Commentaar in de code

Commentaarlijnen kan je gebruiken om opmerkingen voor jezelf in de code te zetten. Deze tekst is alleen zichtbaar in je editor. Als de pyhton-file wordt uitgevoerd zal je hier niets van merken.

Ze kan ook gebruikt worden om verschillende titels aan je code toe te voegen of om een stuk code tijdelijk op non actief te zetten. Dit zonder dat je de code moet wissen.

Een commentaarlijn wordt voorafgegaan door een #

```
# Dit is een stukje commentaar
```

Om niet alles op 1 lijn te zetten maar eerder in en blok zet je voor elke nieuwe regel opnieuw een #

```
# Dit is
# een blok
# commentaar
```

Je kan ook achter een lijn code een stuk commentaar toevoegen.

```
print('Hello World') # dit print hello world
```

Omdat bij python de insprong(aantal tabs) van de code belangrijk is plaats je de commentaar best op de zelfde hoogte als de code waar ze naar verwijst.

Variabelen

Variabelen komen in alle programmeertalen voor. Ze zijn kleine doosjes waarin je een bepaalde waarde gaat bewaren. Je kan deze dan oproepen in je code wanneer je die waarde nodig hebt. Zo kan je lange getallen of hele zinnen opslaan zodat je die niet steeds op nieuw hoeft te typen.

Stel dat je het nummer 105462872045621 veel nodig hebt in je code. Een fout kan hier makkelijk insluipen als je het getal telkens opnieuw moet ingeven. Daarom kan je het getal best opslaan in een variabele. Dit doe je als volgt.

```
mijn_getal =  105462872045621
```

- ***mijn_getal*** is de naam van de variabele.
- = zorgt ervoor dat python weet dat je iets wil opslaan.
- ***105462872045621*** is de waarde die je in de variabele wil opslaan.

Door de naam van de variabele te gebruiken roep je de waarde ervan op.

```
print(mijn_getal)
        geeft
105462872045621
```

In python moet je de variabele niet benoemen. In veel andere programmeertalen moet je het datatype(nummer, tekst,...) meegeven.

Je kan nu ook berekeningen met deze variabele uitvoeren. Of je kan de uitkomst van een berekening ineens opslaan in een variabele.

Opdracht:
Sla een getal op in een variabele.
Print deze variabele.
Print deze variabele, maar probeer met een berekening. (plus of min een ander getal)
Sla een bewerking rechtstreeks op in een variabele en print deze bewerking.

Een variabele kan ook overschreven worden. Dit doe je door later in je code een nieuwe waarde aan dezelfde naam toe te voegen.

Sla een willekeurige waarde op in een variabele.
Print deze.
Sla een andere waarde op in dezelfde variabele en print deze.
Run je programma.

Benaming

De naam van een variabele mag je helemaal zelf bepalen. Er zijn wel een aantal beperkingen. Verder zijn er een aantal afspraken zodat er in de code uniformiteit zit. Probeer ook steeds een duidelijke omschrijving als naam te kiezen.

Toegestaan	Niet toegestaan	regel
mijn_var	mijn-var	Koppelteken niet toegestaan, underscore wel
var2	2var	Nooit beginnen met een nummer
mijn_var	$mijn_var	Geen speciale tekens gebruiken buiten _
mijn_var	mijn var	Nooit meer dan 1 woord gebruiken.

Verder zijn er dus ook een aantal stijlregels.

Een variabele begint nooit met een hoofdletter.

mijn_var ipv Mijn_Var

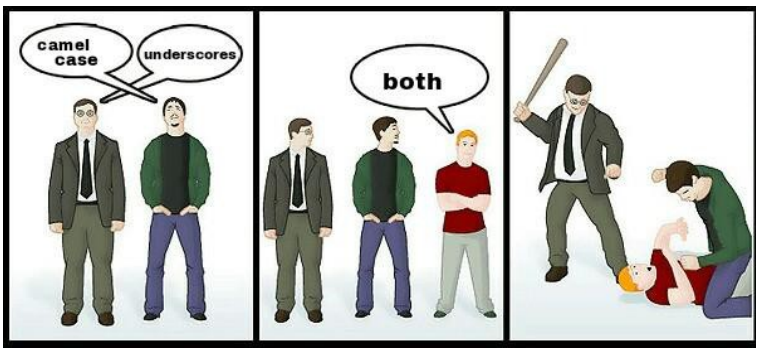
Werken met een underscore (_) wordt verkozen boven het gebruik van camelcasing. Camelcasing is wanneer elk nieuw woord met een hoofdletter begint.

mijn_var <=> mijnVar

mijn_geweldig_duidelijke_var <=> mijnGeweldigDuidelijkeVar

Werken met underscores maakt de tekst leesbaarder.
Benaming van een variabele is hoofdlettergevoelig. Maak het jezelf dus gemakkelijk en probeer hoofdletters te vermijden.

Je zal beide tegenkomen als je op zoek gaat naar voorbeelden. Er is reeds een hele tijd discussie over dit onderwerp.
Wat je ook gebruikt mix nooit de 2 manieren in 1 project!



Datatypes: String

Algemeen

Een string is de benoeming voor het datatype dat met tekst te maken heeft.
In de code duid je dit aan met aanhalingstekens. Alles wat tussen die aanhalingstekens valt is de waarde van de string.
Je mag hiervoor dubbele of enkele aanhalingstekens gebruiken. Deze geven hetzelfde resultaat.

Gebruik je nu best dubbele of enkele aanhalingstekens?
Dit hangt van de inhoud van de tekst af. Er kunnen problemen ontstaan als je enkele aanhalingstekens gebruikt en in de tekst komt ook een enkel aanhalingsteken voor.

print('deze avond')
Wat is het resultaat?

print(' 's avonds')
Wat is hier het resultaat?

In het 2de geval zou het beter zijn om met dubbele aanhalingstekens te werken.
Dan omzeil je het probleem. Er zijn ook andere manieren om dit op te lossen. Deze komen later aan bod.

Je kan ook tekst samenvoegen. Dit doe je op eenzelfde manier zoals je getallen zou optellen, met het plusteken.

print('hello' + 'world')
Wat merk je op?

print('hello ' + 'world')
Wat is het verschil met de voorgaande code?

print('hello' + 7)
Werkt dit ook?

print('hello' * 9)
Geeft dan weer....?

Dit zijn enkele basis eigenschappen om met het datatype string om te gaan.

Je kan je tekst ook in eerst in een variabele opslaan en dan de variabele aanroepen om te printen.

Sla een willekeurige string op in een variabele.
Print nu deze variabele.

Tekst opmaak

In veel gevallen wil je meer controle over je string/tekst. Dit vooral om het leesbaarder te maken. Hiervoor gebruik je leestekens, spaties, witregels....
Omdat Python werkt met spaties, tabs en witregels moet je hiervoor een specifieke manier van werken hanteren.

Aanhalingstekens gebruiken

Als je toch verschillende aanhalingstekens nodig hebt in je opbouw. Dan kan je de speciale tekens “escapen”. Dit wil zeggen dat je deze uit zijn normale functie gaat halen om ze als gewone leestekens te gebruiken.
Dit doe je door gebruik te maken van een backslash \.
Let op! geen gewone slash! /

```
print(“Leo zei: “Ik zal 's avonds naar de les komen.””)
```

Maak gebruik van de backslash op de juiste plaats om de tekstuele aanhalingstekens te “escapen”.

Tekst op meerdere lijnen

Als je tekst op meerdere lijnen kan je niet meer gewoon de aanhalingstekens gebruiken. Je zal merken dat er dan een error verschijnt.
Je kan dit oplossen door de tekst tussen 3 paar aanhalingstekens te zetten.

```
print('''Deze tekst
loopt over
verschillende lijnen.''')
```

Tekst letterlijk gebruiken

Als je tekst letterlijk wil printen met alle symbolen die erin staan. Dus alle quotes en alle backslashes, dan moet je een r plaatsen voor het begin van je string.
BUITEN de eerste quotes.

```
print(r“Leo zei: \“Ik zal \'s avonds naar de les komen.\””)
```

String functies

Zoals al aangehaald heeft Python een uitgebreide standaard bibliotheek meer voorgeprogrammeerde functies. Zo zijn er voor het datatype string ook een hele hoop functies. Deze functies kunnen behulpzaam zijn bij het opmaken van de tekst, maar ze kunnen je ook een hoop informatie over de string verschaffen.

Hoofdletters en kleine letters

Je kan ervoor zorgen dat de tekst altijd in hoofdletters of steeds in kleine letters afprint wordt. Dit is handig wanneer het ingeven van tekst in het één of het ander moet gebeuren. Mensen maken fouten, mensen zijn slordig. Computers zijn dit niet. Als je hen vertelt dat alles in hoofdletters moet als dit belangrijk is zal het ook altijd zo gebeuren.

Voorbeeld: Een persoon logt in in de databank met zijn naam en gebruikt de ene keer geen hoofdletter, de andere keer wel. Door alles in de databank in kleine letters op te slaan en alle input naar de databank naar kleine letters te forceren krijg je minder fouten.

De functies die je hiervoor gebruikt zet je rond je tekst.

```
print(str.upper('hello world'))
print(str.lower('HELLO WORLD'))
```

Je kan dit ook rechtstreeks aan een variabele koppelen.
Dan plaats je de var niet tussen de (), maar vervang str in het begin van de functie door de naam van de variabele.

```
Sla hello world en HELLO WORLD op in een var

print de ene var in hoofdletters,
de andere var in kleine letters.
```

Je kan ook elk woord apart met een hoofdletter laten beginnen.
Hiervoor gebruik je de functie: str.title()

```
Probeer dit uit op je aangemaakte variabelen.
Wat valt er op?
```

Tekst samenvoegen, splitsen en vervangen

join()

Je kan nu de bestaande tekst al minimaal aanpassen. Er zijn ook methodes om de boodschap/tekst zelf aan te passen.

Tekst toevoegen doe je met de methode str.join() en die gaat als volgt te werk:

```
"toe te voegen tekst".join(mijn_var)
```

Sla een willekeurige zin op in en var.
Print deze zin, maar voeg een extra spatie toe.
Wat valt er op? Waar komt de extra spatie?

De methode str.join() kan ook gebruikt worden om tekst volledig om te draaien.
Verwijder de spatie tussen de aanhalingstekens voor join.
Plaats daarna de methode reversed() tussen de haken van de join methode.
De var plaats je dan tussen de haken van de reversed() methode.

Je merkt wel op dat dit niet zo heel handig is als er tussen elke letter dan een toevoeging van de nieuwe string is.
Later als je met het datatype lijsten aan de slag gaat wordt het nut wel duidelijk.

split()

Je kan tekst ook uit elkaar gaan halen. Dit doe je met split().
Hier plaats je de var die je wil opsplitsen VOOR de methode split.

Splits je var op met de methode methode str.split().

Probeer dit met een string dat bestaat uit 1 woord en een string met meerdere woorden.

Wat merk je op?

Je kan ook splitsen na een bepaalde letter of teken. Hiervoor vul je de gewenste letter in tussen aanhalingstekens en tussen de haken van de split methode.

Sla de volgende zin op in een var.
'De kat krapt de krollen van de trap.'
Splits deze zin bij de letter a.

replace()

Als je goed hebt opgelet merkt je dat in de vorige oefening een fout stond! Geen programmeer fout, maar een schrijffout.

Het zou dus best zijn als je dit woord kan vervangen.
Hiervoor kan je de methode str.replace() gebruiken.

Je roept eerst de string of var op die je wil aanpassen. Tussen de haken van de replace() methode geeft je dan eerst het woord op dat je wil vervangen. Dan een komma en tenslotte het nieuwe woord.

Vervang de schrijffout uit de zin van de vorige oefening.

SAMENGEVAT

str.upper()	▫	alles in hoofdletters
str.lower()	▫	alles in kleine letters
str.title()	▫	eerste letter van elk woord met een hoofdletter
str.join()	▫	tekst tussenvoegen bij elke letter
str.join(reversed)	▫	tekst omdraaien
str.split()	▫	een lijst maken van de verschillende woorden i.e. String
str.split('ltr')	▫	splitsen na een bepaalde letter
str.replace()	▫	tekst vervangen in een string

OPDRACHT String:
Maak verschillende variabelen aan. Kies een duidelijke benaming!

2 variabelen met je voor en achternaam apart.
Een variabele met een string van verschillende lijnen met je adres.

Print je voornaam volledig in Hoofdletters en je achternaam in kleine letters.
Voeg deze samen in een nieuwe var naam waarin dus je voor en achternaam zit.
Print die nieuwe var af waarbij elk woord begint met een hoofdletter.
Print je voornaam 5.
Print je volledige naam achterstevoren.

Voeg de letter p toe na elke letter in je achternaam.
Maak een lijst van alle woorden in je adres.
Vervang de gemeente in je adres door het woord fabeltjesland.

Lengte van een string tellen

Er zijn ook een hele hoop methode die je meer informatie kunnen geven over je string. De lengte van een string bepalen bijvoorbeeld.
Dit doe je zo:

```
print(len(mijn_var))
```

Let wel op! Deze methode zal ALLES in je string tellen! Dus ook de spaties, leestekens,....

Bepaalde positie van eens string

De positie van een teken wordt ook de index genoemd.

LET OP! Bij programmeren telt het getal 0 wel degelijk mee. De index begint dus bij 0!

indexering van hello world

```
h e l l o   w o r l d
0 1 2 3 4  5  6 7 8 9 10
```

Je kan ook omgekeerd te werk gaan. Door op te vragen welk teken op een bepaalde index staat.

```
print(hw[3])
Geeft l weer.
```

Als je een teken nodig hebt dat ergens achteraan de string staat kan je ook negatief te werk gaan. Hier begint de index niet met 0 maar ineens met -1.

```
 h e  l  l  o     w o r l d
-11-10-9 -8 -7 -6  -5 -4 -3 -2 -1
```

Tel hoeveel tekens je naam, voor- en achternaam, bevat.
Print het 5de teken in je naam en het derde laatste teken.