Start to Program: Python, Les 4

Number: methodes

Buiten de standaard wiskundige berekeningen kan je ook vooringestelde methodes gebruiken.

Absolute waarde

Soms heb je een berekening waarbij je een negatief getal als uitkomst krijgt. Toch is het voor de verdere berekening niet handig om met negatieve getallen te gaan werken. Of is de uitkomst die verkregen is gewoon niet logisch als een negatief getal.

Je kan dan die uitkomst als ze negatief is gaan omzetten naar een positief getal door een minteken ervoor te plaatsen.
MAAR je kan niet altijd eerst gaan controleren wat juist de uitkomst is.
Daarom kan je die uitkomst altijd forceren naar de absolute waarde met een methode.

```
abs(mijn_getal)
```

combinatie delen en modulo

Delen en de modulo(restwaarde) zijn berekeningen die nauw aan elkaar verwant zijn.
Soms is het ook handig om beide tegelijk te verkrijgen.
Dit kan met:

```
divmod(getal1,getal2)
```

Hier krijg je 2 waarden als uitkomst.
Zoals meestal wil je de waarde opslaan in de variabelen. Liefst elke waarde apart.
Dit doe je als volgt:

```
a,b = divmod(getal1, getal2)

a zal de afgeronde deling geven.
b zal de modulo bevatten.
```

machtsverheffing

Machtsverheffing heeft ook een kortere manier door het gebruik van **, maar kan ook worden verkregen met de methode:

```
pow(getal1,getal2)
```

afronden

De methode round() gaat je getallen afronden.
Let op! Dit wil niet zeggen dat je getal een integer wordt!

Je kan ook meerdere waarden meegeven in de round() methode. Zo kan je bepalen tot hoeveel cijfers na de komma wordt afgerond.

```
round(getal,aantal_cijfers_na_de_komma)
```

Opdracht:
Maak een variabele voor afgelegde kilometers met waarde 5 en nog af te leggen kilometers met waarde 15. Bereken het verschil tussen afgelegd en nog af te leggen. Zorg ervoor dat dit steeds een positief getal is.

Opgegeven:
Totaal aantal tekens van mijn boek is 8000.
Op formaat A passen 300 tekens per pagina.
Op formaat B passen 250 tekens per pagina.
Welk formaat past het beste voor mijn project?
Wanneer heb ik het de best gevulde pagina's?

Opgegeven:
De totale rekening is 82,65.
Je bent met 4 personen.
Hoeveel moet iedereen betalen? Hou er rekening mee dat het minste dat iemand kan betalen 1 eurocent is.

Datatype: boolean

Om complexere structuren te maken is het datatype boolean essentieel.
De naam is afkomstig van de bedenker, de wiskundige George Boole.

De waarden die opgeslagen kunnen worden in dit datatype zijn heel beperkt.
Je gaat steeds de waarde *True* of *False* verkrijgen.
Let op! Deze zijn steeds met een hoofdletter aangeduid. Dit omdat ze speciale waarden zijn voor Python.

Je kan hiermee vergelijkingen gaan maken. Later gaan we die vergelijkingen gebruiken om ons programma de ene of de andere kant uit te sturen.

Vergelijking operatoren

Om een boolean te verkrijgen maak je gebruik van operatoren. Dit wordt gebruikt om 2 waarden te vergelijken.

operator	Wat doet de operator
==	Zijn de waarden gelijk?
!=	Zijn de waarden niet gelijk?
<	Is de waarde kleiner dan de andere?
>	Is de waarde groter dan de andere?
<=	Is de waarde kleiner of gelijk aan?
>=	Is de waarde groter of gelijk aan?

LET OP!
Je kan als je iets wil printen niet 2 datatypes gaan mengen/optellen.
Je kan ook meer datatypes dan numbers vergelijken.

print('5 == 8' + 5 == 8) ▫ Dit zal dus niet werken!
Print('5 == 8', 5 == 8) ▫ Dit zal werken, maar geeft eigenlijk een opsomming van de 2 waarden. (string, boolean)

Maak 2 variabelen met een integer aan.
Print telkens de vergelijking tussen de integers.
Zet ook in een string erbij welke operator je juist gebruikt.

Vergelijk je voornaam met hoofdletter met je voornaam zonder hoofdletter.
Wat is het resultaat.
Hoe zou je dit kunnen omzeilen?

Vergelijk het getal 5 met de string 5. Wat is het resultaat?

logische operatoren

Logische operatoren worden gebruikt om meerdere vergelijkingen te evalueren.
Dit wordt gebruikt wanneer er aan één of meerdere vergelijkingen moet worden voldaan.

Deze operatoren zijn basis voor alles wat digitaal is!

operator	Wat betekend de operator?
and	True wanneer beide stellingen waar zijn.
or	True wanneer minstens 1 stelling waar is.
not	True wanneer de stelling False is.

Wat bekom je dan met de volgende operatoren?

```
print((9 > 7) and (2 < 4))
print((8 == 8) or (6 != 6))
print(not(3 <= 1))
```

Waarheidstabellen

Deze operatoren zijn redelijk universeel in alle programmeertalen. Ze zijn de basis waarop algoritmes gebouwd worden.

Door deze operatoren bekom je een aantal tabellen met wiskundige logica.
Als je deze onder de knie hebt zal het bedenken van creatieve oplossingen voor je algoritmes vlotter verlopen.

== -tabel

X	□==	Y	uitkomst
True	==	True	
True	==	False	
False	==	True	
False	==	False	

and -tabel

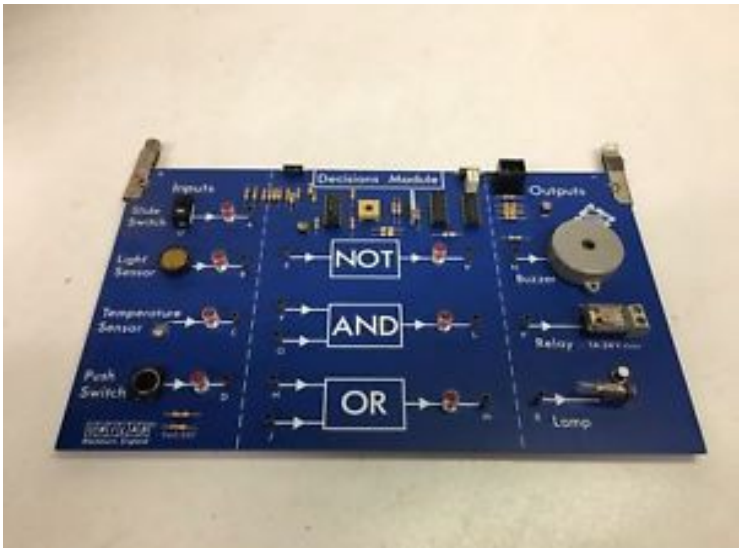
X	and	Y	uitkomst
True	and	True	
True	and	False	
False	and	True	
False	and	False	

or -tabel

X	or	Y	uitkomst
True	or	True	
True	or	False	
False	or	True	
False	or	False	

not -tabel

not	X	uitkomst
not	True	
not	False	
not	True	
not	False	



Condities

als...dan

Met operatoren kan je gebruik gaan maken van condities. Hierbij wordt een stelling gegeven en als aan deze stelling voldaan wordt dan wordt de bijhorende code uitgevoerd.

ALS …..... DAN ….....

In Python gebruik je hiervoor een woordje dat in bijna alle programmeertalen terugkomt.

if

vb.:

```
if True:
        print(string)
```

LET OP! Vanaf hier moet je extra goed gaan opletten met je spaties en tabs. Als je alles aan op een zelfde niveau zet dan krijg je een foutmelding! Let ook op de : na de if.

Alle code die nadien volgt zal gewoon worden uitgevoerd.

als...dan...anders...

Een uitbreiding hierop is een conditie die ofwel de ene code ofwel de andere code uitvoert.

ALS …..... DAN …...ANDERS....

if
else

vb.:

```
if True:
        print(string1)
else:
        print(string2)
```

als...dan...anders als...anders...

Je kan ook meerdere stellingen controleren.

ALS ……. DAN …..ANDERS ALS....ANDERS

if
elif
else

vb.:

```
if True:
        print(string1)
elif True:
        print(string2)
else:
        print(string3)
```

Let op! Vanaf er een aan een conditie voldaan wordt zal de code rest overgeslagen worden en wordt er onder de conditie verder gegaan.

Geneste condities

Condities kunnen ook genest worden. Dit wil zeggen dat elke *if* of *elif* weer uit een eigen opbouw van condities kan bestaan.

```
if True:
        if True:
                print(string1a)
        else:
                print(string1b)
elif True:
        print(string2)
else:
        print(string3)
```

Opdracht A:

Niet geslaagd: je hebt een gewogen percentage van minder dan 50% behaald.

Geslaagd op voldoende wijze: je hebt een gewogen percentage van minder dan 68% behaald.

Geslaagd met onderscheiding: je hebt een gewogen percentage van ten minste 68% behaald.

Geslaagd met grote onderscheiding: je hebt een gewogen percentage van ten minste 77% behaald.

Geslaagd met grootste onderscheiding: je hebt een gewogen percentage van ten minste 85% behaald.

Geslaagd met grootste onderscheiding en de gelukwensen van de examencommissie: je hebt een gewogen percentage van ten minste 90% behaald.

Maak een constructie met condities om de juiste graad van verdienste weer te geven.
Print dit voor student met een uitslag van 88%.
Maak gebruik van variabelen.