Python 2

DEEL 4, PROGRAMMASTRUCTUREN

Taak 1 – boolean condities

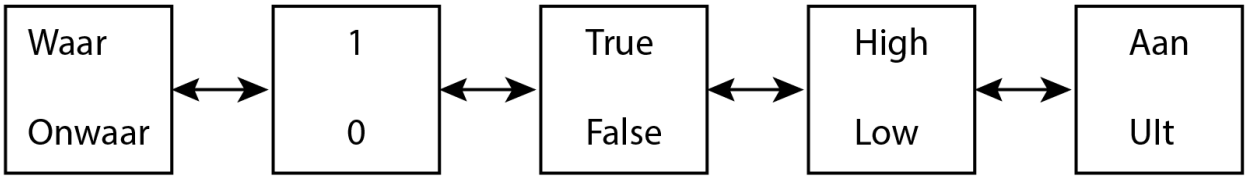
Nu we (enkele) datastructuren hebben besproken, kunnen we ook eens kijken hoe je structuur brengt in jouw programma. Ook heb je inmiddels Thonny geïnstalleerd en die omgeving zullen we gebruiken. Thonny geeft je inzicht in hoe een programma wordt uitgevoerd door de machine.

In de eerste module werd me duidelijk dat velen van jullie nog moeite hebben met de structuur in een programma, met name van de programflow; de volgorde waarin statements worden uitgevoerd. Zo werd een conditie (branching statement) geschreven voordat de variabelen in kwestie een waarde hadden. Dat geeft dan natuurlijk een uitkomst van die conditie die niet is wat je verwacht.

Laten we het eerst eens hebben over de terminologie: Wat bedoelen we met een conditie? En welke termen worden nog meer gebruikt om hetzelfde aan te duiden?

If, Else, Elif = conditie = branche = If statement = …

In elk van die gevallen wordt een statement getest op waar of onwaar. Ook hiervoor bestaan verschillende termen:



In verschillende toepassingen worden deze termen gebruikt om aan te geven of iets "1" of "0" is.
Allemaal gebruiken ze hetzelfde onderliggende datatype: een boolean (of kortweg bool).

Heel wat functies en operatoren in Python geven één van deze waardes terug. Voer de volgende opdrachten maar eens in het shell-veld van Thonny in:

```
>>> namen = ['lies', 'jan', 'kees', 'mireille', 'koen', 'rob']
>>> 'kees' in namen
```

```
>True
>>> 'aniek' in namen
False
>>> namen[0] == 'lies'
>True
>>> namen[1] != 'kees'
>True
>>> 2<5
>True
>>> 2>5
False
>>> 3.1 is_integer()
False
>>> 3.0 is_integer()
>True
```

Zoals je ziet, kun je met if testen of een element zich in een lijst bevindt. Met == test je op de gelijkheid van twee objecten en met != op de ongelijkheid. Vergelijkingen zoals < (kleiner dan) en > (groter dan) geven ook een booleaanse waarde terug. Daarnaast heb je ook <= (kleiner dan of gelijk aan) en >= (groter dan of gelijk aan). En een functie zoals is_integer() is handig om van een float te bepalen of het een geheel getal is.

Hoe lees je nu dergelijke statements zodat je ook begrijpt wat de machine doet? Je leest ze altijd als een vraag waarop je Ja of Nee kunt antwoorden.

Een computer kan alleen omgaan met gesloten vragen.

Bijvoorbeeld:

Python code	**In spreektaal**
'kees' in namen	bestaat (de string) 'kees' in (de lijst) namen?
'aniek' in namen	**vul zelf in:**
namen[0] == 'lies'	is het eerste element van (de lijst) namen gelijk aan (de string) 'lies' ?
namen[1] != 'kees'	is het element met index[1] in (de lijst) namen ongelijk aan (de string) 'kees' ?
2 < 5	is (het getal) 2 kleiner dan (het getal) 5?
2 > 5	is (het getal) 2 groter dan (het getal) 5?
3.1 is_integer	is (het getal) 3.1 een integer?
3.0 is_integer	is (het getal) 3.0 een integer?

Opdracht 1:

Bedenk in totaal 10 gesloten vragen die je over de lijst namen kunt stellen en formuleer ze dan als een statement. Denk hierbij ook aan vragen over het aantal elementen in deze lijst, vragen over het laatste en het eerste element van namen, vragen over een naam die voorkomt in een range binnen de lijst. Maak gebruik van '==' en '=!', '>' en '<'. Stel zoveel mogelijk verschillende vragen.

Schrijf deze vragen eerst op in spreektaal en formuleer dan het bijbehorende Python statement. Lever dit in als een tabel (zoals hierboven).

Met booleaanse waarden zelf kun je ook 'rekenen', namelijk met de operatoren **and, or** en **not.**

- **x or y** is **True** als x òf y of beide objecten **True** zijn.
- **x and y** is **True** als beide objecten **True** zijn.
- **not x** is True als x False is en andersom.

Met twee elementen (a en b) bestaan er 4 verschillende situaties in boolean wiskunde. a kan True zijn, maar ook False en datzelfde geldt voor b.
In een tabel ziet dat er dan zó uit. Hier zijn alle mogelijke combinaties (4) van de status van a en b uitgewerkt:

Status a	Status b
False	False
False	True
True	False
True	True

Opdracht 2: Vul in onderstaande tabel de juiste waarden "TRUE" of "FALSE" in. De laatste twee kolommen zijn echte breinkrakers.

Status a	Status b	a AND b	a OR b	not (a AND b)	not(a OR b)	(not a) AND b	a AND (not b)
False	False						
False	True						
True	False						
True	True						

Over breinkrakers gesproken. Wat nu als je niet twee, maar drie variabelen hebt?
Opdracht 3: Vul onderstaande tabel aan totdat je alle 8 mogelijke combinaties hebt beschreven.

Status a	Status b	Status c
FALSE	FALSE	FALSE

Taak 2 – condities vervolg

Opdracht 4:

We hebben een variabele: persoon. Van dit persoon zijn verschillende gegevens bekend: Naam, woonplaats, leeftijd. Stel je een persoon voor die Ganesh heet, in Aalsmeer woont en 19 jaar is. Je zou dit als een lijst kunnen vastleggen in Python

Persoon = ['Ganesh', 'Aalsmeer', 19]

Hieronder een paar enkelvoudige vragen. Vul de code in Python in achter de vraag.

Heet de persoon in kwestie Ganesh? → **wordt in Python?** →

Woont deze persoon in Amsterdam? → **wordt in Python?** →

Is deze persoon meederjarig? → **wordt in Python?** →

Vervolgens stellen we meervoudige vragen met 'and' en 'or' in de Python code.

Heet de persoon Ganesh? èn Woont de persoon in Amstelveen?

Hier staan dus twee vragen (condities). We moeten dan ook elke vraag formuleren in Python en ze tenslotte combineren met een 'and'. Bij een 'and' moeten beide condities waar (True) zijn, wil het hele statement True zijn.

Eerste vraag: persoon[0] == 'Ganesh'

Tweede vraag: persoon[1] == 'Amstelveen'

Wat wordt de Python Code voor deze twee condities in één regel?
Stap door deze code heen met Thonny om precies te zien hoe de computer dit statement evalueert en tot een antwoord komt.

Als aan één van de twee condities moet worden voldaan, gebruik je 'or'

Opdracht 5:

Stel nu eens de volgende vraag: Heet de persoon "Jaap"? of Woont hij in Aalsmeer?

1. Wat wordt de Python Code? Schrijf het statement in Thonny.
2. Stap door deze code heen met Thonny om precies te zien hoe de computer dit statement evalueert en tot een antwoord komt.

Opdracht 6:

Verzin nog twee condities met een 'and' en twee condities met een 'or'.

1. Formuleer ook deze condities eerst in spreektaal en schrijf ze dan in Python code in Thonny.
2. Gebruik Thonny om te zien hoe jouw condities worden geïnterpreteerd in Python.

Taak 3 - Waardes testen (uitdaging).

Dan gaan we nu een Python programma schrijven. Het programma vraagt de gebruiker om een geheel getal van 1 tot en met 10 in te vullen en toont het getal dan in een woord uitgeschreven. Maar als de gebruiker een getal kleiner dan 1 of groter dan 10 invult, krijgt hij een foutmelding.

Opdracht 7:

Vraag: zou je met deze informatie al zèlf dit programma kunnen maken?
Voor degenen die de makkelijke weg volgen: in de volgende begint de uitleg, daarna de code.
Voor degenen die liever een uitdaging oppakken: ga nog niet naar de volgende pagina, maar probeer het eerst eens zèlf. (en verzamel respectpunten)

Om je op gang te helpen, kun je jezelf de volgende vragen stellen:
Welke datastructuur kan ik hier het best voor gebruiken? Want: hoe koppel ik die getallen aan de juiste tekst? Hoe vraag ik om een input van de gebruiker? En wat ga ik doen met die input? Aan welke voorwaarden (condities) moet die input voldoen? En hoe schrijf ik die condities op in Python?

1. Maak op de eerste regel maken een dictionary aan met de getallen van 1 tot en met 10 als sleutel, met telkens als bijbehorende waarde een string met het getal in een woord uitgedrukt.
2. Roep de functie input aan met als argument een string met de vraag om een getal in te voeren. De functie input geeft als waarde de string terug die de gebruiker heeft ingevoerd.
3. Zet de string met int om naar een geheel getal, op voorwaarde natuurlijk dat de gebruiker daadwerkelijk een getal heeft ingevoerd.
4. Test met if getal < 1 of het getal kleiner is dan 1.
   Is dit het geval dan wordt de regel erna uitgevoerd.
   Is dit niet het geval dan controleert Python de voorwaarde in de regel: elif getal > 10.

Zo ja, dan wordt de regel erna uitgevoerd; zo nee, dan wordt de regel na else: uitgevoerd.

Zo'n blok als hierboven kan meerdere controles met elif bevatten: die worden dan één voor één getest tot er aan een voorwaarde is voldaan. Overigens zijn de controles met elif en else optioneel. Je kunt dus eenvoudig alleen met if een voorwaarde testen: in het geval niet aan die voorwaarde is voldaan, gebeurt er niets en gaat het programma gewoon verder.

Nog één kans om het programma zèlf te maken na deze uitleg. Zo niet, dan moet je dit op z'n kop lezen.

Het programma ziet er zo uit:

```
getalLen = {1: 'een', 2: 'twee', 3: 'drie', 4: 'vier', 5:
'vijf', 6: 'zes', 7: 'zeven', 8: 'acht', 9: 'negen', 10:
'tien'}
getal = int (input ("Voer een geheel getal van 1 tot en met 10
in:"))
if getal < 1:
    print("Het getal is kleiner dan 1.")
elif getal > 10:
    print("Het getal is groter dan 10.")
else:
    print(getalLen[getal])
```

Opdracht 8:

1. Maak het programma volgens de uitleg.
2. Test de verschillende onderdelen van je programma eens door een getal in te voeren dat te klein, te groot of binnen het bereik ligt. Na elke invoer druk je op Enter en krijg je het resultaat te zien. Voer het programma daarna opnieuw uit voor die volgende poging.

Taak 4 - Alternatieven voor de code in taak 4.

De hele riedel if's en elifs kan vervangen worden door één statement.

Weet je nog dat we het 'in range' statement gebruikten in de Python Basic module? Welnu, die constructie kunnen we ook in een if statement gebruiken: Als volgt:

```
if getal in range(1,10) :
```

Hierna volgt dan de uitvoer van het getal in letters.

Gevolgd door een 'else' met een andere output wanneer het getal 'out of bounds' is.

Opdracht:9

Ons programma heeft nog één groot minpunt: als de gebruiker een waarde buiten het bereik invoert, stopt het programma gewoon en moet de gebruiker het programma opnieuw uitvoeren. Waarom kan

het programma in dat geval niet uit zichzelf opnieuw naar een getal vragen?
Stel je nu eens voor dat het programma net zolang om een goede input blijft vragen totdat de gebruiker een getal in de range geeft. Met andere woorden, het programma herhaalt de vraag om input totdat de gebruiker het goed doet. Hoe zou je dat kunnen programmeren?

Vragen die je jezelf kunt stellen:
*Hoe maak ik een loop die eindeloos herhaald wordt zolang de conditie True is. Welke conditie moet ik dan maken zodat die True is bij een antwoord dat **niet** in range(1,10) is? Wanneer (op welk moment in het programma) moet ik nu de input vragen?*
Tip: Zoek eens op wat het keyword 'not' in Python betekent.

Pas nu het programma aan zodat:

- Er maar één statement is waarin we vaststellen of de input van de gebruiker tussen 0 en 10 ligt.
- Er een eindeloze lus is waarin het programma net zolang de vraag herhaalt tot de invoer klopt. De lus wordt verlaten wanneer de invoer goed is.

Taak 6 - Samenvatting

In dit hoofdstuk hebben we opnieuw de stap gezet van afzonderlijke Python-opdrachten naar Python-programma's. Dat deden we in de ontwikkelomgeving Thonny. We bekeken het datatype bool en werkten met voorwaarden in if-blokken en we hebben een loop gemaakt die alleen wordt uitgevoerd zolang een conditie true is, (zonder gebruik van if – else). Daarbij maakten we ook gebruik van de functie range voor een bereik van getallen. We gebruikten input en print voor invoer en uitvoer. Het volgende deel gaat volledig over invoer en uitvoer, en we bespreken hoe je door elementen in een lijst of dictionary loopt.

Het maken van een programma begint met het stellen van de juiste vragen aan jezelf. Pas wanneer je de antwoorden op die vragen hebt, kun je beginnen met coderen.