

Start to Program: Python, Les 6

Flowcharts







Als je een programma zoals *hoger-lager* uitwerkt is het handig om een plan te maken. Bekijk het zo:

Als je een klein tuinhuis bouwt heb je geen architect of plan nodig. Bouw je echter een huis dan komt een vooraf uitgewerkt plan wel van pas.

Dit doe je ook bij het programmeren. Je kan een flowchart opstellen als steun. Hiermee stel je een bepaalde **sequentie**, **volgorde**, op voor je programma.

Een flowchart of schema kan je gewoon maken op een blaadje papier, maar er zijn ook een aantal handige tools. Er zijn ook een aantal universele afspraken onder programmeurs zodat iedereen je schema kan lezen.

De basis ziet er als volgt uit:

Name	Symbol	Usage
Start or Stop		The beginning and end points in the sequence.
Process		An instruction or a command.
Decision		A decision, either yes or no.
Input or Output		An input is data received by a computer. An output is a signal or data sent from a computer.
Connector		A jump from one point in the sequence to another.
Direction of flow		Connects the symbols. The arrow shows the direction of flow of instructions.

Je kan deze symbolen terugvinden in tools om een schema/flowchart uit te werken.

<https://www.draw.io/>

Opdracht:

Om te oefenen neem je een uitgewerkt programma. In alle andere gevallen stel je eerst een flowchart op en ga dan pas aan de slag.
Stel een flowchart op voor het programma hoger-lager.

Lussen

for-lus

Vorige lessen had je al een **lus of iteratie** gezien, namelijk de **while-loop** of **conditionele lus**. Hierbij moet er aan een bepaalde conditie voldaan worden.

Een andere lus is de **for-loop**. Deze kan je gebruiken als je weet hoeveel keer je code moet worden herhaald. Het is dus een **eindige lus** waarbij je het einde kent.

Deze lus is uit verschillende delen opgebouwd:

```
for variabele_naam_teller in range(start,stop,stappen):  
    code
```

De variabele_naam_teller is de naam van je teller. Deze gaat heel vaak de letter i zijn. Die staat voor index.

Het instellen van de teller gebeurt door een bereik op te geven.

Met de methode **range()**

Deze kan uit 3 waardes bestaan.

Start: de eerste waarde waar de teller begint. INCLUSIEF het getal.

Stop: de waarde wanneer de teller eindigt. Exclusief dit getal.

Stappen: De waarde waarbij de teller elke loop vermeerderd wordt.

De eerste en de laatste waarden zijn optioneel. Standaard wanneer niet ingevuld zijn 0 voor de start en 1 voor de stappen.

Dit is zeer gelijkend zoals indexering van letters in een string.

Vb:

```
for i in range(5):  
    print(i)
```

Opdracht:

- a) Print alle getallen tussen 20 en 26.
- b) Print alle even getallen tussen 30 en 50
- c) Tel af van 100 tot 0. Print alleen de tientallen.
- d) Tel van -100 tot 0. Inclusief 0 print alleen de tientallen.

Je kan natuurlijk meer dan alleen getallen printen.

Als je een string erbij haalt stellen zich meteen een aantal extra mogelijkheden.

Bij een string had namelijk elk symbool een index of nummer toegewezen gekregen.

Je moet hier geen bereik opgeven, maar de variabele waar de string in opgeslagen zit.

```
for letter in voornaam:  
    print(letter)
```

Je kan ook de indexering van de string gebruiken:

```
for letter in voornaam[2:6]:  
    print(letter)
```

OEFENEN:

Maak voor alle opdrachten eerst een flowchart.
Met andere woorden eerst denken en dan doen.

1.Schrijf een programma dat alle getallen weergeeft tussen 1500 en 2700 (inclusief 2700) die deelbaar zijn door 5 en 7.

2.Vraag een input van de gebruiker. Je programma geeft weer hoeveel cijfers en hoeveel letters er in de string voorkomen.

TIP: Probeer dit op 2 manieren. Eerst met de try wijze. Bij de 2de manier kan je gebruik maken van de methodes `var_naam.isdigit()` en `var_naam.isalpha()`

3.Maak een programma dat temperatuur kan omzetten van graden Celcius naar Fahrenheit.

INPUT: Voer de temperatuur in die je wil omzetten: (vb. 120C of 45F)
Zoek online naar de omzetformule.

4.Maak de volgende figuur:

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
*
```

TIP: Standaard staat je er op het einde van een print een `\n` om een nieuwe lijn te maken. Met `end=` kan je dit veranderen. Gebruik dus: `print("", end=" ")`

5.Hoeveel even en oneven getallen zijn er in de reeks: 15489723455614 ?

6.Maak een programma dat de Fibonacci reeks tot 50 weergeeft.

Kan je de getallen ook naast elkaar laten verschijnen in een string met een komma ertussen?

7. Binaire code bestaat slecht uit 1 en 0. Zet 01011011 om.
Vervolledig je programma zodat een reeks van MAX 8 nullen en enen omgezet kan worden. Het programma mag dus ook alleen strings aanvaarden die uit 1 en 0 bestaan.

