PROGRAMMEREN 2: TOPIC 4

CONTROLESTRUCTUREN

UCLL HOGESCHOOL

OVERZICHT

- Introductie
- Sequentie
- Selectie
- Iteratie
- Opdrachten

INTRODUCTIE

ALGORITMES EN CONTROLERSTRUCTUREN

Hoe zou je het concept 'algoritme' uitleggen?

ALGORITMES EN CONTROLERSTRUCTUREN

Hoe zou je het concept 'algoritme' uitleggen?

Een algoritme is een voorschrift om in een eindig aantal stappen een welbepaalde taak uit te voeren.

ALGORITMES EN CONTROLERSTRUCTUREN

Hoe zou je het concept 'algoritme' uitleggen?

Een algoritme is een voorschrift om in een eindig aantal stappen een welbepaalde taak uit te voeren.

Controlestructuren beschrijven in welke volgorde de stappen van een algoritme worden uitgevoerd.

CONCEPTEN

Onder controlestructuren worden volgende concepten gegroepeerd:

- Sequentie: voer opdrachten stap voor stap uit
- Selectie: een splitsing of keuze in het programma
- Iteratie: een of meerdere stappen herhalen

VOORSTELLING

Computerprogramma's zijn *een* manier om algoritmes voor te stellen met controlestructuren, maar er zijn er nog ander. Kan je er zelf bedenken?

VOORSTELLING

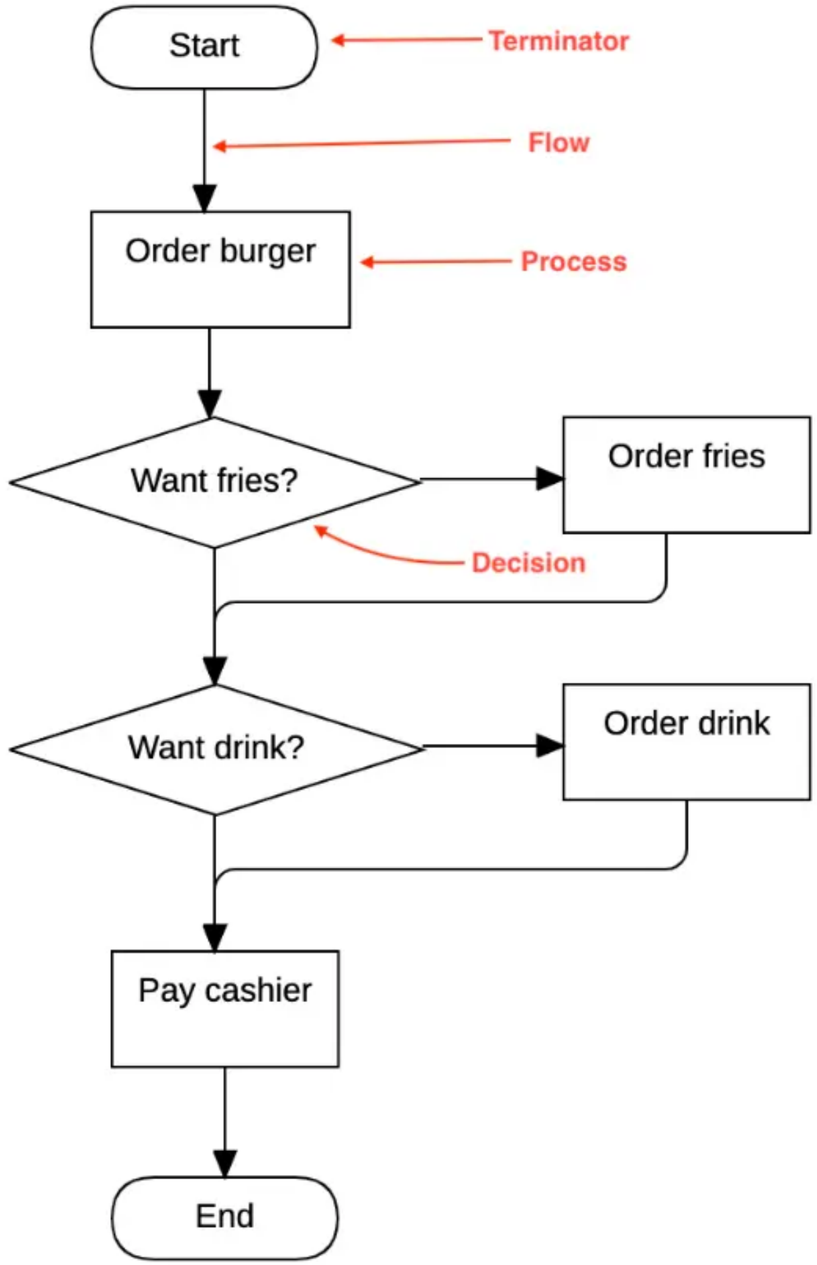
Computerprogramma's zijn *een* manier om algoritmes voor te stellen met controlestructuren, maar er zijn er nog ander. Kan je er zelf bedenken?

- Genummerd stappenplan in mensentaal
- Een filmpje (bv. voor een recept)
- Stroomdiagrammen (Eng: *flowcharts*)
  → veel gebruikt in informatica

VOORSTELLING

Computerprogramma's zijn *een* manier om algoritmes voor te stellen met controlestructuren, maar er zijn er nog ander. Kan je er zelf bedenken?

- Genummerd stappenplan in mensentaal
- Een filmpje (bv. voor een recept)
- Stroomdiagrammen (Eng: *flowcharts*)
  → veel gebruikt in informatica

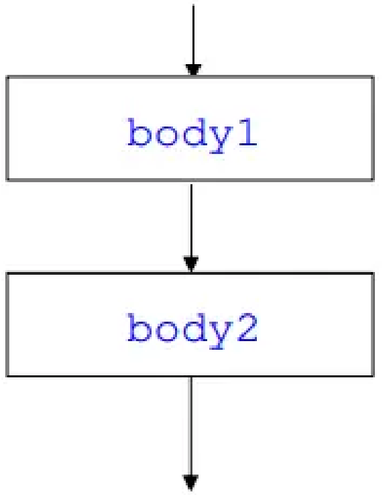


STAP VOOR STAP

SEQUENTIE

SEQUENTIE

> *"Doe eerst dit, dan het volgende."*



De meest gangbare programmeertalen volgen deze regel, maar er bestaan talen die het anders aanpakken.

> **Let op:** *stap voor stap* betekent niet noodzakelijk *van boven naar onder* in code!

KEUZES IN PROGRAMMA'S

SELECTIES

ALGEMEEN

Selecties stellen een splitsing of vertakking voor in een algoritme. Welke tak gekozen wordt, hangt af van de voorwaarde (de **conditie**).

Welke speciale woorden gebruik je in Python voor selecties?

Wat is het datatype van condities in Python?

ALGEMEEN

Selecties stellen een splitsing of vertakking voor in een algoritme. Welke tak gekozen wordt, hangt af van de voorwaarde (de **conditie**).

Welke speciale woorden gebruik je in Python voor selecties?

`if..elif..else`

Wat is het datatype van condities in Python?

ALGEMEEN

Selecties stellen een splitsing of vertakking voor in een algoritme. Welke tak gekozen wordt, hangt af van de voorwaarde (de **conditie**).

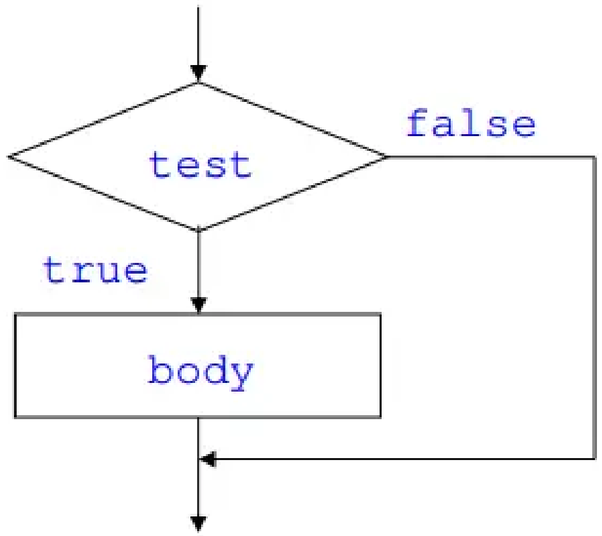
Welke speciale woorden gebruik je in Python voor selecties?

`if..elif..else`

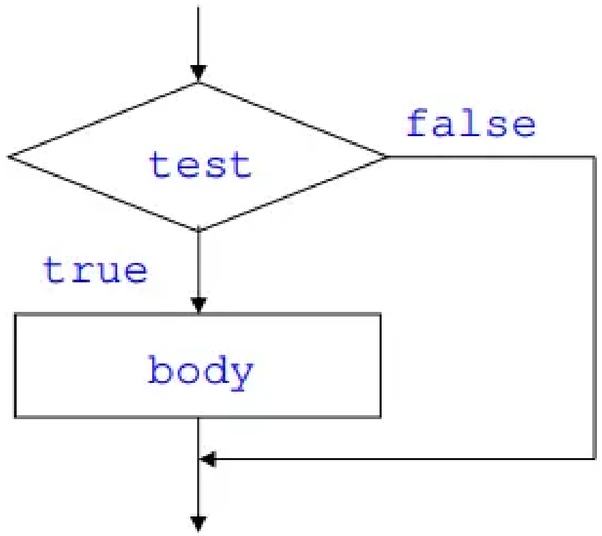
Wat is het datatype van condities in Python?

`bool`: elke conditie is ofwel `true`, ofwel `false`

EENZIJDIGE SELECTIE



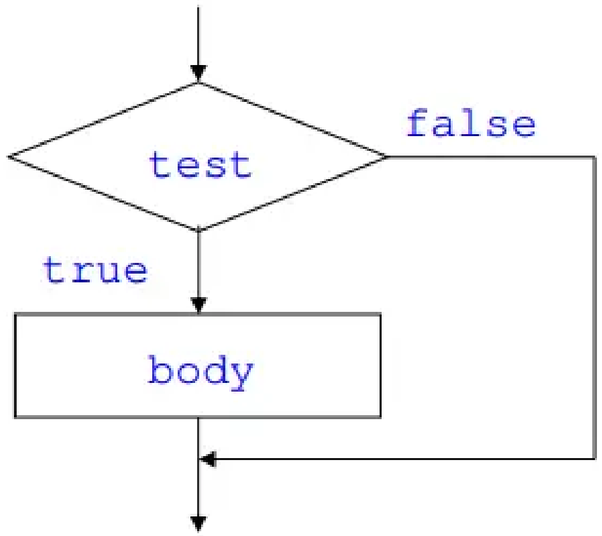
EENZIJDIGE SELECTIE



Python

```python
if test:
  body()

# Ga verder met rest van programma
```

EENZIJDIGE SELECTIE

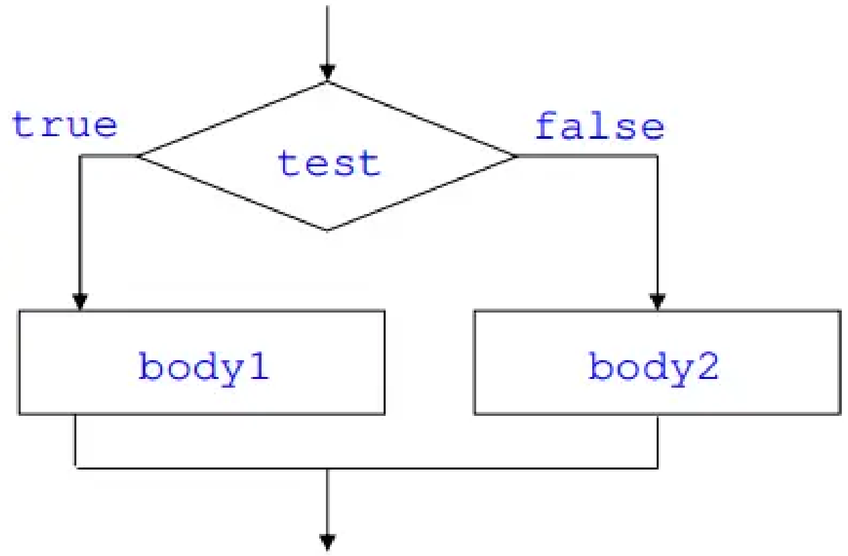


Python

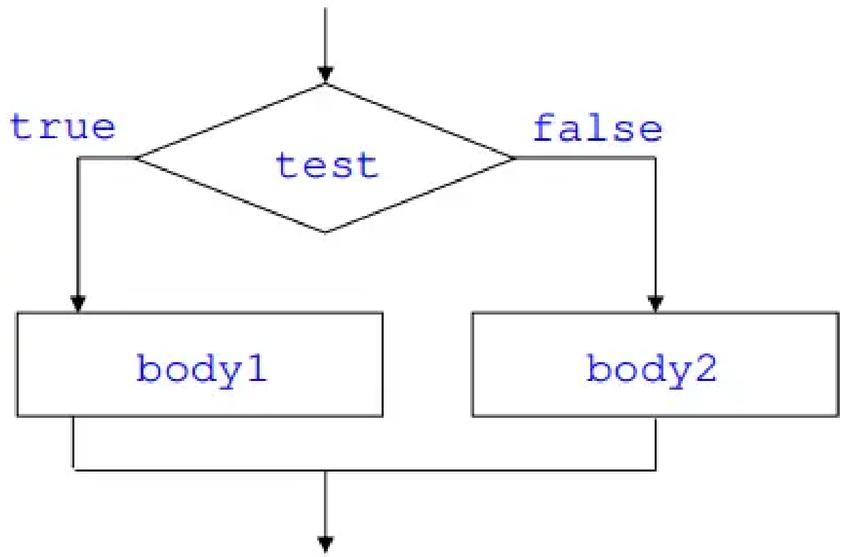
```
if test:
  body()

# Ga verder met rest van programma
```

Indentatie van code is belangrijk in Python! Denk altijd goed na welke lijnen naar links/rechts geïntendeerd moeten zijn.

TWEEZIJDIGE SELECTIE



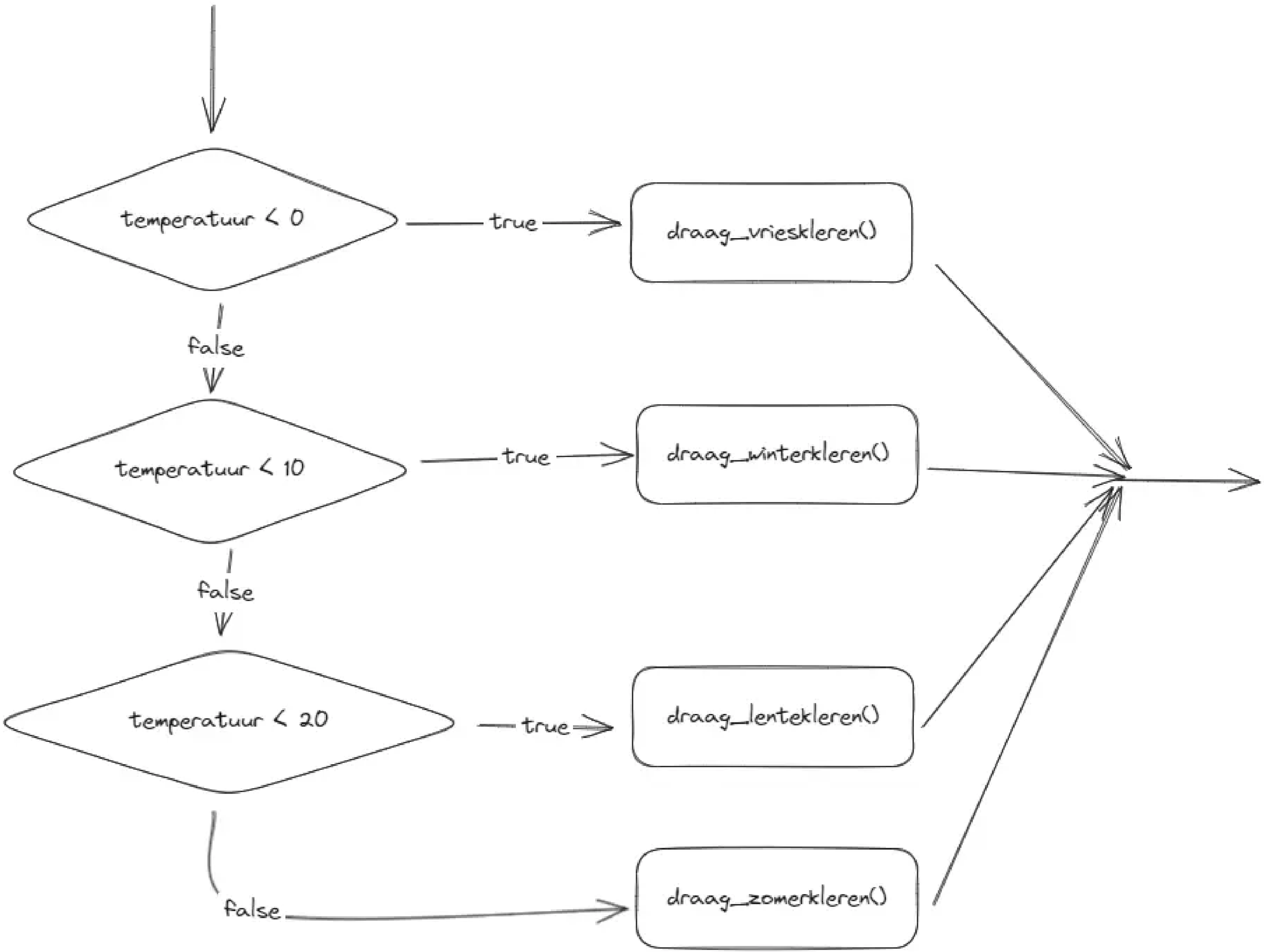
TWEEZIJDIGE SELECTIE



Python

```
if test:
  body1()
else:
  body2()

# Ga verder met rest van programma
```

MEERVOUDIGE SELECTIE



MEERVOUDIGE SELECTIE: CODE

Eerste poging: geneste `if..else` statements:

```python
if temperatuur < 0:
    draag_vriesskleren()
else:
    if temperatuur < 10:
      draag_winterkleren()
    else
        if temperatuur < 20:
            draag_lentekleren()
        else:
            draag_zomerkleren()

# Ga verder met rest van programma
```

MEERVOUDIGE SELECTIE: CODE

Tweede poging: gebruik `elif` om de code te ontnesten:

```python
if temperatuur < 0:
  draag_vrieskleren()
elif temperatuur < 10:
  draag_winterkleren()
elif temperatuur < 20:
  draag_lentekleren()
else:
  draag_zomerkleren()

# Ga verder met rest van programma
```

MEERVOUDIGE SELECTIE: `switch`

Sommige talen ondersteunen een 'gevorderde' notatie voor meervoudige selecties. Meestal is dit in de vorm van een `switch` statement. Bijvoorbeeld in JavaScript:

JavaScript

```
switch (fruit) {
  case "appel":
    maakAppelTaart();
    break;
  case "banaan":
    maakBananaSplit();
    break;
  case "citroen":
    maakLimoncello();
    break;
  default:
    gaEtenKopen();
```

MEERVOUDIGE SELECTIE: `switch`

Python ondersteunt dit niet. De makers hadden deze filosofie:

> *"There should be one-- and preferably only one -- obvious way to do it.*
>
> — The Zen of Python

Elke `switch`-statement kan herschreven worden met `if..elif..else`.
`switch` is dus optioneel in programmeertalen.

TIPS VOOR ONDERWIJS

Laat leerlingen selecties uittekenen als flowcharts om ze beter te begrijpen. Laat ze nadien pas programmeren.

Hou `switch` als iets optioneel voor meer gevorderde lessen (als de taal het al ondersteunt). Zorg dat leerlingen eerst `if..elif..else` goed begrijpen.

ITERATIES

ALGEMEEN

Iteraties dienen om stappen in een algoritme te **herhalen**.

Op een flowchart is een iteratie te herkennen aan een lus in de tekening.

Iteraties zijn cruciaal voor informatica. Ze laten toe om repetitieve (mentale) taken eenvoudig te beschrijven, op een gestructureerde manier.

Welke twee speciale woorden worden in Python (en veel andere talen) gebruikt voor iteraties?

ALGEMEEN

Iteraties dienen om stappen in een algoritme te **herhalen**.

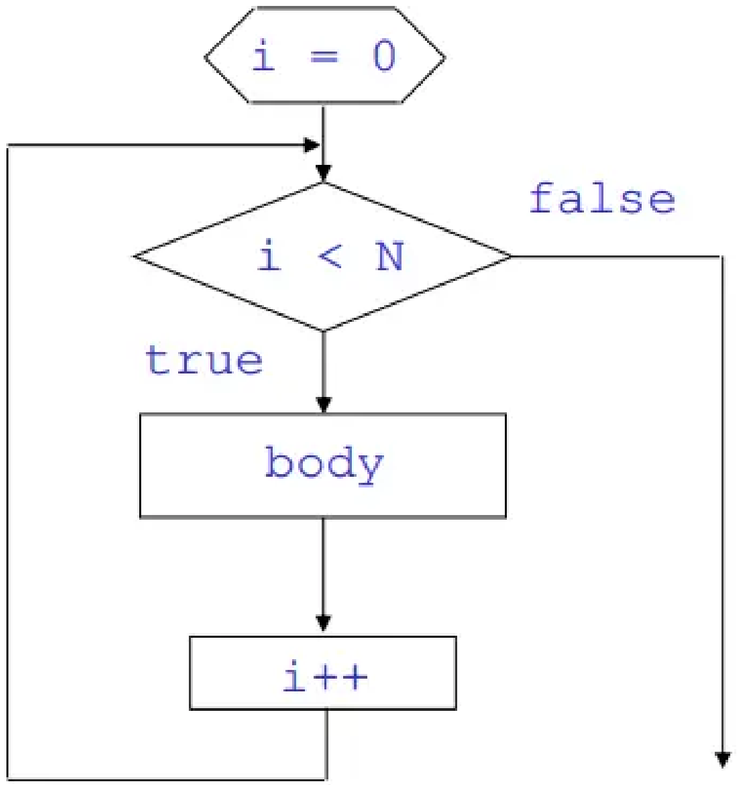
Op een flowchart is een iteratie te herkennen aan een lus in de tekening.

Iteraties zijn cruciaal voor informatica. Ze laten toe om repetitieve (mentale) taken eenvoudig te beschrijven, op een gestructureerde manier.

Welke twee speciale woorden worden in Python (en veel andere talen) gebruikt voor iteraties?

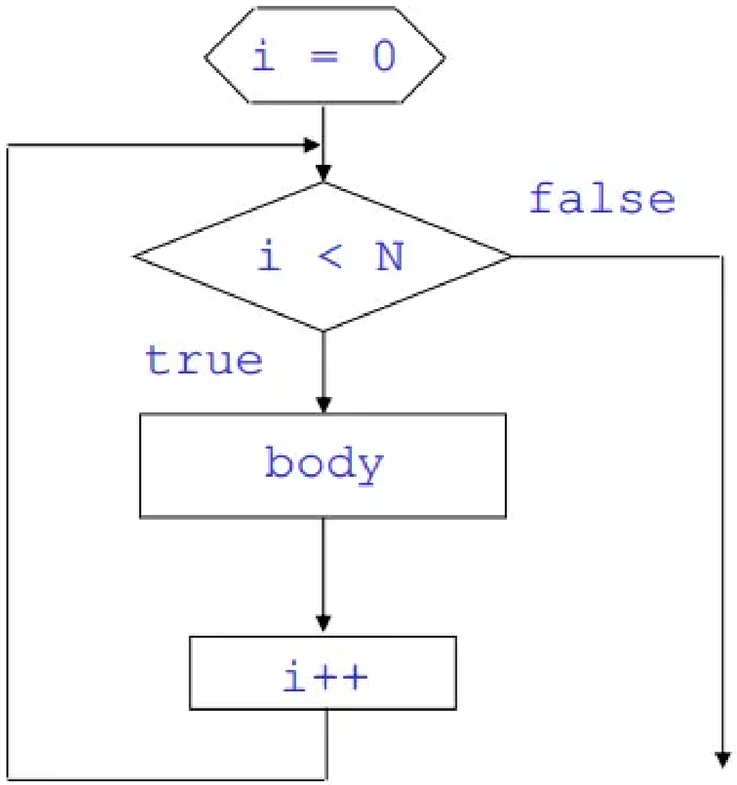
`for` en `while`.

EINDIGE HERHALING



> **i** `i++` betekent: 'verhoog `i` met 1'. Veel talen ondersteunen deze syntax, Python echter niet. Equivalente notaties zijn `i += 1` en `i = i + 1`.

EINDIGE HERHALING



Python

```python
for i in range(0, N):
  body()

# Ga verder met rest van programma
```

i `i++` betekent: 'verhoog `i` met 1'. Veel talen ondersteunen deze syntax, Python echter niet. Equivalente notaties zijn `i += 1` en `i = i + 1`.

DE `range()` VAN EEN `for`-LUS IN PYTHON

```for i in range(3):```
Python

Waardes van `i`:

- `0` ➞ start lus
- `1`
- `2`
- `3` ➞ stop lus, voer NIET uit

```for i in range(2, 5):```
Python

Waardes van `i`:

- `2` ➞ start lus
- `3`
- `4`
- `5` ➞ stop lus, voer NIET uit

```for i in range(10, 4, -2):```
Python

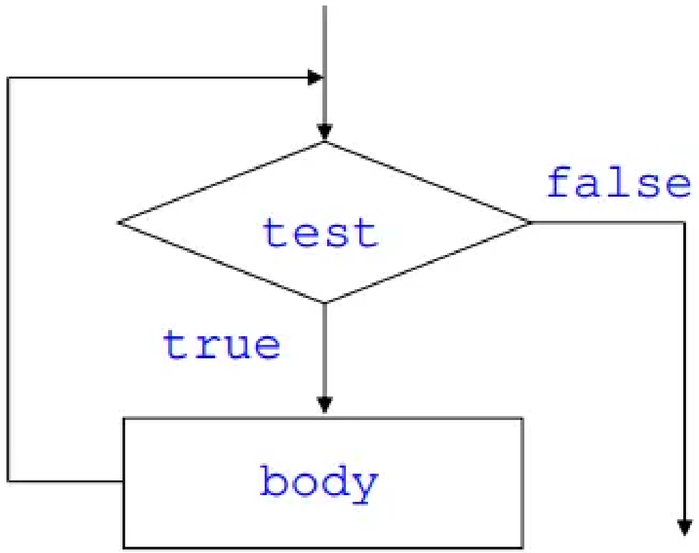
Waardes van `i`:

- `10` ➞ start lus
- `8`
- `6`
- `4` ➞ stop lus, voer NIET uit

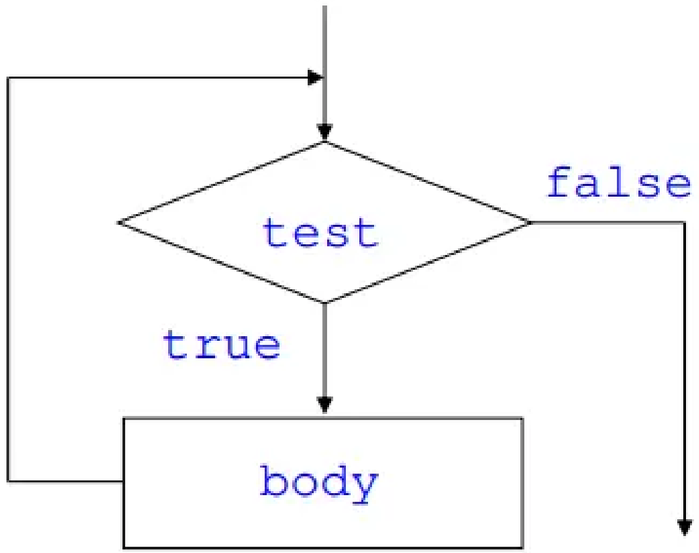
ALGEMEEN: `range(stop, start, step)`

- `stop` is altijd verplicht. De lus stopt net wanneer deze waarde wordt bereikt
- `start` is optioneel, standaard is dit `0`. De lus start met deze waarde
- `step` is optioneel, standaard is dit `1`. De lus verhoogt de teller met deze waarde na elke iteratie
```

VOORWAARDELIJKE HERHALING



VOORWAARDELIJKE HERHALING

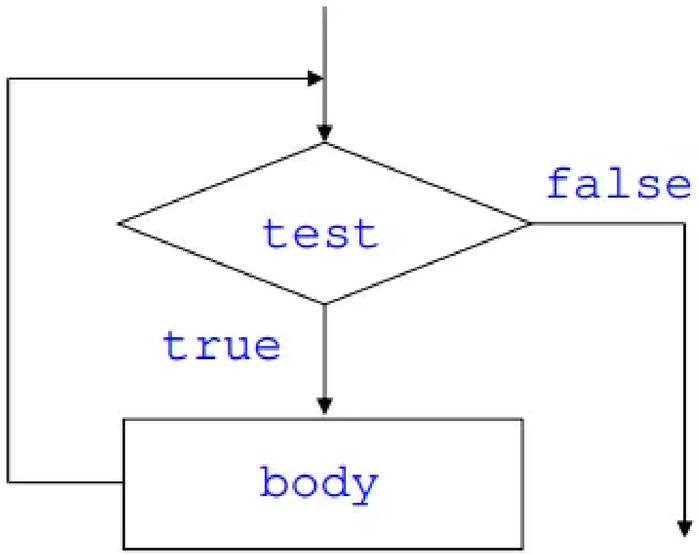


Python

```
while test:
  body()

# Ga verder met rest van programma
```

VOORWAARDELIJKE HERHALING



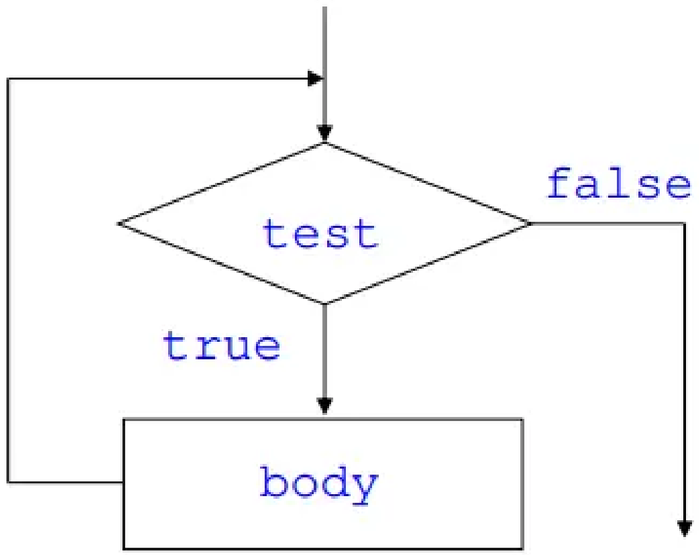
Python

```
while test:
  body()

# Ga verder met rest van programma
```

In deze context is *zolang* een betere vertaling van `while` dan *terwijl*.

VOORWAARDELIJKE HERHALING



Python

```
while test:
  body()

# Ga verder met rest van programma
```

> In deze context is *zolang* een betere vertaling van `while` dan *terwijl*.

Hoeveel keer zal dit type herhaling minstens uitgevoerd worden?

0

1

VOORWAARDELIJKE HERHALING (2)

De `while`-lus zorgt in bepaalde gevallen voor minder elegante code.
Bijvoorbeeld:

```python
film = input("Welke film wil je bekijken?")
while not is_beschikbaar(film):
  film = input("Welke film wil je bekijken?")
```

Merk op dat we de `input` twee keer moeten schrijven.

VOORWAARDELIJKE HERHALING (3)

Een tweede poging met minder duplicatie van code:

```python
vraag_film = True
while vraag_film:
  film = input("Welke film wil je bekijken?")
  if is_beschikbaar(film):
    vraag_film = False
```

Minder duplicatie, maar langer en complexer. Al bij al beter dan eerste poging.

VOORWAARDELIJKE HERHALING (4)

Het kernprobleem is dat in deze situaties we de conditie op het **einde** van de lus willen controleren. De `while-` lus laat enkel toe om de conditie in het **begin** na te kijken.

VOORWAARDELIJKE HERHALING (4)

Het kernprobleem is dat in deze situaties we de conditie op het **einde** van de lus willen controleren. De `while-` lus laat enkel toe om de conditie in het **begin** na te kijken.

Een aantal talen hebben hier een speciale oplossing voor: `do while`

JavaScript

```
let film = "";  // initialiseer met dummywaarde
do {
  film = prompt("Welke film wil je bekijken?");
} while (isBeschikbaar(film));
```

VOORWAARDELIJKE HERHALING (4)

Het kernprobleem is dat in deze situaties we de conditie op het **einde** van de lus willen controleren. De `while-` lus laat enkel toe om de conditie in het **begin** na te kijken.

Een aantal talen hebben hier een speciale oplossing voor: `do while`

JavaScript

```
let film = "";  // initialiseer met dummywaarde
do {
  film = prompt("Welke film wil je bekijken?");
} while (isBeschikbaar(film));
```

Hoeveel keer zal dit type herhaling minstens uitgevoerd worden?

0	1

`while` VERSUS `do while`

Python ondersteunt `do while` **niet**. De reden is dezelfde als bij `switch`: elke `do while` kan eenvoudig herschreven worden met `while` (zoals in de voorgaan de slides werd gedemonstreerd). Dat de code iets langer wordt, is een acceptable trade-off voor de makers van Python.

ANDERE TECHNIEKEN I.V.M. ITERATIES

`break`

Wanneer Python een `break` tegenkomt, wordt de lus onmiddellijk gestop. Andere woorden hiervoor zijn `continue`, `exit`, ...

Het stoppen van lussen op deze manier is sterk af te raden in eender welke taal. Het maakt redeneren over iteraties over het algemeen moeilijker.

`return`

Een `return` stopt een functie onmiddellijk. Het kan dus ook een lus onderbreken. Sommige programmeurs zien hierin hetzelfde probleem als `break` en keuren het dus af. De meerderheid laat `return` binnen een lus echter wel toe om pragmatische redenen.

TIPS VOOR ONDERWIJS

Let op je taalgebruik bij iteraties, zowel in het Nederlands als in het Engels. Sommige talen (bv. Scratch) gebruiken `until` in plaats van `while`. Dit betekent dat condities worden omgekeer.

Hou `do while` als gevorderde leerstof voor latere lessen, of slaag het gewoon over.

Ontmoedig het gebruik van `break`. Hou het voor gevorderde lessen, als extra.

OPDRACHTEN

OPDRACHTEN

1. Maak oefeningen uit hoofdstuk 6 en 7 van het handboek
2. Onderzoek hoe je `for i in range(10, 4, -2):` precies schrijft in volgende talen:
   - Java
   - Rust
3. Onderzoek de `switch-`statement in detail in bijvoorbeeld JavaScript of Java. Hoe werkt `break` hierin? Kan je meerdere `case`s combineren? Is `default` altijd nodig?
4. Onderzoek hoe lussen werken in de blokgebaseerde taal Scratch. Wat zijn de verschillen en gelijkenissen met Python? Kan je je vinden in de keuzes die de ontwikkelaars van Scratch hebben gemaakt?