PROGRAMMEREN 1

LES 2: FUNCTIES, CONDITIES EN SELECTIES

Download PDF-versie

UCLL HOGESCHOOL

OVERZICHT

- Herhaling vorige les + bespreking opdracht
- Functies
- Condities en selecties
- Computationeel denken: algoritmisch denken

VORIGE LES

QUIZ + BESPREKING OPDRACHT

QUIZ

Wat is de uitvoer van volgende code?

```python
print(5 + 2 * 3 - 4 / 2)
```

- 9
- 8.5
- 9.0
- Iets anders/het programma werkt niet

QUIZ

Wat is de uitvoer van volgende code?

```python
x = 2
y = 4 + x
x = y - 1
z = y + x
print(z)
```

- 11
- 8
- '6'
- Iets anders/het programma werkt niet

QUIZ

Wat is de uitvoer van volgende code als als input 70 wordt ingevuld?

```python
antwoord = input("Geef seconden in: ")
rest = antwoord % 60
print("Er blijven", rest, "resterende seconden over")
```

- Er blijven 10 resterende seconden over
- Er blijven10resterende seconden over
- Er blijven 1.166667 resterende seconden over
- Iets anders/het programma werkt niet

RESULTATEN ONDERZOEK

- `print()` in andere talen
- `input()` in andere talen
- vierkantswortel in andere talen
- willekeurig getal genereren in andere talen

FUNCTIES

```python
str(5)
abs(-7)
pow(2, 5)
input("Leeftijd: ")
```

FUNCTIES

```python
antwoord = input("Geef kortingscode in: ")
```

- Functies zijn bestaande stukken code die je kan oproepen
- Programmeertalen hebben een heel aantal ingebouwde functies voor veel voorkomende taken
- Functies zijn herkenbaar aan de haakjes achter hun naam
- Functies geven soms een waarde terug die we kunnen gebruiken in een expressie, of in een variabele kunnen opslaan
- Je kan soms extra parameters meegeven aan een functie
- In latere lessen gaan we zien hoe we zelf functies kunnen definiëren

Functies in de context van programmeren hebben een andere betekenis dan in de context van wiskunde!

FUNCTIES: VOORBEELDEN

- `print(...)`: print een of meerdere parameters in de console. Geeft niets terug
- `input(question)`: toont `question` in de console. Geeft het antwoord van de gebruiker terug
- `int(val)`: converteer `val` naar een `string` en geeft dit terug als resultaat

Zie ook: slides vorige les, handboek hoofdstuk 5

FUNCTIES: MEERDERE PARAMETERS

Sommige functies verwachten meerdere parameters. In dat geval **is de volgorde van parameters belangrijk.**

Python

```
print("appel", "peer", "banaan")  # output: appel peer banaan
print("banaan", "peer", "appel")  # output: banaan peer appel

# pow(a, b) berekent a tot de macht b
pow(2, 3)  # 2 * 2 * 2
pow(3, 2)  # 3 * 3
```

CONDITIES EN SELECTIES

- Booleans
- Logische operatoren
- Selecties

BOOLEAANSE WAARDEN EN EXPRESSIES

De Booleanse logica is vernoemd naar de Engelsman George Boole. In die vorm van logica bestaan er maar twee waarden: `True` en `False`.

Booleanse logica wordt veel gebruikt in de informatica omdat:

- Het vrij makkelijk is voor mensen om te leren
- Het vrij makkelijk te bouwen is op hardwareniveau

Booleanse logica bestaat in alle programmeertalen, maar de exacte notatie verschilt van taal tot taal.

Een expressie die evalueert naar `True` of `False` noemt men een **Booleaanse expressie**.

VERGELIJKINGEN

Vergelijkingen zijn de eenvoudigste categorie van Booleaanse expressies:

```Python
<    kleiner dan
<=   kleiner dan of gelijk aan
==   gelijk aan
>=   groter dan of gelijk aan
>    groter dan
!=   niet gelijk aan
```

Let op: verwar `==` (een gelijkheid) niet met `=` (een variabele een waarde toekennen)!

De meeste talen gebruiken deze notaties, maar er zijn uitzonderingen.

VOORBEELDEN VAN BOOLEAANSE EXPRESSIES EVALUEREN

Python

```
> True
True  # True en False evalueren naar zichzelf

> 6 == 5
False

> 12 <= 18
True

> "appel" == 129
False  # Tijdens de evaluatie wordt rekening gehouden met types
```

QUIZ

Naar welke waarde zal volgende expressie evalueren?

```python
27 == "27"
```

Python

- True
- False
- Iets anders
- Error/programma werkt niet

LOGISCHE OPERATOREN

Je kan Booleaanse waarden combineren:

- `A and B`: evaleert naar `True` als A en B allebei tegelijk `True` zijn
- `A or B`: evaleert naar `True` als minstens A of B `True` is. A en B mogen ook beiden `True` zijn
- `not A`: evaleert naar `True` als A `False` is

Je kan logische operatoren aan elkaar schakelen. Bv. `A and B or not C`. Evaluatie gebeurt van links naar rechts, tenzij er haakjes staan.

Let op: soms zijn haakjes noodzakelijk om de juiste expressie te bekomen!

`(A and B) or not C` is niet hetzelfde als `A and (B or not C)`

Bij twijfel: schrijf de haakjes, ook al is het misschien niet nodig.

SELECTIES

Een **selectie** is een programmeerstructuur die toelaat om een **keuze** te maken binnen een programma. Er bestaan verschillende soorten selecties, maar veruit de meest gebruikte is de `if`- statement.

Selecties maken gebruik van Booleaanse logica.

DE `if`-STATEMENT

```python
if 10 > 5 and "abc" != "xyz":
  print("if-statement was True")
  print("Dit staat nog binnen de if-statement")

print("Deze lijn staat buiten de if-statement")
```

Met een `if`-statement kan je een stuk code laten uitvoeren als aan een bepaalde voorwaarde voldaan is.

- Start met `if`
- Daarna komt een Booleaanse expressie (mag simpel of complex zijn)
- Op het einde komt een dubbelpunt (`:`)
- **De code binnen de `if`-statement is naar rechts gïndenteerd**

Tussen een `if`-blok en de code erna wordt meestal een lege lijn gelaten om de leesbaarheid te verhogen.

DE `else`-STATEMENT

```python
if leeftijd >= 18:
  print("meerderjarig")
else:
  print("minderjarig")
```

- De `else` wordt uitgevoerd als de voorgaande `if` **niet** uitgevoerd wordt
- Voor de rest werkt het exact zoals een `if`
- Je kan nooit een `else` hebben zonder een `if`. Veel programmeertalen zullen dit als een fout rapporteren

DE `elif`-STATEMENT

```python
if levens >= 0:
  print("herkansing, verlies 1 leven")
elif score >= 1:
  print("herkansing, verlies je volledige score")
```

- De `elif` wordt uitgevoerd als de voorgaande `if` **niet** uitgevoerd wordt
- Een `elif` verwacht **wel** een conditie, in tegenstelling tot een `else`
- Voor de rest werkt het exact zoals een `if`

QUIZ

Wat is de uitvoer van volgende code als `score = 12` en `geslaagd_op_andere_vakken = False`?

```python
if score >= 18:
  print("uitmuntend")
elif score >= 10:
  print("geslaagd")
elif score >= 8 and geslaagd_op_andere_vakken:
  print("gedelibereerd")
else:
  print("niet geslaagd")
```

"uitmuntend"

"geslaagd"

"gedelibereerd"

"niet geslaagd"

Geen uitvoer

Error/programma werkt niet

BOOLEANS EN VARIABELEN

Het resultaat van een Booleaanse expressie kan je opslaan in een variabele:

```python
luchtvochtigheid_percent = 80
windsnelheid_km_h = 60
regen = True

slecht_weer = (luchtvochtigheid_percent > 90 and windsnelheid_km_h > 40) or regen

if slecht_weer:
  print("Iemand zin om te studeren in de bib?")
else:
  print("Iemand zin in een terrasje op de Oude Markt?")
```

Correct gebruik van variabelen in combinatie met goede naamgeving, kan je code veel eenvoudiger maken om te begrijpen.

OVER INDENTATIES EN LEESBAARHEID VAN CODE

In veel talen is indentatie optioneel. Het is aan te raden voor leesbaarheid, maar programma's werken hetzelfde met of zonder indentatie. In plaats daarvan worden speciale karakters gebruikt (meestal accolades `{ }`) om het begin en einde van een blok code aan te duiden.

De volgende twee stukken code doen exact hetzelfde in JavaScript:

```JavaScript
if(6>5){if("a"==="b"){console.log("A")}}else{console.log("B")}
```

```JavaScript
if (6 > 5) {
  if ("a" === "b") {
    console.log("A")
  }
} else {
  console.log("B")
}
```

OVER INDENTATIES EN LEESBAARHEID VAN CODE

In Python is indentatie betekenisvol. Indentatie toevoegen of weglaten kan de uitvoering van een programma beïnvloeden.

Soms zal Python doorhebben dat je indentatie verkeerd is en een `IndentationError` tonen. Maar in veel gevallen **is indentatie je eigen verantwoordelijkheid.**

Correcte en consistente indentatie verhoogt de leesbaarheid van je programma. Dit helpt bij het opsporen van fouten en het delen van code met anderen.

Algemeen advies:

- Besteed steeds aandacht aan indentatie, ongeacht de taal waarin je werkt
- Kies een consistent hoeveelheid voor je indentatie in je hele programma. Veel voorkomende waarden zijn 2 of 4 spaties
- Gebruik de Tab-toets om indentatie aan te passen in je code. Gebruik Shift + Tab om code naar links te verschuiven

QUIZ

Welke variabelen moeten op `True` staan om `2` in de uitvoer te krijgen?

Python

```
if A:
  if B or D:
    print("1")
  elif C:
    print("2")
elif B and A:
  if not D:
    print("3")
  else:
    print("4")
print("5")
```

A en C

B en D

A, B en D

A,C en D

Iets anders

Onmogelijk

QUIZ

Welke variabelen moeten op `True` staan om `4` in de uitvoer te krijgen?

Python

```
if A:
  if B or D:
    print("1")
  elif C:
    print("2")
elif B and A:
  if not D:
    print("3")
  else:
    print("4")
print("5")
```

A en C

B en D

A, B en D

A,C en D

Iets anders

Onmogelijk

QUIZ

Welke variabelen moeten op `True` staan om `5` in de uitvoer te krijgen?

Python

```
if A:
  if B or D:
    print("1")
  elif C:
    print("2")
elif B and A:
  if not D:
    print("3")
  else:
    print("4")
print("5")
```

A en C

B en D

A, B en D

A,C en D

Iets anders

Onmogelijk

COMPUTATIONEEL DENKEN

DEEL 1

ALGORITMISCH DENKEN

WAT IS COMPUTATIONEEL DENKEN?

> *"Computational thinking (CT) involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science."*
>
> — Jeanette M. Wing

WAT IS COMPUTATIONEEL DENKEN (CD)?

- Een vorm van **probleemoplossend denken**
- 'Denken zoals een computerwetenschapper'
- Kan geleerd worden zonder computers

WAAROM NOOD AAN CD?

- Technologie (en met name computers) heeft een grote impact op ons leven
- Tempo van verandering ligt hoog
- Hoe leerlingen voorbereiden op een toekomst waar technologische nieuwigheden de norm zijn?
- De technologieën veranderen, maar de onderliggende principes niet
- CD focust op het begrijpen en toepassen van deze principes

DEELCOMPETENTIES CD

Iconen: CoDe-platform KU Leuven
Link naar de slides waarop deze leerstof gebaseerd is

Abstractie	Onnodige details weglaten
Veralgemening	Principes veralgemenen zodat ze breder toepasbaar zijn
Decompositie	Opdeling in kleinere, eenvoudigere problemen
Algoritmisch denken	Opstellen van een stappenplan voor het oplossen van een probleem
Evaluatie	Kritisch bekijken van oplossing

ALGORITMISCH DENKEN

> *"Het opstellen van een reeks instructies die stap voor stap uitgevoerd moeten worden."*

Een algoritme is een synoniem voor een stappenplan. Voorbeelden van algoritmes zijn:

- Twee matrices vermenigvuldigen
- Chemie-experiment uitvoeren
- Je favoriete gerecht klaarmaken

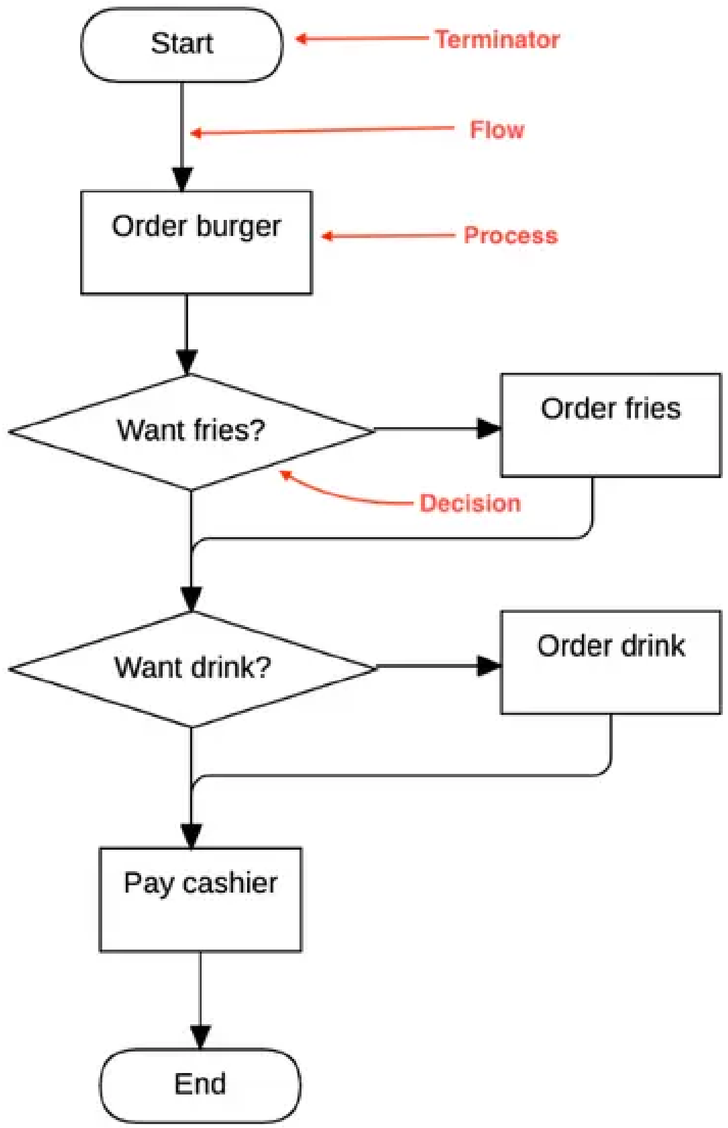
Belangrijk voor CD: het algoritme kan door **een andere persoon (of computer)** uitgevoerd worden. Er is geen ruimte voor interpretatie/vergissingen.

De nadruk bij CD ligt op het **bedenken en opstellen**, niet op het kunnen uitvoeren.

Opdracht: Denk terug aan je voorbije schoolcarrière. Welke algoritmes heb je geleerd? Heb je ze zelf bedacht, of werden ze jou aangeleerd? Hoeveel ruimte is er voor interpretatie?

ALGORITMISCH DENKEN: PRAKTISCH

Algoritmes opstellen is een kunts, maar er zijn technieken om dit aan te leren. Een veelgebruikte methode is het opstellen van een **stroomdiagram** (*Eng: flow chart*).



Je maakt het stroomdiagram **vóór** je begint met code te schrijven.

Een proces ~ een variabele een waarde toekennen of een functie oproepen.

Een beslissing ~ een if-statement of andere selectie.

De vraag in een beslissing ~ een Booleaanse expressie.

ALGORITMISCH DENKEN: PRAKTISCH

Tools om stroomdiagrammen te maken:

- https://app.diagrams.net/
- Microsoft Visio via Office 365
- https://excalidraw.com/
- Pen en papier
- ...

OPDRACHT

Gebruik een van deze tools om een stroomdiagram op te stellen voor de opdracht Deliberatie op Dodona.

VOLGENDE LES

VOLGENDE LES: OPDRACHT (1)

Onderzoek volgende concepten uit les 1 in Kotlin + twee andere programmeertalen uit de lijst van les 1:

- Hoe declareer je een variabele?
- Hoe ken je een variabele een waarde toe?
- Hoe toon je iets in de console, zoals `print()` in Python?
- Hoe vraag je de gebruiker om invoer, zoals `input()` in Python?

VOLGENDE LES: OPDRACHT (2)

Onderzoek selecties en condities in Kotlin + twee andere programmeertalen. Meer specifiek:

- Hoe worden `True` en `False` voorgesteld?
- Hoe worden de logische vergelijkingen (bv. groter dan) en operators (bv. AND) voorgesteld?
- Hoe wordt een if/else if/else structuur geschreven? Maakt de taal gebruik van indentaties?

Per concept:

- Geef aan wat de naam van de functie/methode/... is
- Geef een voorbeeld (1-3 lijnen code) dat toont hoe je ze moet gebruiken
- Geef aan op welke websites je de informatie hebt gevonden. Zoek minstens 2 sites, voeg de URLs toe aan je antwoorden

VOLGENDE LES: OPDRACHT (3)

Dien je gestructureerde en beknopte antwoorden in op Toledo voor de start van volgende les. Ze worden tijdens de les klassikaal besproken.

Schatting tijdsduur: 1u - 1u 30m