Hoofdstuk 12: Lijsten

Na deze les kun je:

- Uitleggen waar lijsten voor gebruikt worden
- Lijsten aanpassen
- Lijst methodes gebruiken: insert(), remove(), index(), count(), sort(), reverse()
- Alle elementen van een lijst doorlopen
- Lijsten nesten (lijsten van lijsten maken)
- Een begin maken met spel boter-kaas-eieren

  (laatste opdracht voor PO Zeeslag)

Havo4 PO Galgje

- Opdracht omschrijving komt op site

- Gebruik hiervoor je kennis over:
  - Lijsten
  - While-loops

VWO4 PO Zeeslag

- Opdracht omschrijving komt op site

- Gebruik hiervoor je kennis over:
  - Lijsten
  - While-loops

Lijsten

- Een "list" (Engelse woord voor "lijst") is een geordende verzameling van data elementen

<u>Voorbeelden:</u>

namenLijst = [“Ben”, “Piet”, “Charlie”, “Dan”, “Edward”]

leeftijdenLijst = [5, 14, 8, 12, 9]

cijferLijst = [8.0, 5.6, 6.0, 8.1,]

- In Python mag je van alles in een lijst stoppen,
  - ook gemengd (bv. getallen en strings)
  - Bv. rommeltje = [3, “banana”, ‘$’, “Hello World!”]

Index van een lijst

- Elk element in een lijst heeft een index (positie).
- Tellen begint altijd bij 0
- Lijst eindigt bij lengte -1 (hier: len(fruits)-1)

Toekennen van waarden aan een lijst

```
fruits = ["Apple", "Mango", "Strawberry", "Banana", "Guava"]
```

index	[0]	[1]	[2]	[3]	[4]
value	"Apple"	"Mango"	"Strawberry"	"Banana"	"Guava"

```
fruits[0] = "Apple"
fruits[1] = "Mango"
fruits[2] = "Strawberry"
fruits[3] = "Banana"
fruits[4] = "Guava"
```

Hoe je lijst er uitziet

Lijsten kun je aanpassen

- Waarden toekennen:

```
fruitlist = ["appel", "banaan"]
```

[“appel”, “banaan”]

- Iets toevoegen aan lijst → lijst wordt langer

```
fruitlist += ["kers"]     #voegt "kers" toe achteraan lijst
```

[“appel”, “banaan”, “kers”]

- Iets in een lijst aanpassen

```
#vervangt element op positie 0 (dus appel) met  aardbei
fruitlist [0] = [“aardbei"]
```

[“aardbei”, “banaan”, “kers”]

- Iets uit lijst verwijderen → lijst wordt korter

```
fruitlist.remove( "banaan" )  #verwijdert eerste voorkomen
```

[“aardbei”, “kers”]

Standaard List functies

numlist = [12, 1, 243, 1]

- Bepaal de grootste getal uit de lijst:

```
max( numlist )                  # levert 243 op
```

- Bepaal de som van getallen in de lijst:

```
sum( numlist )                  # levert 257 op
```

- Bepaal de lengte van de lijst:

```
len ( numlist )                 # levert 4 op
```

- Controleer of getal 999 in de lijst zit

```
999 in numlist                  # levert False op
```

- Tel hoe vaak getal 1 voorkomt

```
numlist.count( 1 )              # levert 2 op
```

- Lijst sorteren

```
numlist.sort()          # sorteert lijst: [1,1,12,243]
```

Lijst handmatig doorlopen.

getallenLijst = [0, 4, 6, 8]

- Druk de eerste element af.
- Druk de tweede element af.
- Druk de derde element af.
- Druk de vierde element af.

```
print( getallenLijst[0] )
print( getallenLijst[1] )
print( getallenLijst[2] )
print( getallenLijst[3] )
```

Stel dat je 100 dingen in je lijst hebt… dan is dit niet handig.

Dit gaan we nu slimmer doen met een **while** loop.

Lijst doorlopen met een while loop

- Schrijf een while-loop die ieder element van de volgende lijst afdrukt:

  getallenLijst = [0, 4, 6, 8]

Tips:

- Maak gebruik van een variabele index die bij 0 begint en steeds na elke element opgehoogd wordt.
- Gebruik len(getallenLijst) om de lengte van de lijst te bepalen.

Herhaling uitleg while loop op volgende pagina.

Terugblik: Standaard opbouw while

Vaak met een teller

De variabele:

1a) Wat is de variabele? Dus wat veranderd er steeds?

1b) Geef de variabele een begin waarde (meestal is dit 0)

De herhaling `while`:

2a) Bepaal de **voorwaarde** voor herhaling: “Zolang **voorwaarde** herhaal … “

2b) Bepaal wat je steeds wilt herhalen, de **acties**.

2c) Verander de variabele (meestal verhogen met 1).

```
getallenLijst = [0, 4, 6, 8]

print( getallenLijst[0] )
print( getallenLijst[1] )
print( getallenLijst[2] )
print( getallenLijst[3] )
```

```
<variabele initialiseren>
while <voorwaarde>:
        <acties>
        <variabele aanpassen>
```

Lijst doorlopen met een while loop

Een oplossing:

```
getallenLijst = [0, 4, 6, 8]

teller = 0

#doorloop elk element in lijst
while teller < len(getallenLijst):
    print ( getallenLijst[teller] ) #druk element af
    teller +=1 #verhoog teller om volgend element te pakken
```

Lijst doorlopen en aanpassen

Stel dit is jouw (dramatische) cijferlijst:

cijferLijst = [8.0, 5.5, 2.3, 4.6, 6.1, 5.6]

- Pas elk cijfer in je cijferlijst aan ze elk met 1 punt opgehoogd wordt.
- Doe dit met een **while** loop.
- Na afloop druk je de cijferLijst af. Dit mag ook met `print( cijferLijst)`
- Het volgende wordt dus afgedrukt: [9.0, 6.5, 3.3, 5.6, 7.1, 6.6]

Tips:

- Gebruik een variabele om de positie van de cijfer in de lijst bij te houden.
- Hoog de cijfer op een bepaalde positie met 1 op.
- Hoog daarna de positie ook op.

Uitbreiding:

- Check dat een cijfer niet hoger dan een 10 uitkomt. Als je na ophoging boven de 10 komt, stel het dan gelijk aan 10.

Lijst doorlopen en aanpassen

Een oplossing:

```
cijferLijst = [8.0, 5.5, 2.3, 4.6, 6.1, 5.6]

positie = 0

while positie < len( cijferLijst ):
      cijferLijst[positie] += 1 #cijfer ophogen met 1
      positie += 1 #positie ophogen om volgende te pakken

print( cijferLijst )
```

Spel: Gok Het Woord

Schrijf een programma voor het volgende spel:

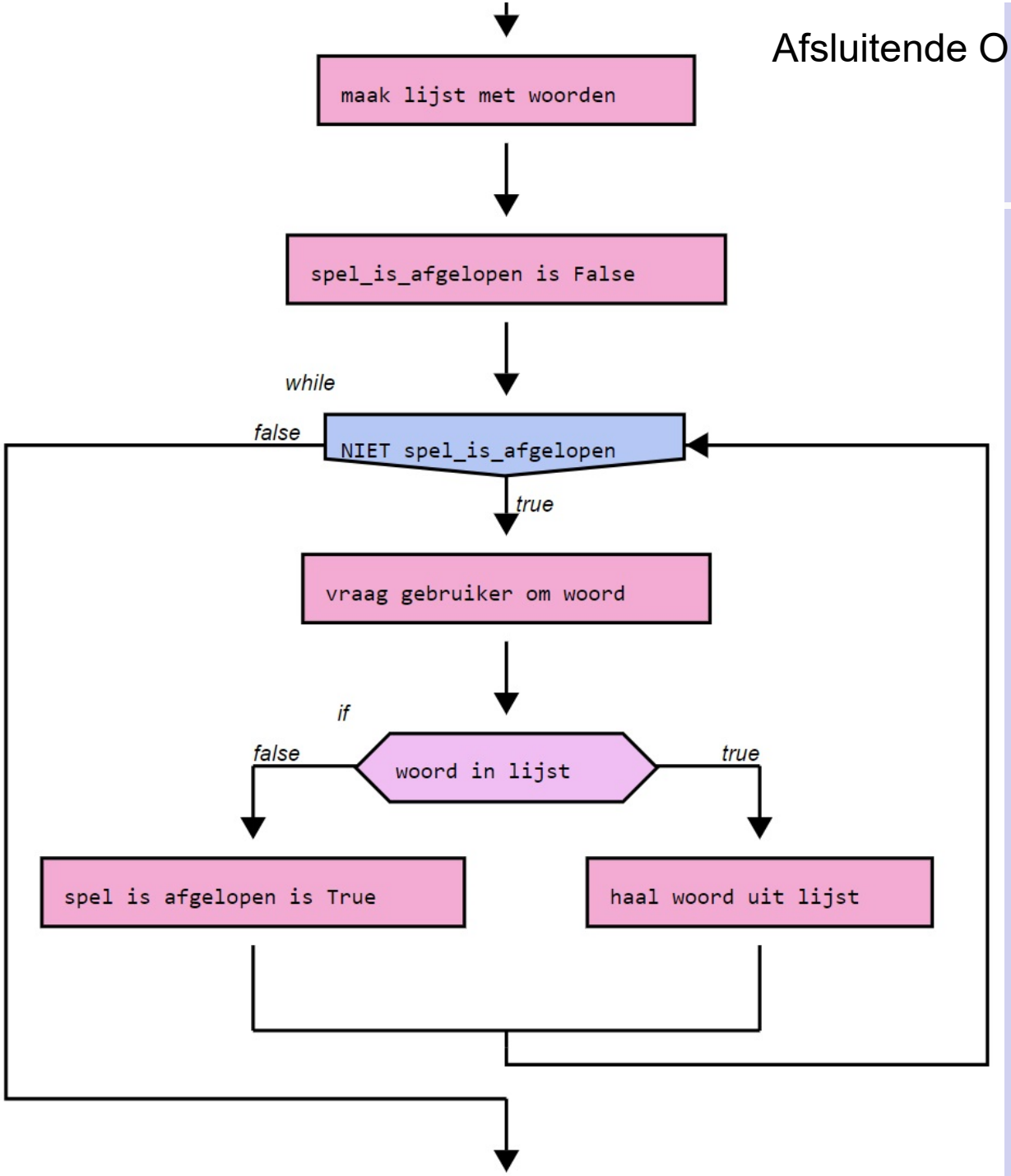
- Maak een lijst met een paar woorden er in
- Vraag de gebruiker om een woord in te typen
- Als het woord voorkomt in de lijst, haal je het woord eruit en mag de gebruiker nog een keer
- Als het woord niet voorkomt in de lijst, is het spel afgelopen.

Tips:

- Teken eerst een stroomdiagram
- In je spel zit herhaling, maak daarom gebruik van een while-loop
- Gebruik een boolean flag, spel_is_afgelopen. Voordat je begint met spelen staat deze op False. Je spel herhaalt zich zolang deze op False staat. Zet het op het juiste moment op True.

Spel

Afsluitende Opgave 12.8



Spel: Gok Het Woord

Oplossing

```
woorden_lijst = ["fiets", "huis"]

spel_is_afgelopen = False  #bij begin: spel nog niet afgelopen

while not spel_is_afgelopen:
  gok = input( "Doe een gok:" ) #vraag gebruiker om een gok
  if gok in woorden_lijst:
       woorden_lijst.remove(gok)
       print(woorden_lijst)  #druk af om te testen
  else:
       spel_is_afgelopen = True
       print("Spel is afgelopen")
```

Afsluiting VWO4

- Planning/opdrachten/sheets: http://course.cs.ru.nl/pythonVO/
- Opdrachten VWO4:
  - Hoofdstuk 12 Lists: Theorie en opgaven
  - Hoofdstuk 12 Lists: Afsluitende opgaven
  - Hoofdstuk 8 Eigen Functies: Afsluitende opgaven
  - Bijwerken evt. andere achterstanden van hoofdstuk 8
- Na de vakantie beginnen we met PO: Galgje
  - Hoofdstukken 8 en 12 zijn daarvoor heel belangrijk!
  - Zorg dat je ze beheerst.

Afsluiting Havo 4

- Planning/opdrachten/sheets: http://course.cs.ru.nl/pythonVO/
- Opdrachten Havo 4:
  - Hoofdstuk 12 Lists: Theorie en opgaven
  - Hoofdstuk 12 Lists: Afsluitende opgaven
- Volgende les beginnen we met PO Galgje
  - Hoofdstukken 7 en 12 zijn daarvoor heel belangrijk!
  - Zorg dat je ze beheerst.