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 met een **while-**loop

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 index van lengte -1 hier: len(fruits)-1

De lijst

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

Hoe voor de computer lijst eruit ziet

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

Toekennen van waarden aan een lijst

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

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]
```

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).

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

Lijst doorlopen en elementen afdrukken (met een while)

getallenLijst = [3, 4, 6, 8]

1. Pak element 0, druk af.
2. Pak element 1, druk af.
3. …
4. Pak element 3, druk af.

Stel dat je 100 dingen in je lijst hebt… dan doe je dit natuurlijk niet handmatig!

Dat doe je slim met een **while** loop.

- Voorwaarde voor herhaling?
  - Zolang positie kleiner is dan aantal elementen in lijst.

Lijst doorlopen met een while loop

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

  getallenLijst = [3, 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.

Terugblik: Standaard opbouw while

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

De variabele:

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

1b) Geef de variabele een beginwaarde (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).

Druk elk element van getallenlijst af:

```
getallenLijst = [3, 4, 6, 8]
```

```
getallenLijst = [3, 4, 6, 8]

positie = 0

#doorloop elk element in de lijst
while __________ :
    _____________#druk element af
    positie += 1 #verhoog
```

Lijst doorlopen met while

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

De variabele:

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

1b) Geef de variabele een beginwaarde (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).

Druk elk element van getallenlijst af:

```
getallenLijst = [3, 4, 6, 8]
```

```
getallenLijst = [3, 4, 6, 8]

positie = 0

#doorloop elk element in lijst
while __________ :
    _____________ #druk element af
    positie += 1  #verhoog
```

Lijst doorlopen met een while loop

Een oplossing:

```
getallenLijst = [3, 4, 6, 8]

positie = 0

while positie < len(getallenLijst):
    print ( getallenLijst[positie] ) #druk element af
    positie += 1 #verhoog 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 )
```