FLOW CONTROL (bis)

→ For loops kunnen gebruikt worden om een codeblok een bepaald aantal keren uit te voeren. Technisch gezien herhaalt een for-lus het codeblok één keer voor elk item in een lijstwaarde.

```
>>> for i in range(4):
        print(i)
```

=

```
>>> for i in [0, 1, 2, 3, 4]:
        print(i)
```

FLOW CONTROL (bis)

for loop & de range() functie

→ Om alle indexen van een lijstwaarde te doorlopen, kunnen de range() en len() functie gecombineerd worden binnen de for loop.

```
>>> materiaal = ['pennen', 'papier', 'nietjes', 'mappen']
>>> for i in range(len(materiaal)):
        print('Index ' + str(i) + ' in materiaal is: ' + materiaal[i])

Index 0 in materiaal is: pennen
Index 1 in materiaal is: papier
Index 2 in materiaal is: nietjes
Index 3 in materiaal is: mappen
```

FLOW CONTROL (bis)

for loop & de enumerate() functie

→ Om de gehele index van de items in de lijst te verkrijgen, kan ook de functie enumerate() gebruikt worden binnen de for loop.

```
>>> materiaal = ['pennen', 'papier', 'nietjes', 'mappen']
>>> for index, item in enumerate(materiaal):
        print('Index ' + str(index) + ' in materiaal is: ' + item)

Index 0 in materiaal is: pennen
Index 1 in materiaal is: papier
Index 2 in materiaal is: nietjes
Index 3 in materiaal is: mappen
```

IN & NOT IN operators

→ Om te bepalen of een waarde zich voordoet binnen een lijst, worden in en not in operatoren gebruikt.

→ Ook deze operatoren bevinden zich tussen twee waarden: de gezochte waarde enerzijds, de lijstwaarde anderzijds.

```
>>> 'howdy' in ['hallo', 'hoi', 'howdy', 'hey']
True

>>> spam = ['hallo', 'hoi', 'howdy', 'hey']
>>> 'hoi' not in spam
False
```

→ Het resultaat van deze expressies is een Booleaanse waarde.

IN & NOT IN operators

→ Bestudeer onderstaand voorbeeld en leg uit wat dit programma doet.

```
huisdieren = ['Soetkin', 'Nala', 'Woef']
naam = input('Noteer de naam van een huisdier:')
if naam not in huisdieren:
    print('Ik heb geen huisdier met de naam ' + naam)
else:
    print(naam + ' is mijn huisdier.')
```

RANDOM module

→ Een module kan beschouwd worden als een soort van ‘mini-bibliotheek’ waarin een aantal (extra) functies opgeslagen zitten. Slechts door deze te importeren in een bestand, kunnen we die specifieke functies gebruiken.

→ De random module staat het toe met willekeurige getallen te kunnen werken .

```
import random
geheimNummer = random.randint(1, 20)
print('Ik heb een nummer tussen 1 en 20 in mijn hoofd.')

# Vraag de speler om 6 keer te raden.
for raadBeurten in range(1, 7):
    print('Raad maar.')
    gok = int(input())
```

RANDOM module

de random.choice() functie

→ De functie random.choice() retourneert een willekeurig geselecteerd item uit een lijst.

```
>>> import random
>>> huisdieren = ['hond', 'kat',
'eland']
>>> random.choice(huisdieren)
'hond'
>>> random.choice(huisdieren)
'eland'
>>> random.choice(huisdieren)
'kat'
```

RANDOM module

de random.shuffle() functie

→ De functie random.shuffle() wijzigt de bestaande lijst; de items binnen de lijst worden op willekeurige wijze van plaats veranderd.

```
>>> import random
>>> mensen = ['Alice', 'Bob', 'Karel', 'David']
>>> random.shuffle(mensen)
>>> mensen
['Karel', 'David', 'Alice', 'Bob']
>>> random.shuffle(mensen)
>>> mensen
['Alice', 'David', 'Bob', 'Karel']
```