


Man Chain illustration by [Frits Ahlefeldt-Laurvig](#) | CC BY-ND

Week 2: Control Flow

Programming (TICT-V1PROG-15)
HBO-ICT propedeuse periode 1 2015-2016



Starten met...

- Formatieve toets...

Week 1: 2



Vervolg van iteraties:

- for-loops
- while-loops
- continue, break, pass

Les 4

3



break

- Stop met waar je nu mee bezig bent:
- Bijv. reis tot Geldermalsen:

```
def reis_tot_geldermalsen(stations):
    for station in stations:
        if station == "Geldermalsen":
            break
        print(station)
```

← stop hier!

```
stations = ["Utrecht Centraal",
            "Geldermalsen", "'s-Hertogenbosch"]
reis_tot_geldermalsen(stations)
```



ander voorbeeld

Blijf net zolang stations vragen en toevoegen aan de lijst, totdat er niks wordt ingevoerd:

```
def vraag_stations():
    stations = []
    while True:
        station = input("Geef station:")
        if station == "":
            break
        stations.append(station)
    print(stations)

vraag_stations()
```

break geldt alleen voor dit while-statement, de rest gaat door (print).



Regels voor break

- Alleen in een loop (dus for of while)
- `break` stopt de huidige "iteratie"
- de rest (buiten de loop) wordt gewoon uitgevoerd



continue

Blijf net zolang stations vragen en toevoegen aan de lijst, totdat er "einde" wordt ingevoerd:

```
def vraagStations():
    stations = []
    while True:
        station = input("Geef een station in:")
        if station == "":
            continue
        elif station == "einde":
            break
        stations.append(station)
    print(stations)

vraagStations()
```

bij een leeg station, ga dan
door met de loop.
bij "einde", stop dan



pass



Poker by Images Money | CC BY



Geen lege loops/functies

- Fout:

```
def stopBijUtrecht():
```

- Melding:

```
File
"/User/PycharmProjects/programming/les4/les4_break_o
pdracht1.py", line 14
    stopBijUtrecht1()
    ^
```

IndentationError: expected an indented block

- Oplossing:

```
def stopBijUtrecht1():
    pass
```



pass is een loop

- ```
for i in range(10):
 pass
```



## pass is nuttig?

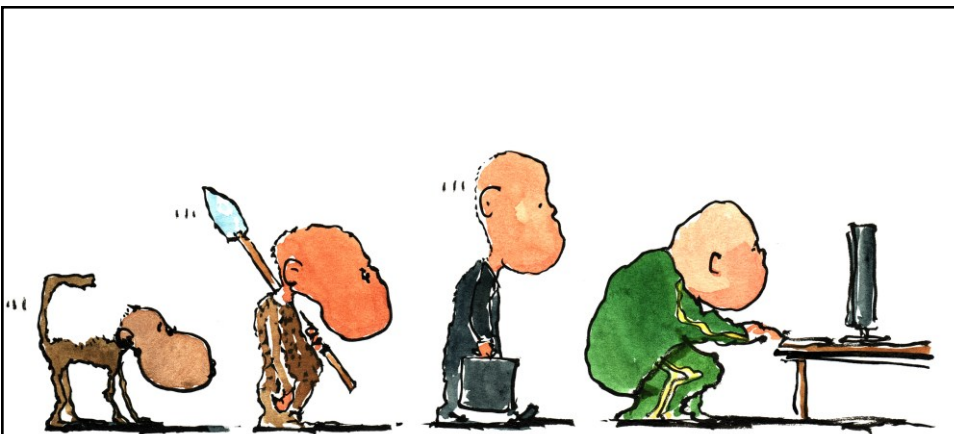
- Met "pass" maak je eerst de structuur van je programma. Daarna vul je de gedefinieerde functies, loops en klassen:

```
def koop_kaartje():
 pass

def zoek_je_trein():
 pass

def treinreis():
 pass

koop_kaartje()
zoek_je_trein()
treinreis()
```



Man Chain illustration by Frits Ahlefeldt-Laurvig | CC BY-ND

## Recursie

Programming (TICT-V1PROG-15)  
HBO-ICT propedeuse periode 1 2015-2016



HOGESCHOOL  
UTRECHT

## Recursie

- Print ieder station in het resterende traject: “Deze trein gaat verder naar Alkmaar, Castricum, Zaandam, ...”
- Vind het grootste getal in de reeks 0, 4, 7, 2, 6, 8, 3, 9
- Tel de getallen 1 t/m n bij elkaar op



[145.365 - May 25, 2010](#) by [Morgan](#) | [CC BY](#)

Les 4

13



## Tel alle getallen 1 t/m n bij elkaar op

- bij  $n = 4$ :  

$$4 + 3 + 2 + 1 = 10$$
- dat is een herhaling, want:  

$$4 + (n-1) + (n-2) + (n-3) = 10$$
- dat kun je programmeren:
  1. neem  $n$
  2. tel daar  $n-1$  bij op
  3. herhaal stap 1 en 2 voor  $n-1$  **tot je 1 bereikt**

Les 4

14



## Nu in code

```
def tel_tm_n_op(n):
 if n == 1:
 return 1
 else:
 return n + tel_tm_n_op(n-1)
```

Les 4

15



## Nu in code

```
def tel_tm_n_op(n):
 if n == 1: Stop bij 1
 return 1
 else:
 return n + tel_tm_n_op(n-1)
 Tel op en herhaal voor n-1
```

Les 4

16





## Nu voor $n = 4$

```
def tel_tm_n_op(n):
 if n == 1: vuurt niet
 return 1
 else:
 return n + tel_tm_n_op(n-1)
 4 + tel_tm_n_op(3)
```

Les 4

17



## Nu voor $n = 4$

```
def tel_tm_n_op(n):
 if n == 1: vuurt niet
 return 1
 else:
 return n + tel_tm_n_op(n-1)
 3 + tel_tm_n_op(2)
```

Les 4

18



## Nu voor $n = 4$

```
def tel_tm_n_op(n):
 if n == 1: vuurt niet
 return 1
 else:
 return n + tel_tm_n_op(n-1)
 2 + tel_tm_n_op(1)
```

Les 4

19



## Nu voor $n = 4$

```
def tel_tm_n_op(n):
 if n == 1: vuurt!
 return 1
 else:
 return n + tel_tm_n_op(n-1)
```

Les 4

20



## Regels bij recursie

|                                      |                            |
|--------------------------------------|----------------------------|
| Er is een eindpunt (stopconditie)    | $n==1$                     |
| Er is een beweging naar het eindpunt | $n-1$                      |
| De functie roept zichzelf aan        | <code>tel_tm_n_op()</code> |

```
def tel_tm_n_op(n):
 if n == 1:
 return 1
 else:
 return n + tel_tm_n_op(n-1)
```

Les 4

21



## Print de resterende stations

```
def resterende_stations(stations):
 if len(stations) == 1: # stopconditie
 print("Dit is station "+ stations[0]+ ".")
 print("Dit is het eindpunt van deze trein.")
 else:
 print("De resterende stations zijn:")
 for station in stations:
 print(station)
 stations = stations[1:] # beweging naar eindpunt
 resterende_stations(stations) # zelf aanroepen
```

```
traject2 = ["Utrecht Centraal",
 "Geldermalsen", "s-Hertogenbosch"]
```

```
resterende_stations(traject2)
```

De resterende stations zijn:  
 Utrecht Centraal  
 Geldermalsen  
 's-Hertogenbosch  
 De resterende stations zijn:  
 Geldermalsen  
 's-Hertogenbosch  
 Dit is station 's-Hertogenbosch.  
 Dit is het eindpunt van deze trein.

Les 4

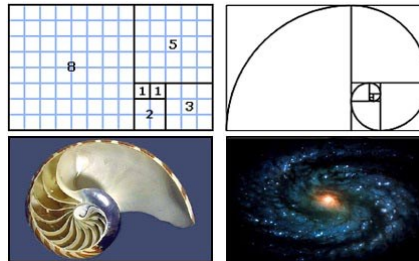
22



## No geen voorbeeld: De Fibonacci-reeks

- Fibonacci-reeks: 1, 1, 2, 3, 5, 8, 13, 21, ...
- Tel de twee laatste cijfers bij elkaar op om een nieuw cijfer te krijgen, begin met "1" en "1":

$$\begin{aligned}
 1 + 1 &= 2 \\
 1 + 2 &= 3 \\
 2 + 3 &= 5 \\
 3 + 5 &= 8 \\
 5 + 8 &= 13
 \end{aligned}$$



<https://www.fibonacci.com/nl/rij-van-fibonacci>

Les 4

23



## De Fibonacci-reeks in code

```
def fibonacci(n):
 voorlaatste = 1
 laatste = 1
 print(voorlaatste)
 print(laatste)

 for i in range(n-2):
 nieuw = laatste + voorlaatste
 print(nieuw)
 voorlaatste = laatste
 laatste = nieuw
```

n-2, immers we kennen de eerste twee cijfers (1,1) al.

Les 4

24



## De Fibonacci-reeks in code

```
def fibonacci(n):
 voorlaatste = 1
 laatste = 1
 print(voorlaatste)
 print(laatste)

 for i in range(n-2):
 nieuw = laatste + voorlaatste
 print(nieuw)
 voorlaatste = laatste
 laatste = nieuw

fibonacci(6)
```

1  
1  
2  
3  
5  
8

Les 4

25



## De Fibonacci-reeks in code, maar nu met recursie

```
def fibonacci_rekursief(n):
 if n == 1:
 return 1
 elif n == 2:
 return 1
 else:
 return fibonacci_rekursief(n-1) +
fibonacci_rekursief(n-2)

for i in range(6):
 print(fibonacci_rekursief(i+1))
```

1  
1  
2  
3  
5  
8

Les 4

26



## Meer over recursie

- Perkovic Hoofdstuk 10
- <http://interactivepython.org/courselib/static/python-nds/Recursion/recursionsimple.html>
- [http://www.python-course.eu/python3\\_recursive\\_functions.php](http://www.python-course.eu/python3_recursive_functions.php)
- <http://openbookproject.net/thinkcs/python/english3e/recursion.html>

Les 4

27



## Tentamenvraag 2

- Wat voor type functie is dit en wat is de uitvoer bij tel(3)?

```
def tel(i):
 if i <= 0:
 print("Klaar!")
 else:
 print(i, sep=" ")
 tel_af(i-1)
```

- A: Iteratief: 3 2 1 Klaar!
- B: Recursief: 3 2 1 0 Klaar!
- C: Iteratief: 3 2 1 Klaar!
- D: Recursief: 3 2 1 0 Klaar!

28



## Opdracht

Maak thuis de opdrachten die horen bij:

- Week 2: Control Flow | College 2
- Week 2: Control Flow | Weekopdracht