



# Python

---

Informatica

Renske Smetsers



# Hoofdstuk 5: eenvoudige functies

---

Leerdoelen:

- ❑ Wat functies zijn
- ❑ Functie namen, parameters en retourwaardes
- ❑ Type casting functies: `float()`, `int()`, en `str()`
- ❑ Basis berekeningen met `abs()`, `max()`, `min()`, `pow()` en `round()`, `len()`, `input()`
- ❑ Gebruik van modules
- ❑ De math functie `sqrt()`
- ❑ De random functies `random()`, `randint()` en `seed()`
- ❑ De pcinput functies `getInteger()`, `getFloat()`, `getString()`, en `getLetter()`



# Herhaling: Toewijzingen en berekeningen

---

1) `aantal_appels` is 5. Wat is `aantal_appels` na:

`aantal_appels = aantal_appels + 3`

`aantal_appels` wordt dan 8

2) `aantal_appels` is 5. Wat is de waarde van `aantal_appels` na:

`aantal_fruit = aantal_appels`

`aantal_fruit += 1`

`aantal_fruit` wordt dan 6

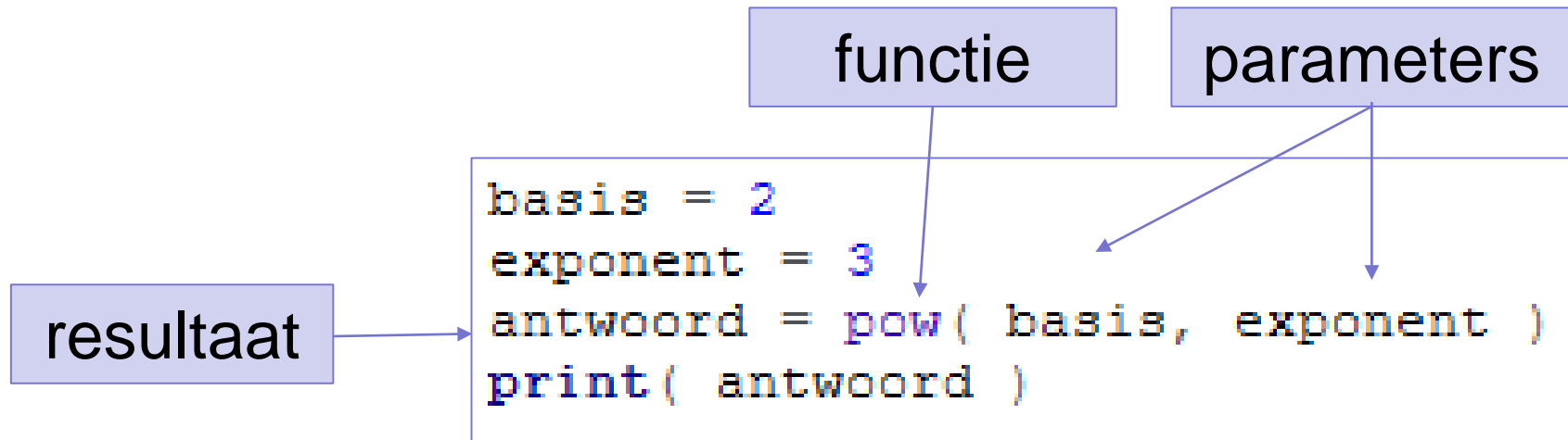
3) Schrijf code om het `aantal_vliegen` met 1 te verlagen

`aantal_vliegen -= 1`



# Functies, parameters, resultaten

Voorbeeld: **pow** is de machtsfunctie



Het programma vult in:

- 2 in plaats van `basis`
- 3 voor `exponent`

Dus `pow(basis, exponent)` wordt `pow(2,3)`

dus  $2^3 = 8$ , dus antwoord krijgt de waarde 8

Er wordt dus afgedrukt: 8



# Standaard berekening

---

- **abs**( ): absoluut, zet negatieve getal om in positieve
- **pow**( ): geeft de eerste getal tot de macht van de tweede getal
- **round**( ): rond een eerste getal af op een gegeven aantal decimalen


| Functie                 | Resultaat |
|-------------------------|-----------|
| <b>abs</b> ( -5 )       | 5         |
| <b>pow</b> ( 2, 3 )     | $2^3 = 8$ |
| <b>round</b> ( 2.34, 1) | 2.3       |



# Standaard berekening

- **max**( ): geeft de grootste van een aantal getallen terug
- **min**( ): geeft de kleinste van een aantal getallen terug
- **len**( ): geeft de lengte van een woord terug  
of aantal dingen in een lijst

| Functie                                 | Resultaat |
|---|-----------|
| <code>max( 3, 6, -2 )</code>            | 6         |
| <code>min( -1, 4 )</code>               | -1        |
| <code>len( "hoi" )</code>               | 3         |
| <code>len( ["appel", "banaan"] )</code> | 2         |



# Modules

---

- ❑ Modules worden gebruikt om code te groeperen
- ❑ Nodig om grote programma's overzichtelijk te houden
- ❑ De code staat in een ander bestand
  
- ❑ Sommige modules zijn ingebouwd in Python
  - *turtle*: op te tekenen
  - *random*: om willekeurige getallen te maken
- ❑ Sommige kun je downloaden
  - *pcinput*: om gebruikers invoer mogelijk te maken



# Module turtle (deze ken je)


- Module kun je importeren (koppelen) met **import**
  - bv. import **turtle**
- De functie zelf roep je daarna aan met **modulenaam.**
  - bv. **turtle.forward(150)**

```
import turtle          # importeer turtle graphics module

turtle.pendown()       # zet pen neer
turtle.forward(150)    # 150 stappen vooruit

turtle.done()          # klaar
```





# Module math


---

- ▣ **sqrt( )**: neemt de wortel van een getal

- ▣ Voorbeeld:

```
import math  
  
wortel = math.sqrt( 4 )  
print (wortel)
```

```
Drukt af: 4
```



# Module random

---

- ❑ Module kun je importeren (koppelen) met **import**
  - bv. import **random**
- ❑ De functie zelf roep je daarna aan met **modulenaam**.
  - bv. **random.randint()**

```
import random  
  
print( "Een toevalsgetal tussen 1 en 10 is", random.randint( 1, 10 ) )
```

**randint()** is een functie dat in  
een module **random** staat


Levert een  
willekeurige getal  
van 1 t/m 10 op



# Module pinput: Gebruikers invoer

---

- Python heeft een basisfunctie `input()`
- Deze heb je al gebruikt, maar is niet zo handig
  - De invoer wordt altijd als tekst gezien
  - Als je een getal wilt vragen, moet je eerst:
    - Controleren of een getal is ingevoerd
    - Casten van string naar getal
- **Beter:** Er is een `module pinput` die dit voor je regelt
  - Deze moet in dezelfde map als je opdracht staan




# Module pcinput

---

- ❑ **getInteger( )**: vraag gebruiker om een geheel getal in te voeren
- ❑ **getFloat( )**: vraag om een kommagetal
- ❑ **getString( )**: vraag om een woord
- ❑ **getLetter( )**: vraag om een letter
  
- ❑ Voorbeeld:

```
import pcinput  
  
getal1 = pcinput.getInteger( "Geef een geheel getal: " )
```

- ❑ Foutmelding als je geen getal invert.



# Hergebruik

---

- ❑ Gebruik je dezelfde code een paar keer?
- ❑ Niet copy-pasten!
- ❑ Maak gebruik van **functies** en **parameters**



# Functies definiëren en gebruiken

```
import turtle

turtle.pendown()

# tekenen van een vierkant
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)

turtle.done()
```

```
import turtle

#Definitie: instructies voor een vierkant
def vierkant():
    turtle.forward(50)
    turtle.right(90)
    turtle.forward(50)
    turtle.right(90)
    turtle.forward(50)
    turtle.right(90)
    turtle.forward(50)

turtle.pendown()
vierkant() #Aanroep: teken de vierkant!

turtle.done()
```

# Functies gebruiken

```
import turtle
```

```
#Definitie: instructies voor vierkant
```

```
def vierkant():
```

```
    turtle.forward(50)
    turtle.right(90)
    turtle.forward(50)
    turtle.right(90)
    turtle.forward(50)
    turtle.right(90)
    turtle.forward(50)
```

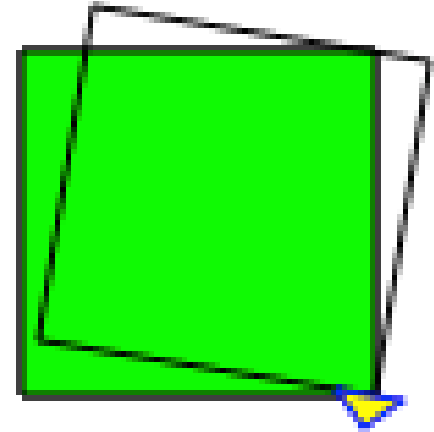
```
turtle.pendown()
```

```
vierkant() #Aanroep: teken vierkant!
```

```
turtle.right(30)
```

```
vierkant() #Aanroep: teken vierkant!
```

```
turtle.done()
```



## Voordeel van functies:

- Je kunt iets tekenen ZONDER bezig te zijn met details
- hergebruik: veel minder code!
- Aanpasbaar op 1 plek



# Parameters gebruiken

```
import turtle

turtle.pendown()

# tekenen van een vierkant
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)

turtle.done()
```

```
import turtle
```

```
#Definitie: instructies voor een vierkant
# met een gegeven lengte
```

```
def vierkant( lengte ):
    turtle.forward( lengte )
    turtle.right(90)
    turtle.forward( lengte )
    turtle.right(90)
    turtle.forward (lengte )
    turtle.right(90)
    turtle.forward (lengte )
```

```
turtle.pendown()
```

```
#Aanroep: teken vierkant met lengte 50
vierkant( 50 )
```

```
turtle.done()
```





# Let op programma verloop!

