# Fouten en foutopsporing

**Programmeurs zijn niet perfect en in de meeste programma's zitten in het begin nog fouten. Die staan bekend als 'bugs' en het opsporen ervan wordt 'debugging' genoemd.**


**ZIE OOK**

❮ **94–95** Fouten

❮ **122–123** Loops in Python

Wat nu? **176–177** ❯


## Soorten bugs

In programma's kunnen drie soorten bugs opduiken: syntax-, looptijd- en logische fouten. Sommige zijn makkelijk te ontdekken, andere moeilijker, maar er zijn manieren om ze op te sporen en te herstellen.

Het sleutelwoord in Python is 'for', niet 'fir'

```
fir i in range(5):
print(i)
```

Dit levert een foutmelding op, want geen enkel getal is deelbaar door 0

```
a = 0
print(10 / a)
```

Leeftijd kan niet tegelijk lager zijn dan 5 en hoger dan 8, dus geen gratis kaartjes

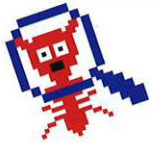
```
if age < 5 and age > 8:
    print('Free ticket!')
```

**△ Eenvoudig op te sporen**
Een syntaxfout is een fout in de woorden of symbolen, zoals tikfouten, ontbrekende haken of verkeerde inspringingen.

**△ Moeilijker op te sporen**
Looptijdfouten verschijnen alleen als het programma bezig is. Mogelijke oorzaken: getallen bij een string optellen of door 0 te delen.

**△ Moeilijkst op te sporen**
Logische fouten zijn fouten in de gedachtegang van het programma. Als je bijvoorbeeld '<' instead of '>' gebruikt of optelt als je moet aftrekken, krijg je deze fouten.

## Een bug vinden en herstellen



Syntaxfouten zijn gemakkelijk te vinden omdat IDLE ze in rood highlight als je programma loopt. Het opsporen van looptijd- en logische fouten kost wat meer tijd.

**1 Probleemprogramma**
Dit programma wil alle getallen vanaf 1 optellen bij de waarde in de variabele 'top_num'. Vervolgens wordt het totaal geprint.

```
top_num = 5
total = 0
for n in range(top_num):
    total = total + n
print('Sum of numbers 1 to', top_num, 'is', total)
```

Het hoogste getal in de reeks van opgetelde getallen

Dit commando print een zin om de gebruiker het resultaat te laten weten

**2 Output** Het antwoord voor het programma moet zijn (1 + 2 + 3 + 4 + 5), maar het getoonde antwoord is '10'. Je moet nu uitzoeken waarom.

```
Sum of numbers 1 to 5 is 10
```

Het antwoord moet zijn '15', niet '10'



**3 Een 'print' en 'input()' toevoegen**
Het programma toont niet wat het per stap doet. Als je hier een 'print'-commando toevoegt, kun je zien wat er gebeurt. Het commando 'input' wacht totdat de entertoets wordt ingedrukt alvorens te beginnen.

```
top_num = 5
total = 0
for n in range(top_num):
    total = total + n
    print('DEBUG: n=', n, 'total=', total)
    input()
print('Sum of numbers 1 to', top_num, 'is', total)
```

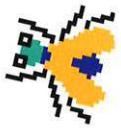
Met dit commando print je de huidige waarde van de loop-variabele en het totaal op dat moment

**4 Nieuwe output**
De loop telt alleen de getallen van 0 tot 4 op en niet die van 1 tot 5. Dit komt doordat een 'for'-loop altijd begint met tellen vanaf 0 (tenzij anders geïnstrueerd) en altijd 1 voor het einde van de reeks ophoudt.

```
DEBUG: n= 0 total= 0
DEBUG: n= 1 total= 1
DEBUG: n= 2 total= 3
DEBUG: n= 3 total= 6
DEBUG: n= 4 total= 10
Sum of numbers 1 to 5 is 10
```

Dit is eigenlijk de som van de getallen 0 t/m 4, niet van 1 t/m 5

**5 De foute regel herstellen**
De reeks moet oplopen van 1 tot 'top_num + 1', zodat de loop de getallen optelt van 1 tot 'top_num' (5).



```
top_num = 5
total = 0
for n in range(1, top_num + 1):
    total = total + n
    print('DEBUG: n=', n, 'total=', total)
    input()
print('Sum of numbers 1 to', top_num, 'is', total)
```

De nieuwe reeks telt vanaf 1 en stopt bij 'top_num' (1 minder dan 'top_num + 1')

**6 Correcte output**
Het 'print'-commando toont dat het programma getallen van 1 tot 5 toevoegt en het juiste antwoord krijgt. De fout is verholpen!

```
DEBUG: n= 1 total= 1
DEBUG: n= 2 total= 3
DEBUG: n= 3 total= 6
DEBUG: n= 4 total= 10
DEBUG: n= 5 total= 15
Sum of numbers 1 to 5 is 15
```

Als 'n= 3' is het totaal (1 + 2 + 3)

Het juiste antwoord wordt nu geprint

# Algoritmes

**Een algoritme is een reeks instructies om een taak uit te voeren. Algoritmes kun je gebruiken voor ingewikkelde opdrachten, maar ook voor eenvoudige, zoals een lijst getallen sorteren.**

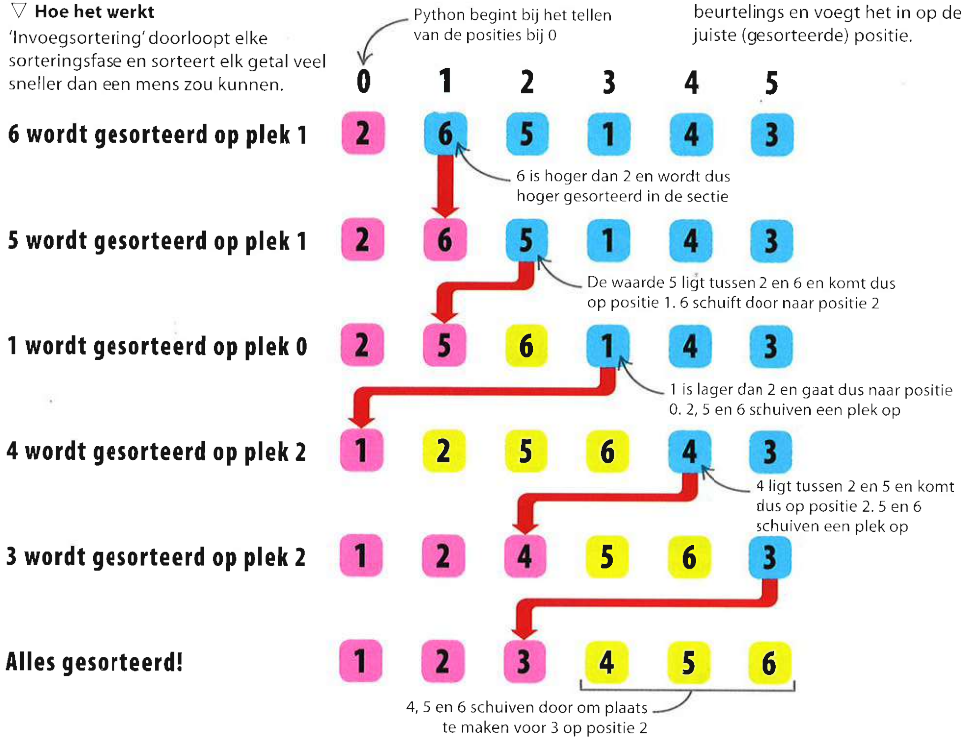
ZIE OOK
‹ 16–17 Denk als een computer
Bibliotheken 152–153 ›

## Invoegsortering

Stel je voor dat je de proefwerken van je klas op volgorde van het laagste naar het hoogste cijfer moet leggen. 'Invoegsortering' maakt een gesorteerde sectie boven aan de stapel en voegt dan elk ongesorteerd proefwerk op de juiste plaats in.

△ **Sorteren op volgorde**
'Invoegsortering' pakt elk proefwerk beurtelings en voegt het in op de juiste (gesorteerde) positie.

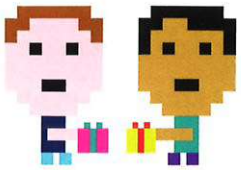
▽ **Hoe het werkt**
'Invoegsortering' doorloopt elke sorteringsfase en sorteert elk getal veel sneller dan een mens zou kunnen.



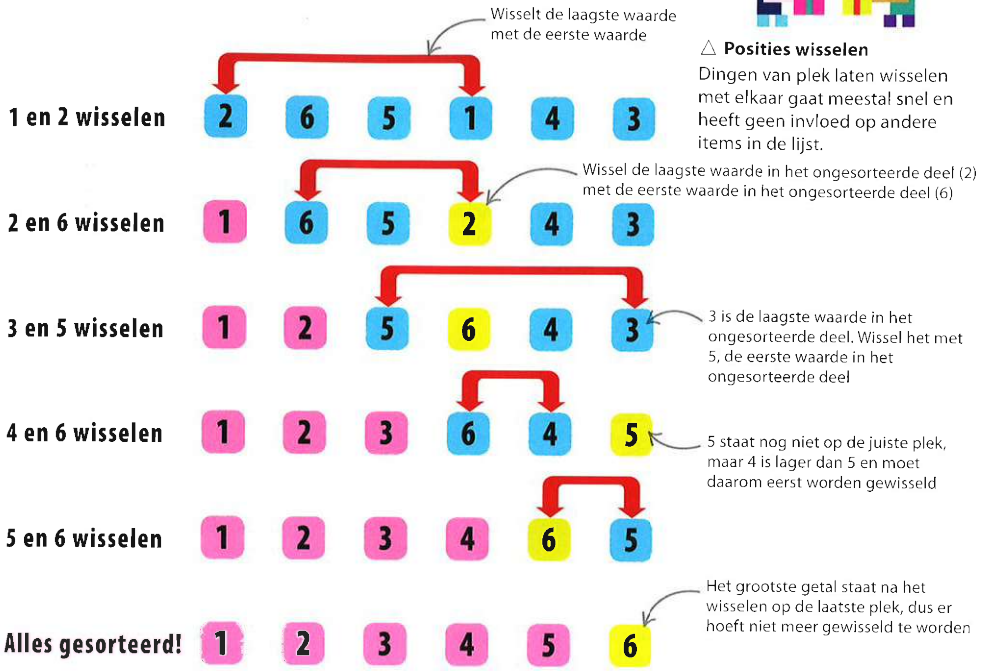



## Selectiesortering

'Selectiesortering' werkt anders dan 'invoegsortering'. Het wisselt itemparen in plaats van telkens alle items door te schuiven. Elke wisseling zet een getal op zijn definitieve (gesorteerde) positie.



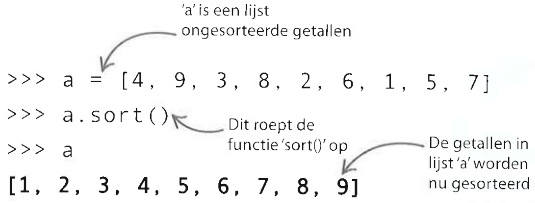
△ **Posities wisselen**
Dingen van plek laten wisselen met elkaar gaat meestal snel en heeft geen invloed op andere items in de lijst.




TIPS VAN EXPERTS

### Sorteren in Python

Er zijn veel verschillende vormen van sorteren, elk met sterke en zwakke punten. Pythons 'sort()'-functie werkt met het algoritme 'Timsort', vernoemd naar zijn ontwerper, Tim Peters. Het is gebaseerd op twee sorterings-algoritmes, 'Invoegsortering' en 'Samenvoegend sorteren'. Typ deze code in om te zien hoe het werkt.



# Bibliotheken

**Het kost tijd een nieuwe codering te schrijven, dus is het handig als je delen van andere programma's kunt hergebruiken. Deze coderingsfragmenten kunnen worden gedeeld via pakketjes die 'bibliotheken' worden genoemd.**

ZIE OOK
Vensters maken 154–155 ❯
Kleur en coördinaten 156–157 ❯

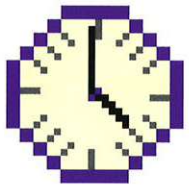
## 'Standard Library'-modules

Python heeft een standaardbibliotheek (Standard Library) waarin veel handige coderingsfragmenten klaar staan voor gebruik. Je kunt stand-alone secties van een bibliotheek, zogenaamde 'modules', aan Python toevoegen om het nog krachtiger te maken.

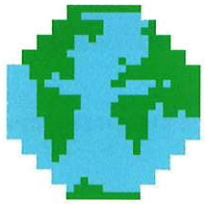
◁ **Inclusief batterijen**
Pythons motto is 'inclusief batterijen'. Dat betekent dat het veel klaar-voor-gebruik-codering levert.

◁ **Random**
Deze module kan een willekeurig getal selecteren of een lijst in een willekeurige volgorde zetten.

▽ **Turtle**
Deze module wordt gebruikt om lijnen en vormen te tekenen op het scherm.

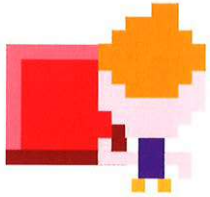


△ **Time**
Deze module geeft de actuele tijd en datum weer en kan data berekenen, bijvoorbeeld welke dag het over drie dagen is

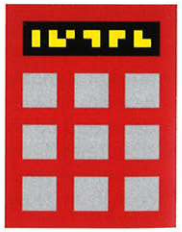


△ **Socket**
Met de code in deze module kun je computers met elkaar verbinden via netwerken en internet.

▽ **Tkinter**
Tkinter wordt gebruikt om knoppen, vensters en andere tekeningen te maken die de gebruiker helpen met programma's om te gaan.



▷ **Math**
Met de wiskunde-module (Math) kun je werken met ingewikkelde wiskundige berekeningen.





## Modules importeren

Voordat je een module kunt gebruiken, moet je je computer vertellen de module te importeren, zodat je hem kunt toepassen in je programma. Zo kunnen de coderingsfragmenten erin voor jou toegankelijk worden. Modules importeren doe je via het commando 'import'. Python kan modules op een paar manieren importeren.

```
import random

random.randint(1, 6)
random.choice(my_list)
```

De modulenaam komt vóór elke functie

◁ **'import random'**
Deze manier van importeren vereist dat je de modulenaam intypt aan het begin van de codering. Dat maakt het lezen ervan makkelijker, omdat je weet van welke module de codering afkomstig is.

▷ **'from random import *'**
Een module als deze importeren werkt goed bij kleine programma's, maar kan verwarrend zijn bij grotere, omdat het niet duidelijk is bij welke module de functie hoort.

Importeert alle functies vanuit de Random-module

```
from random import *

randint(1, 6)
choice(my_list)
```

Deze codering toont niet van welke module de functie afkomstig is

Importeert alleen de functie 'randint'

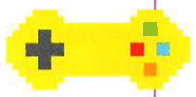
```
from random import randint

randint(1, 6)
```

Alleen de functie 'randint' is beschikbaar

◁ **'from random import randint'**
Je kunt een enkele functie importeren vanuit de module. Dit kan efficiënter zijn dan de hele module importeren als het de enige functie is die je wilt gebruiken.

TIPS VAN EXPERTS

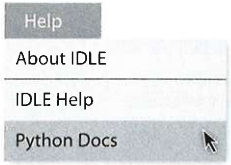
### Pygame



Pygame is een Python-bibliotheek die is ontworpen voor het schrijven van videogames. Pygame geeft je toegang tot geluidsmodules en speciale graphics die je in je games kunt toepassen. Je kunt Pygame pas gebruiken als je de basis van Python onder de knie hebt.

## Hulp en documentatie

Via de Python Library Reference kom je te weten hoe je een module moet gebruiken en welke functies beschikbaar zijn. Klik simpelweg op de bibliotheek waarvan je meer wilt weten. Het is handig om de bibliotheken, modules en beschikbare functies te kennen, dan ben je minder tijd kwijt aan het schrijven van coderingen die al bestaan.




◁ **Help!**
Boven aan elk IDLE-venster kun je klikken op 'Help' en 'Python Docs' kiezen. Er verschijnt een venster waarin je veel nuttige info kunt vinden.

# Vensters maken

**Veel programma's hebben vensters en knoppen om die te besturen. Deze vormen de 'graphical user interface' (grafische gebruikersinterface), of 'GUI' (spreek uit: 'goey').**

ZIE OOK
Kleur en coördinaten 156–157 ❯
Vormen maken 158–159 ❯
Dingen veranderen 160–161 ❯

## Maak een eenvoudig venster

De eerste stap bij het maken van een GUI is een venster maken waarin alles wordt weergegeven. Met Tkinter (uit Python's Standard Library) kun je een eenvoudige maken.

**1 Voer de codering in**

Deze code importeert Tkinter uit de bibliotheek en maakt een nieuw venster. Je moet Tkinter importeren voordat je het venster kunt gebruiken.

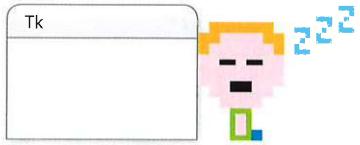
```
from tkinter import *
window = Tk()
```

Dit importeert Tkinter uit de bibliotheek

Dit maakt een Tkinter-venster

**2 Een Tkinter-venster verschijnt**

Voer de code uit en er verschijnt een venster. Het ziet er nog een beetje saai uit, maar het is nog maar het eerste deel van je GUI.




## Knoppen toevoegen aan het venster

Maak de GUI interactiever door er knoppen aan toe te voegen. Er zal een ander bericht verschijnen als de gebruiker op een van de knoppen klikt.

**1 Maak twee knoppen aan**

Schrijf deze code om een eenvoudig venster met twee knoppen te maken.

```
from tkinter import *
def bAaction():
    print('Thank you!')
def bBaction():
    print('Ouch! That hurt!')
window = Tk()
buttonA = Button(window, text='Press me!', command=bAaction)
buttonB = Button(window, text='Don\'t press!', command=bBaction)
buttonA.pack()
buttonB.pack()
```

Dit bericht verschijnt als je op knop A klikt

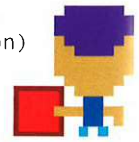
Dit bericht verschijnt als je op knop B klikt

Dit label verschijnt op knop A

Dit vertelt het programma welke functie het moet starten als je op de knop klikt

Deze code vertelt de computer de knoppen in het venster te zetten

Dit label verschijnt op knop B





**2 Klik op de knoppen om berichten te printen**

Als het programma loopt, verschijnt een venster met twee knoppen. Klik op de knoppen en er verschijnen verschillende berichten in het Shell-venster. Je hebt nu een interactieve GUI gemaakt die de gebruikerscommando's beantwoordt.


## Rol de dobbelsteen

Met Tkinter kun je een GUI bouwen voor een eenvoudige applicatie. Met onderstaande codering maak je een programma dat een rollende dobbelsteen nadoet.

**1 Maak een dobbelsteen-simulator**

Dit programma maakt een knop aan die, als je erop klikt, de functie 'roll()' vertelt een willekeurig getal tussen 1 en 6 weer te geven.

```
from tkinter import *
from random import randint
def roll():
    text.delete(0.0, END)
    text.insert(END, str(randint(1,6)))
window = Tk()
text = Text(window, width=1, height=1)
buttonA = Button(window, text='Press to roll!', command=roll)
text.pack()
buttonA.pack()
```

Dit importeert de functie 'randint' vanuit de willekeurige bibliotheek

Deze code wist de tekst in het tekstvenster en vervangt hem door een willekeurig getal tussen 1 en 6

Maakt een tekstvenster om het willekeurige getal in weer te geven

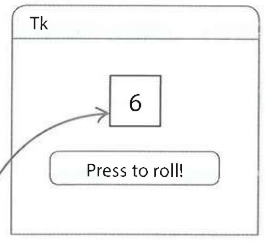
Dit vertelt het programma welke functie moet starten als je op de knop klikt

Dit label verschijnt op de knop

Dit plaatst het tekstvenster en de knop in het venster

**2 Druk op de knop om de dobbelsteen te rollen**

Start het programma, klik op de knop om de dobbelsteen te rollen en kijk naar het resultaat. Je kunt het programma zo aanpassen dat het een dobbelsteen met 12 zijden simuleert of een munt die wordt opgegooid.




Elke keer als je op de knop klikt, verschijnt hier een ander getal

**TIPS VAN EXPERTS**

### Duidelijk en eenvoudig

Als je een GUI ontwerpt, probeer de gebruiker dan niet te overvoeren door het scherm te vullen met te veel knoppen. Label elke knop met een duidelijke naam om de applicatie makkelijk te begrijpen te maken.

# Kleur en coördinaten

**Afbeeldingen en tekeningen op een computerscherm zijn opgebouwd uit heel kleine, gekleurde puntjes: pixels. Als je tekeningen wilt maken, moet je de computer precies vertellen welke kleur elke pixel moet hebben.**

ZIE OOK
« 154–155 Vensters maken
Vormen 158–159 » maken
Dingen 160–161 » veranderen

## Kleuren selecteren

Het is belangrijk de kleuren te beschrijven op een voor de computer te begrijpen manier. Tkinter heeft een handig hulpmiddel om je hierbij te helpen.

**1 Installeer de kleurenselector**

Typ onderstaande code in het Shell-venster om de Tkinter-kleurenselector te installeren.

Dit importeert alle Tkinter-functies

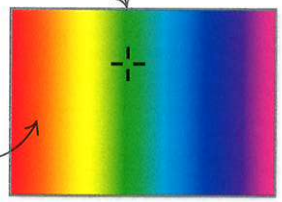
```
>>> from tkinter import *
>>> t = Tk()
>>> colorchooser.askcolor()
```

Gebruik de Amerikaanse schrijfwijze 'color'

**2 Kies een kleur**

Het venster 'color chooser' verschijnt. Kies de gewenste kleur en klik op de knop 'OK'.

Selecteer de gewenste kleur door erop te klikken



Met dit venster kun je makkelijk precies de gewenste kleur kiezen

**3 Kleurwaarden**

Als je een kleur hebt geselecteerd zal in het Shell-venster een lijst met getallen verschijnen. Deze getallen zijn de waarden van rood, groen en blauw, die gemengd de gekozen kleur opleveren.

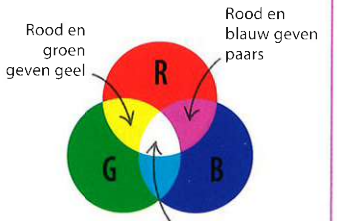
```
((60.234, 190.742, 52.203), '#3cbe34')
```

Roodwaarde — Groenwaarde — Blauwwaarde — Code voor de kleur in hexadecimalen (zie p. 182-183)

TIPS VAN EXPERTS

### Lichtkleuren mengen

Elke pixel kan rood, groen en blauw licht verspreiden. Door deze kleuren te mengen kun je elke voorstelbare kleur maken.





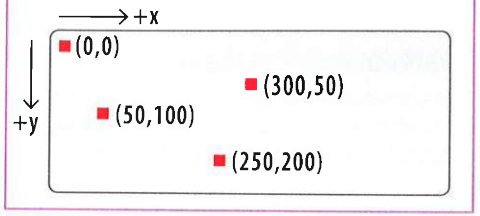

## Tekenen op een canvas

Om graphics te maken met Python moet je een leeg scherm aanmaken om in te tekenen. Dit wordt een canvas genoemd. Via x- en y-coördinaten kun je Python precies vertellen waar het op het canvas moet tekenen.

TIPS VAN EXPERTS

### Coördinaten

In Tkinter worden x-coördinaten hoger als je naar rechts gaat en y-coördinaten als je naar beneden gaat. Het punt (0,0) ligt linksboven in de hoek.




**1 Maak een tekenprogramma**

Met deze code kun je een venster aanmaken waarin je een canvas kunt plaatsen. Het zal willekeurig cirkels tekenen op het canvas.

```
from random import *
from tkinter import *
size = 500
window = Tk()
canvas = Canvas(window, width=size, height=size)
canvas.pack()
while True:
    col = choice(['pink', 'orange', 'purple', 'yellow'])
    x0 = randint(0, size)
    y0 = randint(0, size)
    d = randint(0, size/5)
    canvas.create_oval(x0, y0, x0 + d, y0 + d, fill=col)
    window.update()
```

Dit importeert de functies 'randint' en 'choice' vanuit de Random-module

Dit importeert alle Tkinter-functies

De variabele 'size' bepaalt de afmetingen van het canvas

Maakt een canvas binnen een venster

Door een eeuwige loop zal het programma eindeloos cirkels blijven tekenen

Kiest een willekeurige kleur uit de lijst

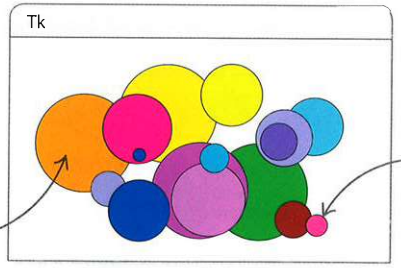
Maakt een cirkel met een willekeurig formaat aan op een willekeurige plek op het canvas

Dit deel van de regel tekent de cirkel

Dit deel vult het met de gekozen kleur ('col')

**2 Gekleurd canvas**

Start de code en het programma begint cirkels te tekenen op het canvas.




Op willekeurige plekken worden cirkels getekend

Elke cirkel heeft een willekeurige afmeting

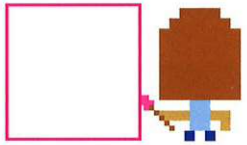
# Vormen maken

**Je kunt Tkinter niet alleen gebruiken om vensters, knoppen en kleuren toe te voegen aan een grafische gebruikers-omgeving (GUI), maar ook om vormen mee te tekenen.**

ZIE OOK
Dingen veranderen 160–161
Reageren op events 162–163

## Basisvormen maken

Met rechthoeken en ovalen kun je van alles tekenen. Als je een canvas hebt gemaakt, kun je er met de volgende functies vormen in tekenen.



```
>>> from tkinter import *
>>> window = Tk()
>>> drawing = Canvas(window, height=500, width=500)
>>> drawing.pack()
>>> rect1 = drawing.create_rectangle(100, 100, 300, 200)
>>> square1 = drawing.create_rectangle(30, 30, 80, 80)
>>> oval1 = drawing.create_oval(100, 100, 300, 200)
>>> circle1 = drawing.create_oval(30, 30, 80, 80)
```

Maakt een canvas om in te tekenen

Bepaalt de vorm van het canvas

Bepaalt plaats en omvang van de rechthoek via coördinaten (zie hierna)

Tekent een rechthoek

Je kunt een vierkant maken door alle zijden even lang te tekenen

Tekent een cirkel

Bepaalt plaats en omvang van de cirkel

## Tekenen met coördinaten

Coördinaten dienen om de computer precies te vertellen waar hij vormen moet maken. Aan het eerste getal ('x') ziet de computer waar hij op het scherm moet beginnen, door het tweede getal ('y') weet hij hoe ver hij naar beneden moet.

```
>>> drawing.create_rectangle(50, 50, 250, 350)
```

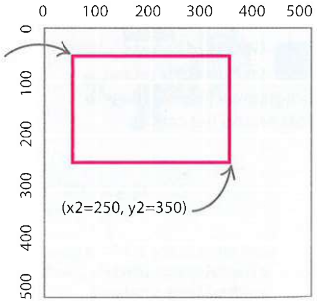
Dit is de naam van het canvas

Coördinaten voor de linkerbovenhoek

Coördinaten voor de hoek rechtsonder

△ **Instellen van de coördinaten**
De eerste twee getallen zijn de coördinaten voor de hoek linksboven, de andere twee getallen bepalen de plaats van de hoek rechtsonder.

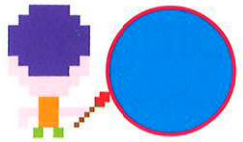
▽ **Coördinatendiagram**
De linkerbovenhoek van de rechthoek ligt op de coördinaten (50, 50), de hoek rechtsonder op (250, 350).






## Kleuren geven aan vormen

Je kunt ook vormen in kleur maken. Met codes kun je kleuren gebruiken voor de omtrek van je vorm en je kunt elke vorm ook opvullen ('fill').



Zorgt voor een volle blauwe cirkel met een rode rand

```
>>> drawing.create_oval(30, 30, 80, 80, outline='red', fill='blue')
```

## Teken een alien

Door vormen met elkaar te combineren kun je bijna alles tekenen. Hier vind je een paar aanwijzingen waarmee je een alien kunt tekenen met ovalen, lijnen en driehoeken.

**1 Maak de alien**
Voor elk onderdeel van de alien moet je de soort vorm, omvang, positie op het canvas en de kleur bepalen. Elke vorm heeft een uniek ID-nummer, dat je in een variabele kunt opslaan.

```
from tkinter import *
window = Tk()
window.title('Alien')
c = Canvas(window, height=300, width=400)
c.pack()
body = c.create_oval(100, 150, 300, 250, fill='green')
eye = c.create_oval(170, 70, 230, 130, fill='white')
eyeball = c.create_oval(190, 90, 210, 110, fill='black')
mouth = c.create_oval(150, 220, 250, 240, fill='red')
neck = c.create_line(200, 150, 200, 130)
hat = c.create_polygon(180, 75, 220, 75, 200, 20, fill='blue')
```

Stelt 'Alien' in als titel van het venster

Maakt het canvas

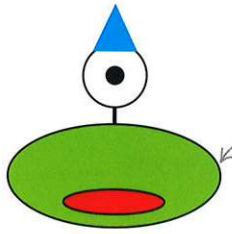
Tekent een groene ovaal voor het lichaam

Tekent een zwarte stip in het oog

Tekent een rode ovaal voor de mond

Tekent een blauwe driehoek voor de hoed van de alien

**2 Ontmoet je alien**
Voer de codering in om de alien te tekenen. Hij heeft een groen lichaam, een rode mond en één oog op een steel. En hij draagt een prachtige blauwe hoed.



Je alien is klaar!

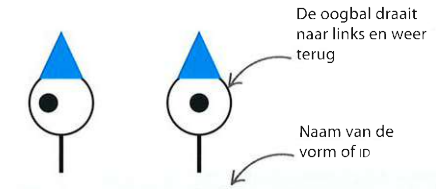
# Dingen veranderen

**Je afbeelding op het canvas hoeft niet altijd gelijk te blijven. Met coderingen kun je het uiterlijk van de graphic veranderen of hem over het scherm bewegen.**

ZIE OOK
❮ 158–159 Vormen maken
Reageren op 162–163 ❯ events

## Vormen bewegen

Als je een vorm wilt bewegen op het canvas, moet je de computer vertellen wat hij moet bewegen (de naam of ID die je de vorm hebt gegeven) en waarheen.




```
>>> c.move(eyeball, -10, 0)
>>> c.move(eyeball, 10, 0)
```

Deze functie beweegt vormen

Bepaalt coördinaten voor de beweging

ONTHOUD

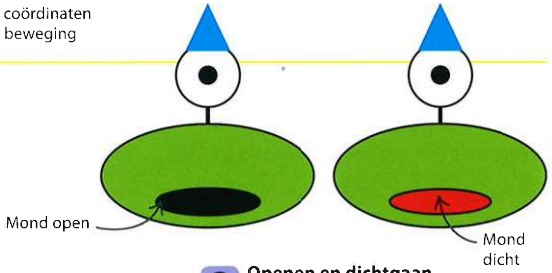
### Veelzeggende namen

Je kunt het beste duidelijke namen nemen waaraan je de vormen op het canvas herkent. Op deze pagina's gebruiken we namen als 'oogbal' ('eyeball') en 'mond' ('mouth'), dus kun je de code makkelijk lezen en begrijpen.

◁ **Bewegende oogballen**
Typ deze codering in het Shell-venster om de oogbal naar links te bewegen en weer terug.

## Kleuren veranderen

Als je wilt dat het lijkt of de mond open en dicht gaat, moet je simpelweg de kleur van de ovaal veranderen.




**1 Schrijf de codering**
Typ deze codering om twee functies te maken waarmee de mond open en dicht lijkt te gaan.

De functie 'itemconfig()' verandert de eigenschappen van vormen die je al hebt getekend

De geopende mond is zwart

```
def mouth_open():
    c.itemconfig(mouth, fill='black')
def mouth_close():
    c.itemconfig(mouth, fill='red')
```

De ID van de vorm

De gesloten mond is rood

**2 Openen en dichtgaan**
Typ deze code in het Shell-venster om de mond open en dicht te laten gaan.

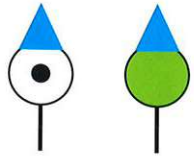
```
>>> mouth_open()
>>> mouth_close()
```

Voer deze commando's in om de alien zijn mond open en dicht te laten doen



## Verbergen en tonen

Je kunt vormen verbergen met de functie 'itemconfig()'. Als je de oogbal verbergt en even later weer toont, lijkt het alsof de alien knipoogt.



◁ **Knipogende alien**
Om de alien te laten knipogen, moet je de pupil verbergen en het oogwit groen maken.

**1 Knipogende functies maken**
Met deze code maak je twee functies, waardoor de alien kan knipogen.

Maakt het oogwit groen

De ID van de vorm

```
def blink():
    c.itemconfig(eye, fill='green')
    c.itemconfig(eyeball, state=HIDDEN)
def unblink():
    c.itemconfig(eye, fill='white')
    c.itemconfig(eyeball, state=NORMAL)
```

Maakt de ogen weer wit

Verbergt de pupil

Toont de pupil weer

**2 Knipogen**
Typ deze code in het Shell-venster om de alien te laten knipogen.

```
>>> blink()
>>> unblink()
```

Met het commando 'unblink()' lijkt het oog weer gewoon open

## Dingen zeggen

Je kunt ook tekst in het scherm invoeren om de alien te laten praten. Je kunt hem zelfs laten antwoorden op gebruikerscommando's.


**1 Tekst toevoegen**
Deze code voegt tekst toe aan de afbeelding van de alien en maakt een functie waarmee je zijn hoed steelt.

Positioneert de tekst op het canvas

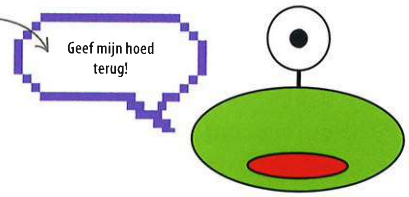
Zet wat je de alien wilt laten zeggen tussen aanhalingstekens

```
words = c.create_text(200, 280, text='I am an alien!')
def steal_hat():
    c.itemconfig(hat, state=HIDDEN)
    c.itemconfig(words, text='Give my hat back!')
```

Dit verbergt de hoed

Zodra de hoed verdwijnt, zal de alien vragen hem terug te geven

Als de hoed verdwijnt, verschijnt een nieuw bericht




**2 Steel de hoed**
Typ deze code in het Shell-venster en zie wat er gebeurt.

Typ dit om de hoed te stelen

```
>>> steal_hat()
```

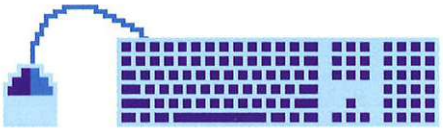
# Reageren op events

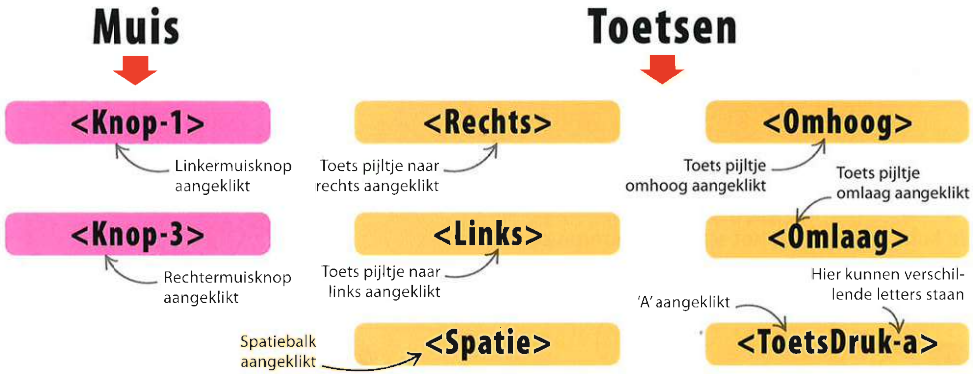
ZIE OOK
158–159 Vormen maken
160–161 Dingen veranderen

**Computers ontvangen een signaal als je op een toets drukt of met de muis beweegt. Dit heet een event of 'gebeurtenis'. Met programma's kun je de computer instrueren om te reageren op elke waargenomen gebeurtenis.**

## Muis- en toets-gebeurtenissen

Veel gebeurtenissen kun je triggeren door apparaten te gebruiken, zoals de muis en het toetsenbord.



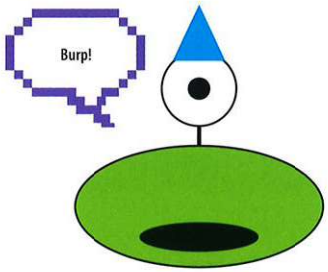



## Muis-gebeurtenissen

Om een programma te laten reageren op een muishandeling kun je een functie gewoon linken (of binden) aan een gebeurtenis. Hier wordt de functie 'burp' aangemaakt en gekoppeld aan de gebeurtenis '<Button-1>'.

```
window.attributes('-topmost', 1)
def burp(event):
    mouth_open()
    c.itemconfig(words, text='Burp!')
c.bind_all('<Button-1>', burp)
```

Dit brengt het Tkinter-venster vooraan op je scherm
Maakt een functie genaamd 'burp'
Linkt klikken op de linkermuisknop aan de 'burp'-functie




△ **Burpende alien**
Als je op de linkermuisknop klikt, laat je de alien burpen. Dit komt doordat je de functie 'burp' gebruikt.

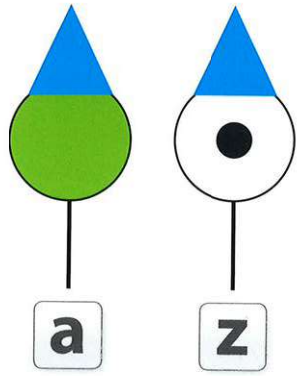


## Toets-gebeurtenissen

Op dezelfde manier kun je ook toetsen op het toetsenbord verbinden aan een gebeurtenis. Met onderstaande code laat je de alien knipogen als je de toetsen 'A' en 'Z' indrukt.

```
def blink2(event):
    c.itemconfig(eye, fill='green')
    c.itemconfig(eyeball, state=HIDDEN)
def unblink2(event):
    c.itemconfig(eye, fill='white')
    c.itemconfig(eyeball, state=NORMAL)
c.bind_all('<KeyPress-a>', blink2)
c.bind_all('<KeyPress-z>', unblink2)
```

Maakt de ogen groen (gesloten)
Verbergt de oogbal
Toont de oogbal
Deze codering linkt functies aan gebeurtenissen
Dit bindt de functie 'unblink2' aan de 'Z'-toets



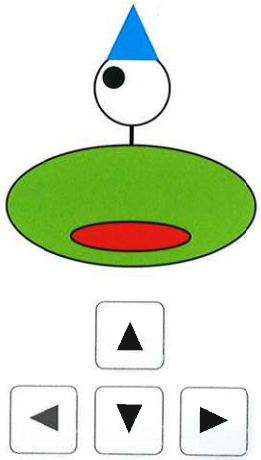

△ **Laat de alien knipogen**
Als deze codering actief is, laat de 'A'-toets de ogen sluiten en gaan ze weer open met de 'Z'.

## Bewegen via toetsen

Door op een toets te drukken kun je ook beweging uitlokken. Deze codering bindt de pijltjestoetsen aan de functies die de oogbal van de alien laten bewegen.

```
def eye_control(event):
    key = event.keysym
    if key == "Up":
        c.move(eyeball, 0, -1)
    elif key == "Down":
        c.move(eyeball, 0, 1)
    elif key == "Left":
        c.move(eyeball, -1, 0)
    elif key == "Right":
        c.move(eyeball, 1, 0)
c.bind_all('<Key>', eye_control)
```

Deze regel zoekt de naam van de toets die ingedrukt wordt
De oogbal beweegt omhoog als de pijtjestoets omhoog wordt ingedrukt
De oogbal gaat naar links als je op de linker pijltjestoets drukt
Activeert de functie 'eye_control' als je op een toets drukt



△ **Oogbalbesturing**
De oogbal beweegt in de richting van de toets die je indrukt.