Programmeren met Python

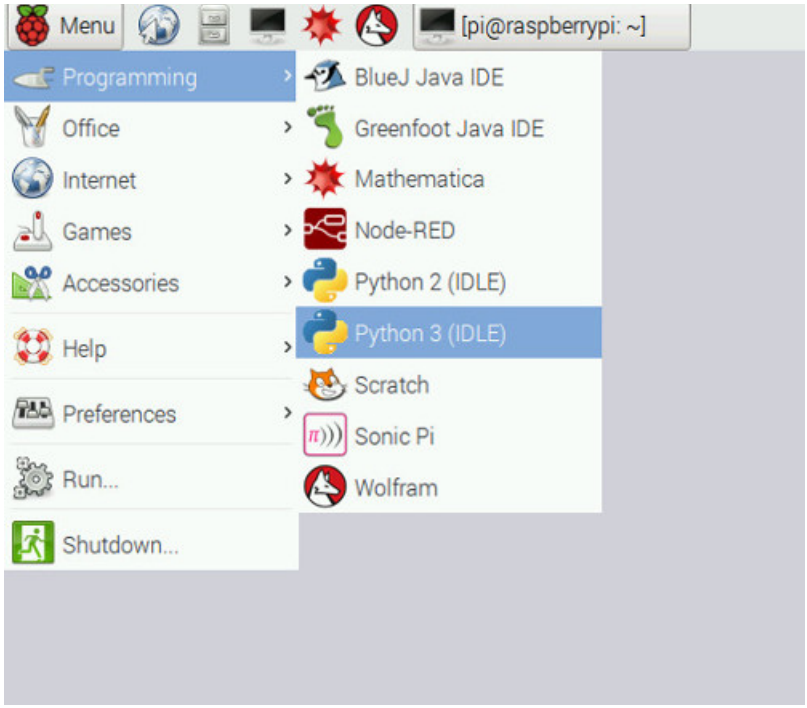
Deze instructies zijn online te vinden op https://codingkids.nl/python-lessen.html

Python is voor kinderen een leuke programmeertaal om mee te beginnen. Het is leuk om te beginnen met de Turtle Graphics, dit maakt het programmeren heel visueel. Voordat je met Python begint is het wel handig om eerst wat ervaring met Scratch te hebben.

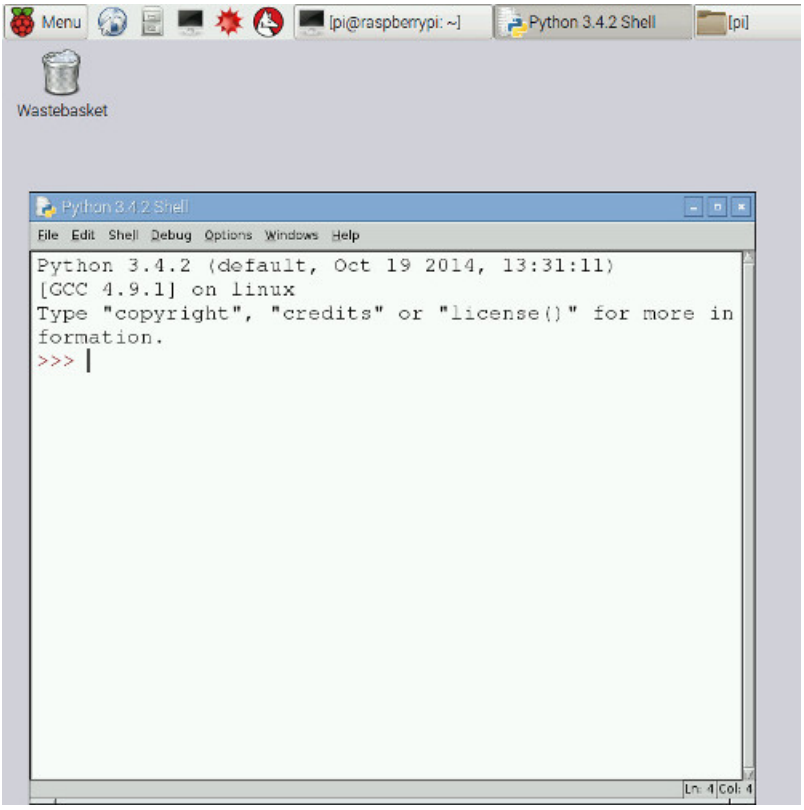
In deze Python lessen wordt gebruik gemaakt van Python 3!

Vierkant tekenen

Open Python 3 op je bureaublad.

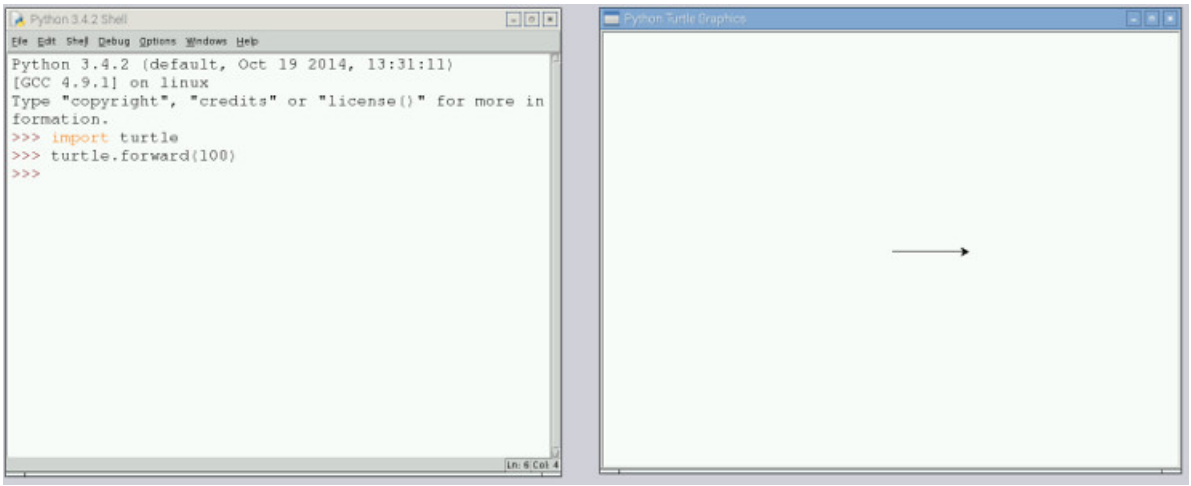


Het scherm wat je nu ziet heet de Python Shell.



In de Python Shell kun je opdrachten geven achter de prompt >>>. Kijk naar het voorbeeld hieronder en typ de volgende twee regels in:

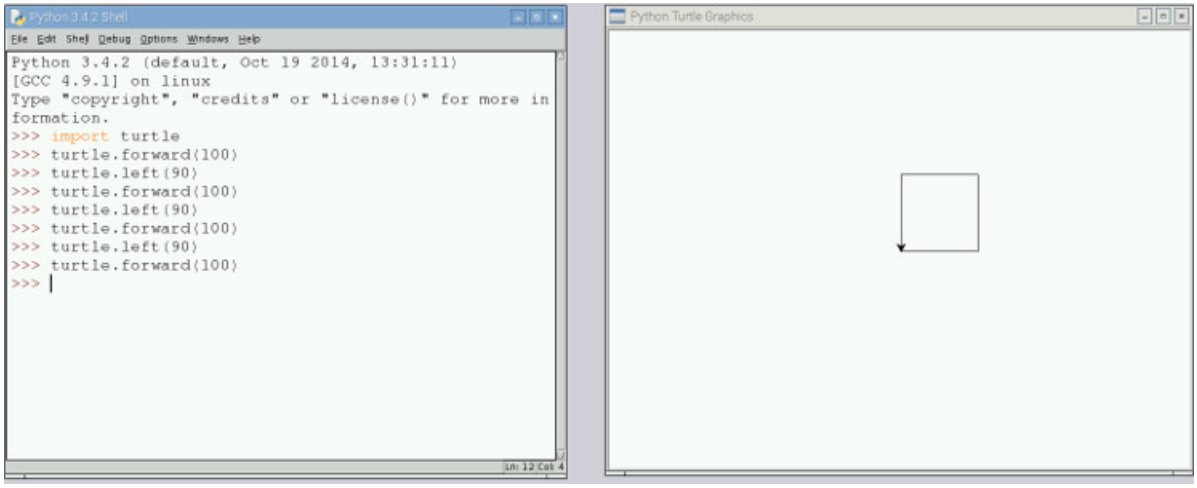
```
import turtle
turtle.forward(100)
```



Je ziet nu dat er nog een venster geopend wordt. Dit is je tekenblaadje van Turtle Graphics. De lijn die je ziet is 100 pixels. We gaan nu proberen een vierkantje te maken. Hiervoor heb je deze opdracht regel nodig, je draait dan 90 graden linksom:

```
turtle.left(90)
```

Herhaal `turtle.forward(100)` en `turtle.left(90)` totdat je een vierkantje hebt zoals het voorbeeld hieronder.



Om je tekenblaadje weer leeg te maken typ je:

```
turtle.clear()
```

For loop

We gaan nu het programma om een vierkant te tekenen wat korter opschrijven. Dan lijkt het al meer op echt programmeren. We gebruiken daarvoor een **for loop**. Een loop is een herhaling of lus en is bedoeld voor het herhalen van stukjes code. We beginnen de loop met deze regel:

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

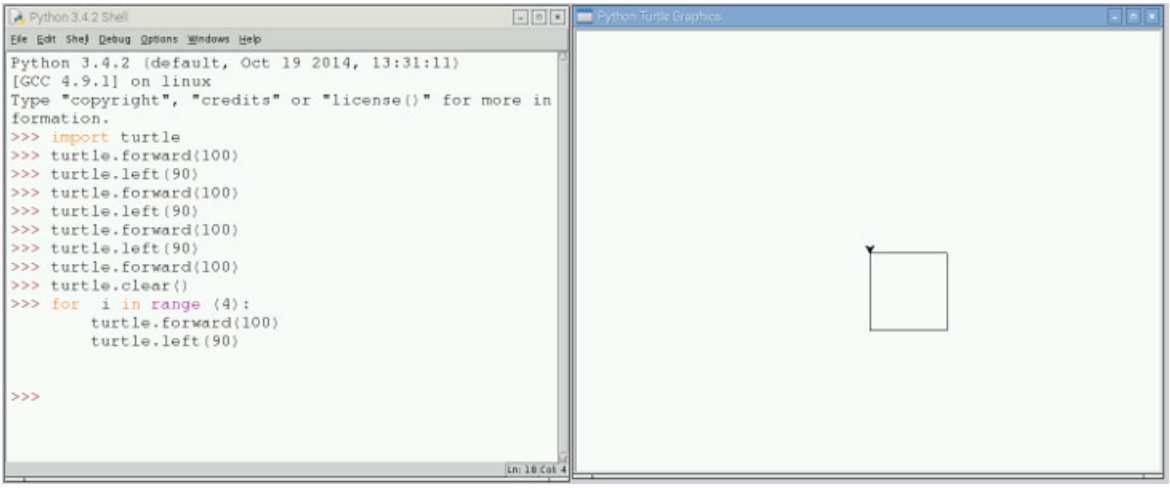
De `i` in deze regel is een variable. Je mag daarvoor ook een andere letter of zelf een woordje bedenken. `in range (4)` betekent dat de regels die onder de **for loop** staan, vier keer herhaald worden, van 0 t/m 3. Vergeet niet de dubbele punt aan het eind `:` Over de functie `range()` krijg je in de volgende les meer informatie.

Om te zien dat de herhaling van 0 t/m 3 gaat kun je even deze twee regels intypen:

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

Nu gaan we het programma schrijven voor het maken van een vierkant.

```
for i in range (4):
  turtle.forward(100)
  turtle.left(90)
```



Opslaan

We hebben de opdrachten tot nu toe in de Python Shell getypt. Je kunt ook een nieuw blaadje openen en daar je programma in schrijven en dit opslaan. Klik in het menu bovenin op **File** en dan op **New File**, je krijgt nu een leeg blaadje waar je het programma in kunt typen. Kijk naar het voorbeeld hieronder:

```
import turtle
for i in range (4):
    turtle.forward(100)
    turtle.left(90)
```

Om het programma op te slaan klik je op **File** en dan op **Save** en dan zie je allemaal mappen, dubbelklik op **Documents** en typ bij **File name** vierkant en klik op **Save**.

Druk nu op je toetsenbord op **F5**, het Turtle Graphics scherm opent nu en als het goed is komt er een vierkant.

Wiskunde

De hoek van een vierkant is altijd 90 graden. Maar waarom eigenlijk? Kunnen we het berekenen? Ja dat kan met de formule 360 / 4. Een circel is 360 graden en we gaan het verdelen in 4 hoeken. Een hoek is dan 90 graden. We kunnen ook een veelhoek programmeren. Willen we bijvoorbeeld een 6 hoekig figuur, dan moeten we 360 graden delen door 6 (360/6). We hoeven dit niet zelf uit te rekenen, dat kan Python ook. Ons Python programma wordt dan:

```
import turtle
for i in range (6):
    turtle.forward(100)
    turtle.left(360/6)
```

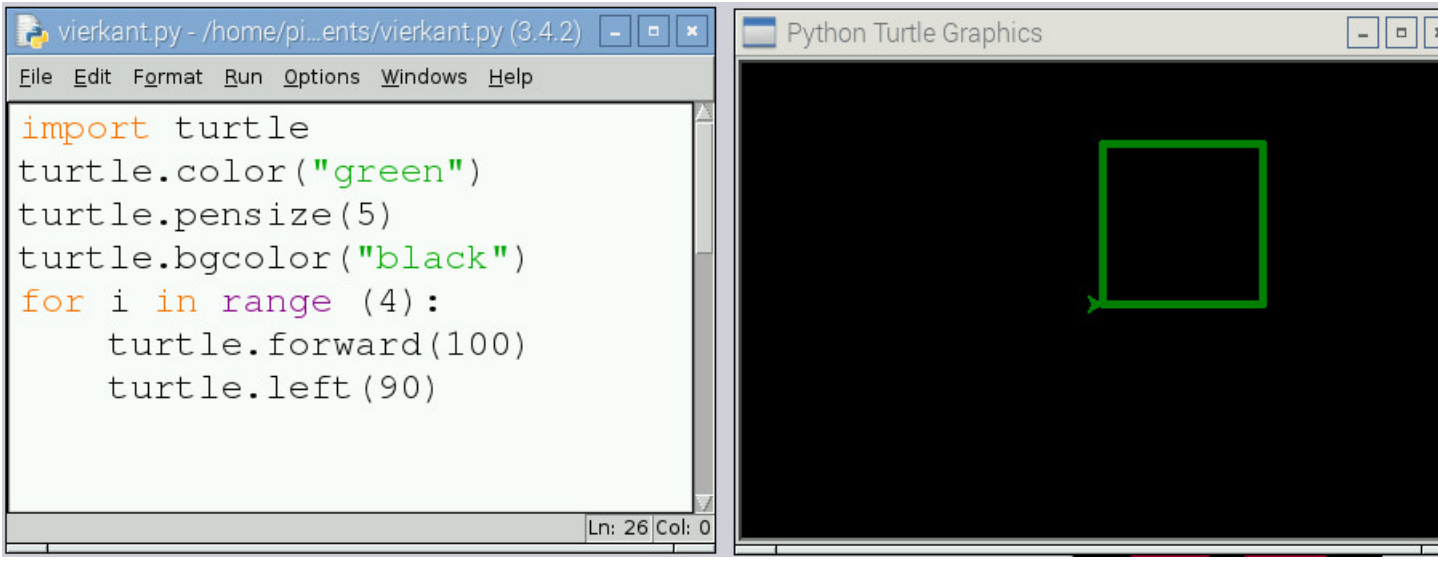
Opdracht: maak een achthoek

Nu je weet hoe je een 6 hoekig figuur maakt kun je ook een 8 hoekig figuur maken. Probeer het maar eens in Python.

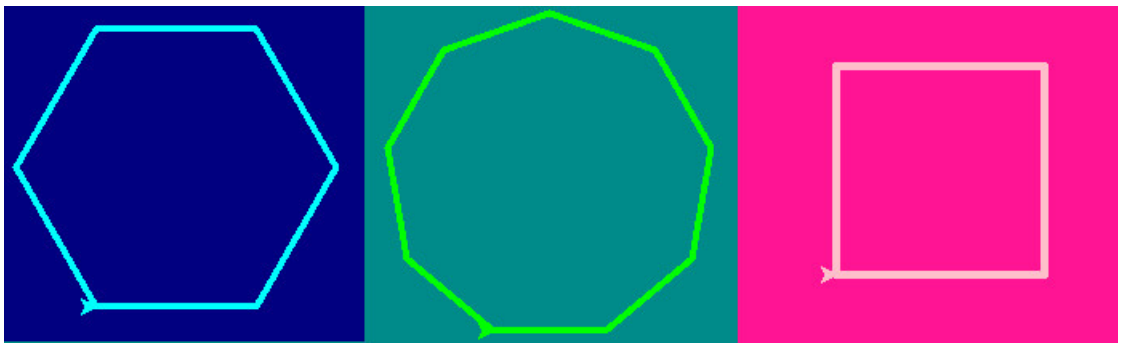
Pimp je vierkant of veelhoek

We gaan nu het vierkant of je veelhoek wat mooier maken, met kleurtjes en dikke lijnen. We voegen daarvoor onderstaande 3 regels toe:

```
turtle.color("green")
turtle.pensize(5)
turtle.bgcolor("black")
```



Druk op **F5** om je programma uit te voeren. Sla het wel eerst op. Probeer nog meer kleuren en veelhoek vormen. Kijk voor de kleuren namen op de afbeelding onderaan de pagina.



Red colors	Green colors	Blue/Cyan color	Yellow colors
IndianRed	GreenYellow	Aqua	Gold
LightCoral	Chartreuse	Cyan	Yellow
Salmon	LawnGreen	LightCyan	LightYellow
DarkSalmon	Lime	PaleTurquoise	LemonChiffon
LightSalmon	LimeGreen	Aquamarine	LightGoldenrodYellow
Red	PaleGreen	Turquoise	PapayaWhip
Crimson	LightGreen	MediumTurquoise	Moccasin
FireBrick	MediumSpringGreen	DarkTurquoise	PeachPuff
DarkRed	SpringGreen	CadetBlue	PaleGoldenrod
Pink colors	MediumSeaGreen	SteelBlue	Khaki
Pink	SeaGreen	LightSteelBlue	DarkKhaki
LightPink	ForestGreen	PowderBlue	**Purple colors**
HotPink	Green	LightBlue	Lavender
DeepPink	DarkGreen	SkyBlue	Thistle
MediumVioletRed	YellowGreen	LightSkyBlue	Plum
PaleVioletRed	OliveDrab	DeepSkyBlue	Violet
Orange colors	Olive	DodgerBlue	Orchid
LightSalmon	DarkOliveGreen	CornflowerBlue	Fuchsia
Coral	MediumAquamarine	RoyalBlue	Magenta
Tomato	DarkSeaGreen	Blue	MediumOrchid
OrangeRed	LightSeaGreen	MediumBlue	MediumPurple
DarkOrange	DarkCyan	DarkBlue	BlueViolet
Orange	Teal	Navy	DarkViolet

De functie: range()

Een range is een reeks opeenvolgende getallen. Als je een herhaling gaat gebruiken met bijvoorbeeld een `for loop` dan heb je ook de `range()` functie nodig. Er zijn veel mogelijkheden voor de `range()` functie. Hieronder een aantal voorbeelden:

`range(5)` geeft 0,1,2,3,4 ofwel 0 tot 5. Zoals je ziet begint Python bij 0 te tellen als je geen begin opgeeft. Met een `for()`loop en een `print()` functie kun je de getallen printen, zie het voorbeeld hieronder.

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

Je kunt ook een begin getal opgeven. Je geeft dan eerst het begin getal en daarna het getal waar Pyton onder moet blijven. `range(1,11)` geeft 1,2,3,4,5,6,7,8,9,10 ofwel 1 tot 11

Je kunt ook nog getallen overslaan door de **stapgrootte** op te geven. `range(1,11,2)` geeft 1,3,5,7,9 en telt dus met sprongen van 2

Je kunt ook de stapgrootte negatief maken waardoor je bijvoorbeeld kan beginnen met 10 en eindigen met 1.
`range(10, 0,-1)` geeft 10, 9,8,7,6,5,4,3,2,1

Opdracht 1

Maak een programma dat de getallen 2, 4, 6, 8 print. Gebruik de functies: `for()` en `range()` en `print()`

Opdracht 2

Maak een programma dat de getallen 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 print. Gebruik de functies: `for()` en `range()` en `print()`

Python ster tekenen

Voordat je aan deze les begint moet je eerst de les **vierkant tekenen** gedaan hebben. In deze les gaan we verschillende sterren tekenen, we beginnen met een ster met 5 punten.

Open Python 3 en open een nieuw blaadje met **File, New File**.

Hieronder zie je de regels die je moet intypen.

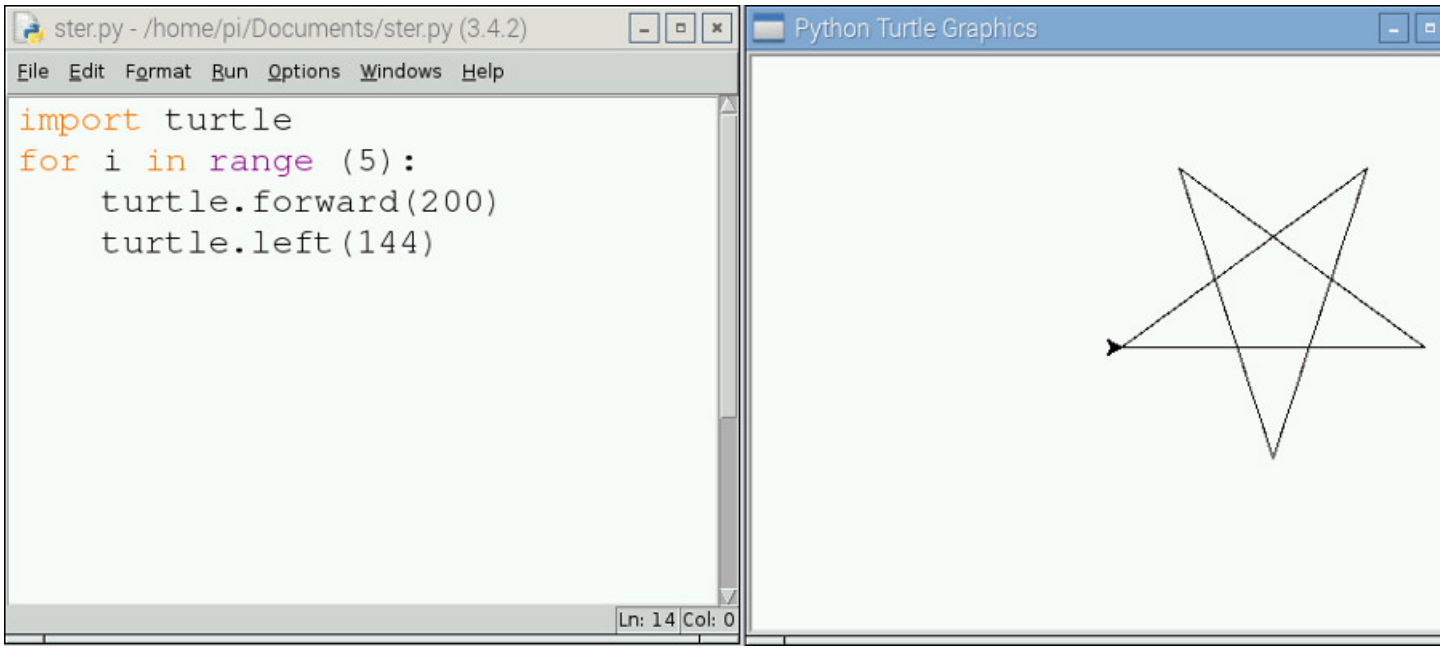
```
import turtle
for i in range (5):
        turtle.forward(200)
        turtle.left(144)
```

De eerste regel `import turtle` is nodig als eerste regel omdat je daarmee aangeeft dat je de Turtle module in je programma gebruikt, er zijn heel veel modules, bijvoorbeeld de random module

voor genereren van willekeurige getallen de tkinter module voor het maken van windows voor je programma.

De volgende regel `for i in range (5):` geeft aan dat de regels die eronder staan 5 keer herhaald moeten worden. De regel `turtle.forward(200)` laat de turtle 200 pixels naar voren gaan, de regel `turtle.left(144)` geeft aan dat de turtle 144 graden draait.

Sla het op in **Documents** en geef het een naam. Test het programma met F5.



Kun je nu zelf bedenken hoe je de lijn een kleurtje geeft? En de lijn dikker maken? En een achtergrondkleur? Je kunt ook nog even terug kijken in de les vierkant maken, onderdeel kleuren.

Ster met veel punten

Laten we nu een ster maken met veel punten. Typ onderstaad programma over en sla het op en test het met F5:

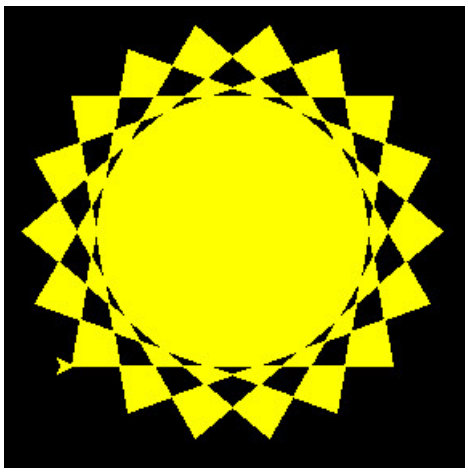
```
import turtle
for i in range (18):
    turtle.forward(200)
    turtle.left(100)
```

En probeer ook eens dit voorbeeld:

```
import turtle
for i in range (90):
    turtle.forward(200)
    turtle.left(92)
```

Kun je zelf ook een ster bedenken met heel veel punten? Probeer ook met kleurtjes te werken. In het onderstaande voorbeeld komt er de opdracht `turtle.begin_fill()`en `turtle.end_fill()`bij. Type het over en kijk wat er gebeurt:

```
import turtle
turtle.color("yellow")
turtle.bgcolor("black")
turtle.begin_fill()
for i in range (18):
    turtle.forward(200)
    turtle.left(100)
turtle.end_fill()
```



Verplaatsen op het papier

Tot nu toe begonnen we altijd te tekenen met de Turtle in het midden van het blaadje. We gaan nu de turtle verplaatsen zodat hij ergens anders op het papier kan beginnen. De turtle kan met `turtle.up()`van het papier af, en met `turtle.down()` op het papier gezet worden. met `turtle.setposition(x, y)` kun je de turtle naar een andere plek toe brengen. Je moet dan dus de x en y waarde van die plaats hebben. Probeer onderstaand voorbeeld maar eens uit. Je kunt het kopieren en plakken.

```
import turtle
turtle.color("blue")
turtle.begin_fill()
for i in range (5):
    turtle.forward(100)
    turtle.left(144)
turtle.end_fill()

turtle.up()
turtle.setposition(0, 100)
turtle.down()

turtle.color("red")
turtle.begin_fill()
for i in range (5):
    turtle.forward(100)
    turtle.left(144)
turtle.end_fill()

turtle.up()
```

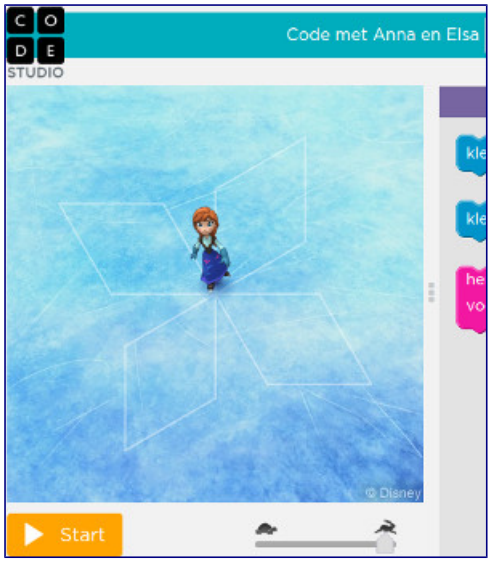
```
turtle.setposition(-150, 50)
turtle.down()

turtle.color("green")
turtle.begin_fill()
for i in range (5):
    turtle.forward(100)
    turtle.left(144)
turtle.end_fill()
```

Kun je nog meer sterren op één tekenblaadje maken? Kopieer en plak de code voor één ster en plak het onderaan je programma. Pas `turtle.setposition(x, y)` aan met andere waarde voor x en y. Test je programma met F5

Parallellogram

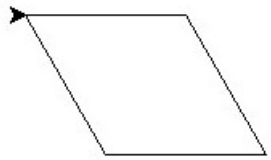
Voordat je aan deze Python les begint is het leuk om op de Code.org website eens te oefenen met programmeren in het thema Frozen, met Anna en Elsa. Code.org



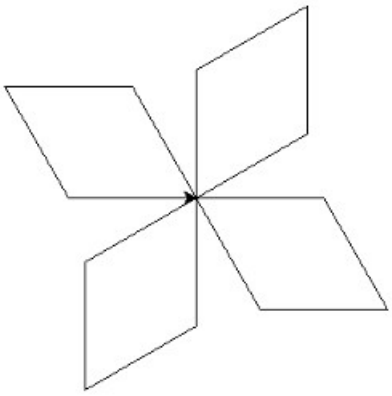
Net als in Code.org gaan we parallelogrammen programmeren, maar nu in een echte programmeertaal: Python. We gebruiken daarvoor weer een `for loop`. De naam elsa wordt hier als variabele gebruikt voor de Turtle() functie.

```
import turtle
elsa=turtle.Turtle()
for x in range (2):
    elsa.forward(100)
    elsa.right(60)
    elsa.forward(100)
    elsa.right(120)
```

Sla je programma op en noem het maar elsa. Druk weer op F5 om je programma te testen.



Is het gelukt? Zie je een parallellogram? Krijg je foutmeldingen controleer nog eens je programma. De volgende stap is deze parallellogram 4 keer herhalen met een draai van 90 graden. Je krijgt dus een for loop in een for loop. Let goed op het inspringen met tabs.

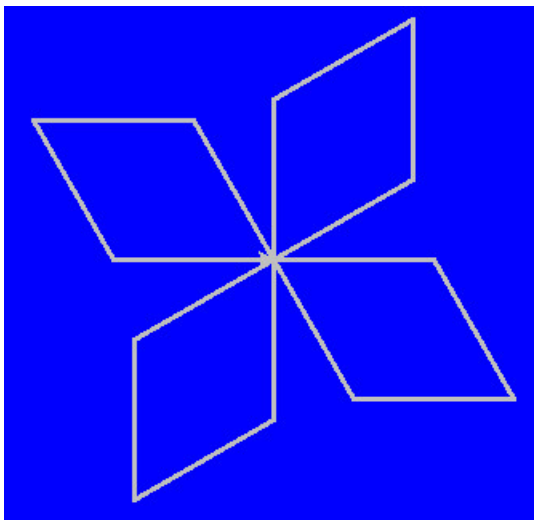


```
import turtle
elsa=turtle.Turtle()
for x in range (4):
        for x in range (2):
                elsa.forward(100)
                elsa.right(60)
                elsa.forward(100)
                elsa.right(120)
        elsa.right(90)
```

Gelukt!

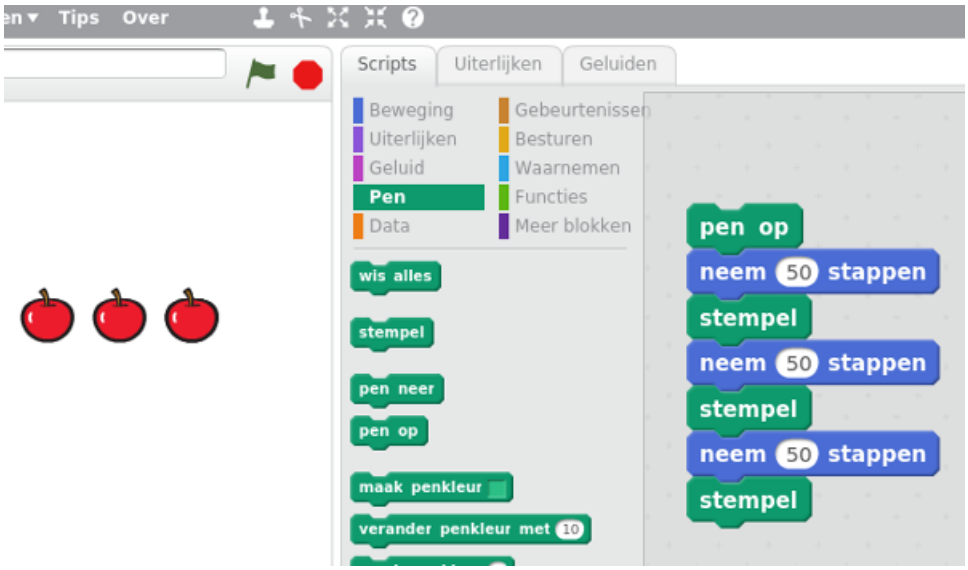
We gaan nu nog wat kleur gebruiken voor de achtergrond `turtle.Screen()` en de kleur en dikte van de lijn. Kies zelf welke kleuren je wilt.

```
import turtle
elsa=turtle.Turtle()
wn=turtle.Screen()
wn.bgcolor("blue")
elsa.color ("silver")
elsa.pensize (3)
for x in range (4):
    for x in range (2):
        elsa.forward(100)
        elsa.right(60)
        elsa.forward(100)
        elsa.right(120)
    elsa.right(90)
```



Stempelen

In deze les leer je hoe je in Python kunt stempelen. In Scratch kun je dit ook doen, kijk maar eens naar het voorbeeld hieronder:



In Python kun je stempelen met de code regel: turtle.stamp() of zoals in het voorbeeld hieronder met mijnstempel.stamp() omdat we hebben geschreven mijnstempel=turtle.Turtle(). mijnstempel is hier de variabele. Open in Python een New File en typ het voorbeeld hieronder over, sla het op en druk op F5

```
import turtle

mijnstempel = turtle.Turtle()

mijnstempel.shape("circle")

mijnstempel.penup()
mijnstempel.forward(50)
mijnstempel.stamp()
```

```
mijnstempel.forward(50)
mijnstempel.stamp()

mijnstempel.forward(50)
mijnstempel.stamp()
```

Bewaar je programma door het op te slaan. Test je programma uit met F5.

Ging alles goed? Zag je ook deze 3 circels?

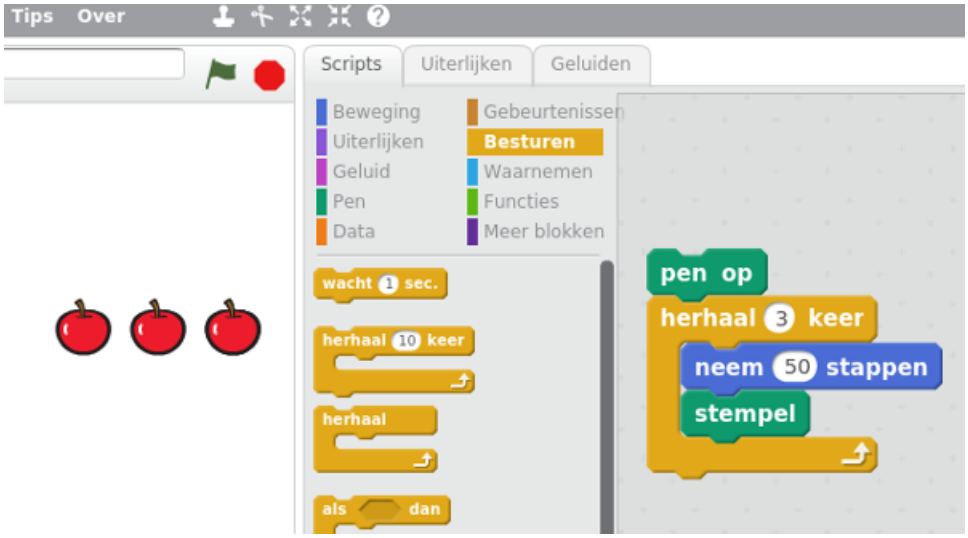
Dan heb je het goed gedaan! Lees verder.

Je kunt de vorm van de stemel ook aanpassen, je moet dan in plaats van "circle" de naam van een andere vorm intypen, je kunt kiezen uit: **"arrow", "turtle", "circle", "square", "triangle", "classic"** Probeer het maar eens!


en


Gelukt? Ga dan verder

Je kunt door een herhaling te gebruiken het programma ook wat korter maken, kijk maar eens hoe dat ging in Scratch:



In Python kun je daar deze regel voor gebruiken: `for i in range (3):`
i is hier ook een variabele, je mag zelf een letter bedenken. Bewerk je vorige stempelprogramma en maak er onderstaand voorbeeld van:

```
import turtle

mijnstempel = turtle.Turtle()
```

```
mijnstempel.shape("turtle")

mijnstempel.penup()

for i in range (3):
    mijnstempel.forward(50)
    mijnstempel.stamp()
```

Je kan jouw stempel ook een kleur geven met `mijnstempel.color("red")` . Weet jij waar je deze regel ertussen moet zetten? Probeer het maar gewoon uit.

Wil je zien welke kleurennamen er nog meer zijn? [Klik hier voor alle kleuren namen in Python.]

Hoe kun je elke stempel een andere kleur geven?

Om dit te bereiken moeten we de **for loop** `for i in range (3):` aanpassen. In plaats van `range (3)` kunnen we daar ook 3 namen van kleuren schrijven. Hieronder een voorbeeld:

```
for i in ["orange", "cyan","purple"]:
    mijnstempel.color(i)
    mijnstempel.forward(50)
    mijnstempel.stamp()
```

Je kunt ook een **lijst** maken met de kleuren namen. Die lijst geef je mee aan een variabele. In plaats van `range (3)` kunnen we daar de varabele van de lijst neerzetten.

```
kleuren =["orange", "cyan","purple"]
for i in kleuren :
    mijnstempel.color(i)
    mijnstempel.forward(50)
    mijnstempel.stamp()
```

Tot zover de les stempelen. Het maken van lijsten komt ook nog terug in de volgende lessen. (Lijsten)

Loops of lussen

In deze les leer je hoe je in Python een While loop maakt. Een While loop is net als een For loop bedoeld voor het herhalen van stukjes code. Een For loop herhaalt het aantal keer dat je opgeeft, bijvoorbeeld `for i in range (4)`, hier wordt 4 keer herlaald. Bij een While loop stopt de herhaling pas als aan een bepaalde conditie wordt voldaan. While betekent ook terwijl, bijvoorbeeld `while herhaling < 4` betekent: terwijl herhaling is kleiner dan 4. Als de conditie True (waar) blijft zal de loop doorgaan. Je ziet in het programma hieronder dat **herhaling** steeds ééntje meer wordt, net zo lang tot herhaling 4 is, de conditie wordt dan False (niet waar) en dan stopt het programma voordat het weer de loop ingaat.

```
import turtle

herhaling = 0
while herhaling < 4:
    turtle.forward(100)
    turtle.left(90)
    herhaling = herhaling + 1
```

Probeer het programma hierboven maar eens uit. Bewaar je programma door het op te slaan. Test je programma uit met F5.

De regel `herhaling = herhaling + 1` kun je ook anders en korter opschrijven: `herhaling += 1`.

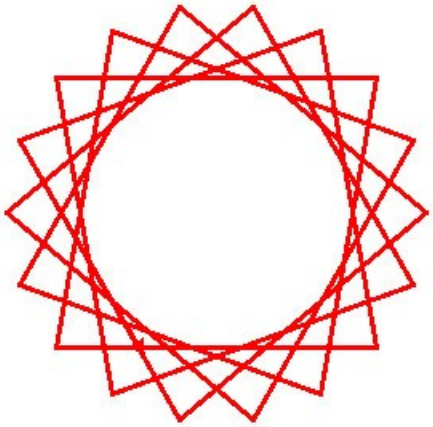
Een while loop kan ook oneindig doorgaan. Dit gebeurt bijvoorbeeld als de conditie nooit bereikt wordt. Haal de laatste regel van je programma maar eens weg, save nog een keer en run met F5.

De conditie `herhaling < 4` zal altijd True blijven, en de loop blijft doorgaan.

while True

Een oneindig doorgaande lus kun je ook maken met `while True:`. Let wel op de hoofdletter T van True.

```
import turtle
turtle.pensize (3)
turtle.color ("red")
while True:
    turtle.forward(200)
    turtle.left(100)
```



if , elif en else statements

In deze les leer je hoe je in Python gebruik kunt maken if, elif en else statements.
`if` betekent: als
`elif` betekent: anders als
`else` betekent: anders

`if` betekent: als. We gebruiken `if` om een bepaalde conditie te vergelijken. Bijvoorbeeld `if leeftijd < 18:`, hier staat dus: als leeftijd is kleiner dan 18: De uitkomst van de vergelijking kan zijn : True of False (waar of nietwaar). Als de conditie True is wordt de code onder het `if` statement uitgevoerd.

Voorbeeld:

```
leeftijd=float(input('hoe oud ben je? typ het cijfer hier:'))
if leeftijd < 18:
                print("je bent nog niet volwassen")
```

Na een `if` statement komt vaak een `else` statement. Een `else` statement bevat een stukje code dat wordt uitgevoerd als de conditie van het if statement 0 of False is.

Voorbeeld:

```
leeftijd=float(input('hoe oud ben je? typ het cijfer hier:'))
if leeftijd < 18:
        print("Je bent nog niet volwassen!")
else:
        print("Je bent volwassen!")
```

Tussen if en else kan ook nog een `elif` statement staan. Elif is een samentrekking van de woorden `else if` en betekent: anders als. Net als een `if` statement moet je achter het `elif` statement ook een conditie schrijven. Bijvoorbeeld `elif leeftijd >= 80:`.

```
leeftijd=float(input('hoe oud ben je? typ het cijfer hier:'))

if leeftijd < 18:
    print("je bent nog niet volwassen")

elif leeftijd >= 80:
    print("Gefeliciteerd! Je bent best oud!")

else:
print("Je bent volwassen!")
```

Je kunt ook meerdere `elif` statements gebruiken. Kijk maar eens naar het voorbeeld hieronder en bedenk daarna zelf een programma met `if`, `elif` en `else` statements

.

```
import turtle
turtle.pensize(3)
kleur=input('kies uit rood, groen, blauw of zwart:')

if kleur=="rood":
     turtle.color("red")
     turtle.circle(30)
elif kleur=="groen":
     turtle.color("green")
     turtle.circle(30)
elif kleur=="blauw":
     turtle.color("blue")
     turtle.circle(30)
elif kleur=="zwart":
     turtle.color("black")
     turtle.circle(30)
else:
     print("Je kon alleen kiezen uit rood, groen,blauw of zwart!")
```

Operators

Operators zijn de symbolen die die in Python de wiskundige bewerkingen uitvoeren. Voorbeelden zijn:

Wiskundige operators:

```
+       plus
-       min
*       keer
/       delen
%       modulus (rest van een deelsom)
//      geeft een geheel getal na deling afgerond naar beneden
```

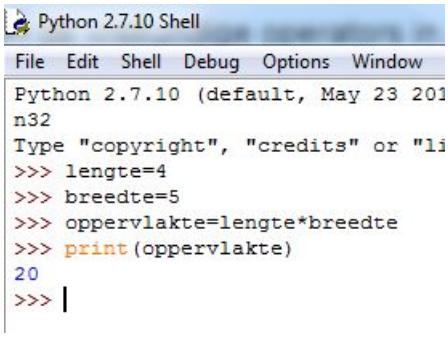
```
**      exponent
```

Voorbeelden in Python

Laten we nu een kort programma schrijven waarin één van de wiskundige operators zit.

```
lengte=4
breedte=5
oppervlakte=lengte*breedte
print(oppervlakte)
```

Open Python en typ de regels in de Python Shell zoals het voorbeeld hieronder:



Opdracht

Hierboven zag je een programma dat de oppervlakte van een rechthoek berekend. Kun je nu een programma schrijven waarmee de omtrek van de rechthoek berekend wordt?

Vergelijkende operators:

Vergelijkende operators worden gebruikt om waardes met elkaar te vergelijken. Ze geven True of False terug, dus waar of niet waar, afhankelijk van de vergelijking. Deze waarde noem je een **boolean** waarde. Voorbeelden van deze operators zijn:

```
<       kleiner dan
<=      kleiner dan of gelijk aan
>       groter dan
>=      groter dan of gelijk aan
==      gelijk aan
!=      niet gekijk aan
```

Voorbeelden in Python

Test deze voorbeelden maar eens uit in Python:

```
a = 10 < 20
print(a)
```

```
>>> a=10<20
>>> print(a)
True
>>>
```

```
b = 5 == 5
print(b)

c = 3 <= 3
print(c)
```

Bedenk nu vooraf wat de boolean waarde zal zijn van onderstaande vergelijkingen:

d = 10 < 9

e = 5 != 5

f = 3 >= 4

En wat is nou het verschil tussen == en =? Gebruik == wanneer je wil vragen of iets gelijk is, gebruik = wanneer je iets wil toekennen. Hieronder een voorbeeld:

```
a = 4                  // toekenning,  we maken a gelijk aan 4
b = 5                  // toekenning,  maken we b gelijk aan 5
if a == b:             // is a gelijk aan b?
    print ("a is gelijk aan b")


if a != b:          // is a niet gelijk aan b?
    print ("a en b zijn niet gelijk")
```

Logische operators:

Er zijn drie logische operators:

`and` betekent **en**

`or` betekent **of**

`not` betekent **niet**

Als deze operators in een vergelijking worden gebruikt kunnen ze `True` of `False` terug geven.

`and` geeft `True`,als zowel links als rechts van de vergelijking `True` is, anders `False`. Een voorbeeld kan zijn wanneer je moet inloggen op een website met een gebruikersnaam en wachtwoord. Zowel de gebruikersnaam en het wachtwoord moet je goed invullen anders ben je niet ingelogd. Als de vergelijking `True` geeft ben je ingelogd, als de vergelijking `False` geeft ben je niet ingelogd. Hieronder een voorbeeld:

```
gebruikersnaam = "tamara"
wachtwoord = "appelmoes"

if gebruikersnaam == "tamara" and wachtwoord == "appelmoes":
    print("je bent ingelogd")

else:
    print ("je bent niet ingelogd")
```

In het voorbeeld hierboven geeft de vergelijking natuurlijk altijd `True` omdat de variabele gebruikersnaam en wachtwoord al gegeven zijn. Het is leuk om met de functie `input()` zelf je gebruikersnaam en wachtwoord in te typen. Probeer onderstaand voorbeeld maar eens uit:

```
gebruikersnaam = input("geef je gebruikersnaam: ")
wachtwoord = input("geef je wachtwoord: ")

if gebruikersnaam == "tamara" and wachtwoord == "appelmoes":
    print("je bent ingelogd")

else:
    print ("je bent niet ingelogd")
```

`or` geeft `True` als links **of** rechts van de vergelijking `True` is, of als **beide** zijde van de vergelijking `True` zijn. Hieronder een voorbeeld:

```
naam = "Tom"

if naam == "Tom" or naam == "Tim":
    print ("Je naam is Tom of je naam is Tim.")
```

`not` maakt van een True waarde False en andersom, van een False waarde True. Hieronder een eenvoudig voorbeeld van `not`:

```
regen = False
print (not regen)
```

Er zijn nog meer soorten Operators, maar voor nu zijn bovenstaande voldoende om mee te oefenen.

Lists of Lijsten

Een list (Engels voor lijst) is één van de ingebouwd data-type in Python. Zoals de naam het al zegt wordt het gebruikt om lijsten op te slaan. Zoals lijsten van gebruikersnamen, personen, leeftijden, enz. Het voordeel van een lijst is dat de data heel ordelijk kan worden opgeslagen. Zo kan de lijst bijvoorbeeld gesorteerd worden op alfabet. Een lijst wordt aangemaakt door middel van deze `[ ]` vierkante haakjes en de individuele waarden zijn gescheiden door komma's. Bijvoorbeeld:

```
boodschappen = ["kaas", "thee", "boter", "eieren"]
```

Als de variabele in de lijst woorden zijn moeten ze tussen aanhalingstekens`" "` of `' '`, als het getallen zijn hoeft dat niet, behalve als het decimalen zijn.

Bij een lijst kun je de inhoud aanpassen. Iets toevoegen kan met de functie `append()`. Met de `append()` functie kun je maar één item toevoegen aan de lijst. Iets weghalen kan met de functie `remove()`. Je moet de functie dan aan de naam van je lijst plakken met een puntje ertussen.

Bijvoorbeeld iets **toevoegen** aan de boodschappen lijst: `boodschappen.append("chips")`.
Bijvoorbeeld iets **weghalen** uit de boodschappen lijst: `boodschappen.remove("thee")`.

Wil je **meer items tegelijk toevoegen** aan je lijst dan kun je de functie `extend()` gebruiken
Bijvoorbeeld: `boodschappen.extend(["cola", "hagelslag"])`

Wil je een item uit de lijst printen kun je dat doen met de functie: `print()`. Python houdt de items in de lijst bij met een telling of volgnummer, het eerste item in de lijst is 0, het tweede it item de lijst is 1, enzovoort. Wil je bijvoorbeeld het **vijfde item** uit de lijst printen doe je dat zo:
`print(boodschappen[4])`

Opdracht

Open Python en maak een lijst met 4 kleuren. Voeg daarna nog 1 kleur toe met de functie `append()`, en voeg daarna nog 2 kleuren toe met de functie `extend()`. Print de laatste kleur uit de lijst.
Weet je niet hoe het moet? Kijk dan naar het

Er zijn nog andere functies die je kunt gebruiken:.
`len()` geeft de lengte van de lijst, dus hoeveel items erin staan.
`count()` hiermee kun je python vragen hoevaak eenzelfde item in de lijst voorkomt.
`index()` met deze functie kun je het volgnummer van een item opzoeken.
`sort()` hiermee kan Python je lijst sorteren. Bij strings op alfabetische volgorde of bij getallen oplopend.
`sorted()` hiermee kan je een gesorteerde volgorde laten zien, bijvoorbeeld met print functie, de echte volgorde zal niet veranderen.

For loop

Tot slot nog een voorbeeld hoe je alles in de lijst onderelkaar kan printen met een `for()` loop.

```
week = ["maandag", "dinsdag", "woensdag", "donderdag", "vrijdag", "zaterdag",
"zondag"]

lengte = len(week)


for i in range(0,lengte):
    print (week[i])
```