



# Python

Auteurs	Maaike van Kessel ; J.D.T. Thomson ; Maaike van Kessel
Team	Wikiwijs Maken Auteurs
Laatst gewijzigd	25 maart 2021
Licentie	CC Naamsvermelding 4.0 Internationale licentie
Webadres	<a href="https://maken.wikiwijs.nl/162263/">https://maken.wikiwijs.nl/162263/</a>



Dit lesmateriaal is gemaakt met Wikiwijs van Kennisnet. Wikiwijs is hét onderwijsplatform waar je leermiddelen zoekt, maakt en deelt.

# Inhoudsopgave

Links en tips .....	2
Installeren .....	3
Omschrijving .....	3
Praktijk .....	4
Huiswerk .....	19
Turtle .....	21
Omschrijving .....	21
Praktijk .....	23
Huiswerk .....	35
Loops .....	36
Omschrijving .....	36
Praktijk .....	43
Huiswerk .....	47
If/Else .....	49
Omschrijving .....	49
Praktijk .....	51
Calculator .....	53
Omschrijving .....	53
Praktijk .....	58
Over dit lesmateriaal .....	63

# Links en tips

## Web

Als je zelf wil beginnen met programmeren kun je gebruik maken van de volgende links:

- Online cursus w3schools: <https://www.w3schools.com/python/default.asp>
- Turtle: <https://realpython.com/beginners-guide-python-turtle/>
- Turtle library (hier staan alle bestaande funcites in) <https://docs.python.org/3/library/turtle.html>

## Boeken

Locatie waar je boeken vandaan haalt: <https://pythonbooks.revolunet.com/>

- Automate the Boring Stuff with Python: <https://automatetheboringstuff.com/>

# Installeren

## Omschrijving

Als ICT'er ga je mooie en coole dingen ontwikkelen. Of het nou om applicaties, websites of servers gaat, je gaat programmeren tegen komen. Omdat je als medewerker in IT programmeren in moet kunnen zetten voor bijvoorbeeld webdesign en linux en het aanpassen van applicaties gaan we met Python programmeren.

Hiervoor moet Python eerst geïnstalleerd worden. Om Python te gebruiken heb je ook een programmeeromgeving nodig. Er zijn heel veel programmeeromgevingen, in dit geval gaan we Visual Studio Code gebruiken.

## Leerdoelen:

1. Je leert de software te installeren: Python en een programmeeromgeving.
2. Je leert hoe je je eerste code schrijft.

## Theorie

Een **interpreter** is een [computerprogramma](#) dat als tolk fungereert. De interpreter vertaalt de [broncode](#) van computerprogramma's naar een voor de [processor](#) begrijpelijke taal, en voert de vertaling ook meteen uit.

Sommige programmeertalen gebruiken een [compiler](#). De compiler slaat de vertaalde broncode op, zodat deze later uitgevoerd kunnen worden.

De interpreter helpt ons om in mensentaal (Python) te communiceren met de computer in machinecode (01).

Als je Python installeert dan installeer je de interpreter en standaard [library](#) van Python.

In een library kun je al kant-en-klare functionaliteit vinden die jou helpt met het maken van goede code. Je hoeft door deze library niet meer zelf alle functionaliteit uit te vinden. Als voorbeeld biedt Python (de interpreter) een hele bazale netwerkverbinding. De library bouwt hier bovenop de mogelijkheid om met deze bazale netwerkverbinding een [http](#), [ssh](#) of [smtp](#) verbinding te maken.

Als je code aan het schrijven bent, kan je natuurlijk fouten maken. Hier is een handige tool voor genaamd [debugger](#).

Een **debugger** is een computerprogramma dat gebruikt wordt om andere computerprogramma's te controleren. De debugger zoekt de oorzaak van de fout (dit wordt in programmeren een [bug](#) genoemd).

Als het programma dat je aan het debuggen bent op een onverwachte manier wordt afgebroken vanwege een fout, laat de debugger, de positie in de broncode zien waar de fout optreedt.

! Let hierbij wel op dat de fout ook in een andere regel kan zitten die te maken heeft met het onderwerp dat wordt aangetoond.

## Praktijk

### ***Installeren van Python en Visual Studio Code***

Om python te gebruiken heb je zowel python als een programmeer omgeving nodig.

#### ***Installeren Python***

Het kan zijn dat je Python al hebt geïnstalleerd. Het is verstandig om dit eerst te controleren.

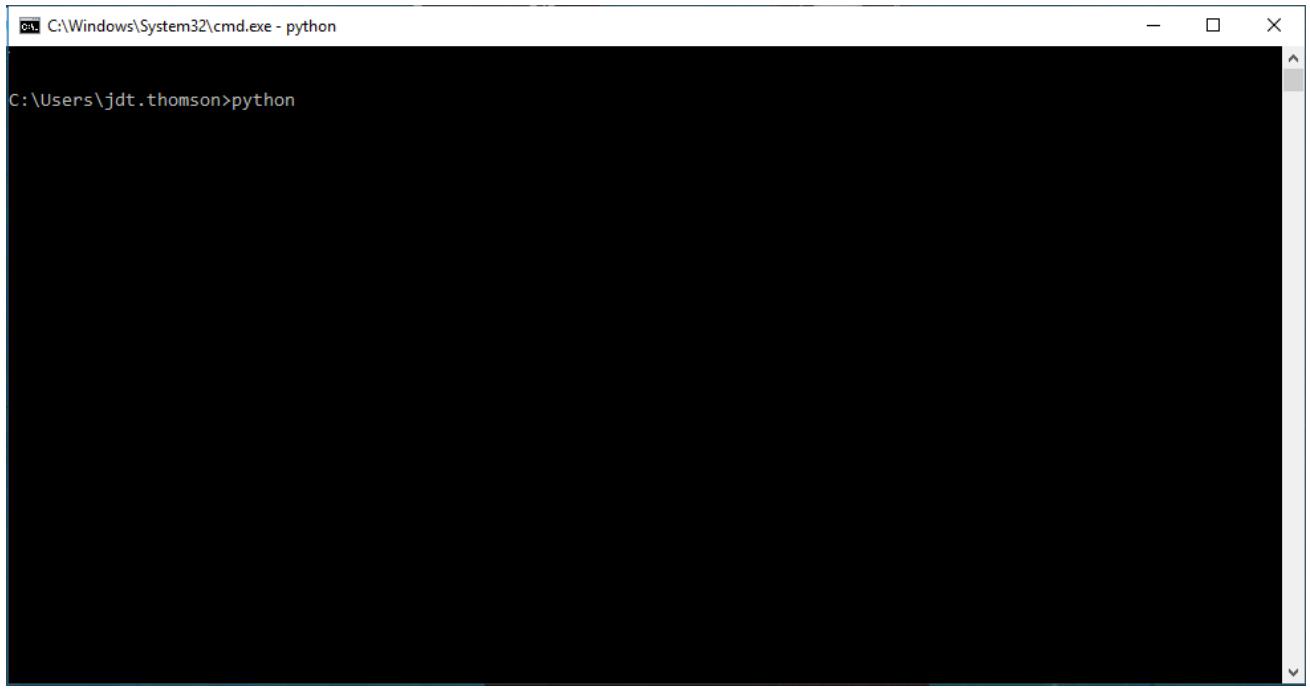
##### *Controle*

Stap 1: Ga naar je command prompt "CMD.exe"



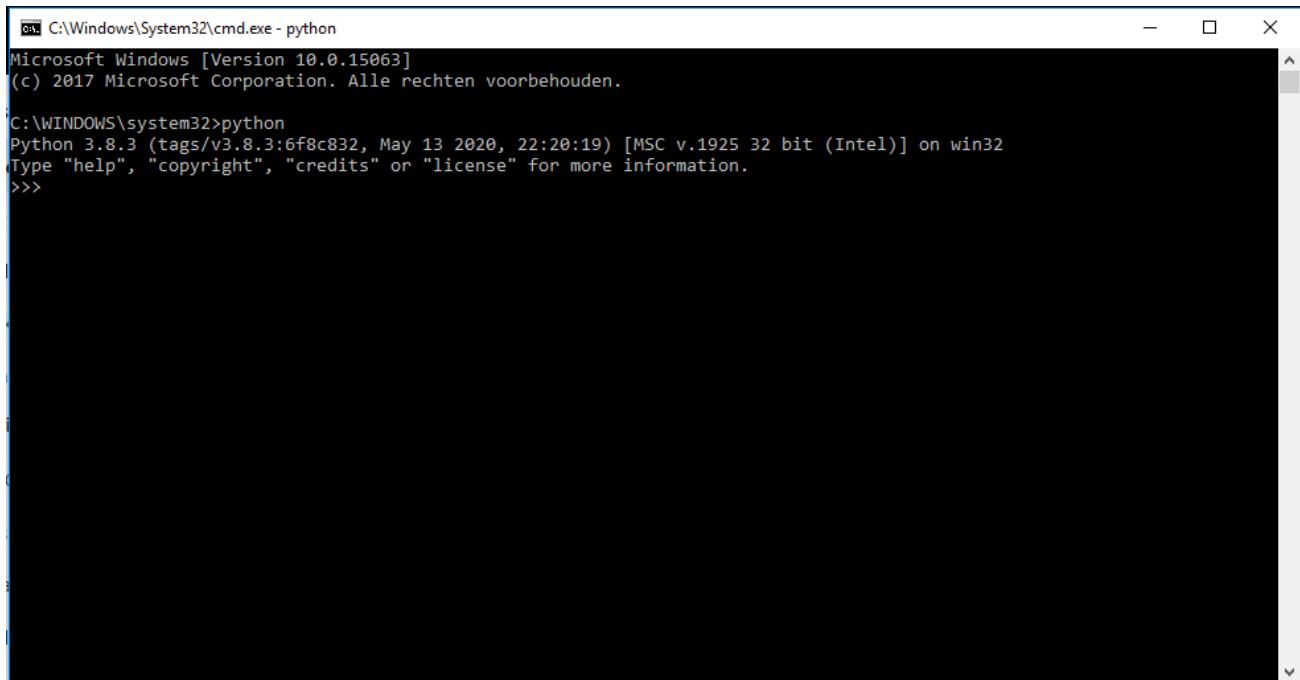
```
C:\Windows\System32\cmd.exe - python
C:\Users\jdt.thomson>
```

Stap 2: Typ Python in en controleer of het geïnstalleerd is.



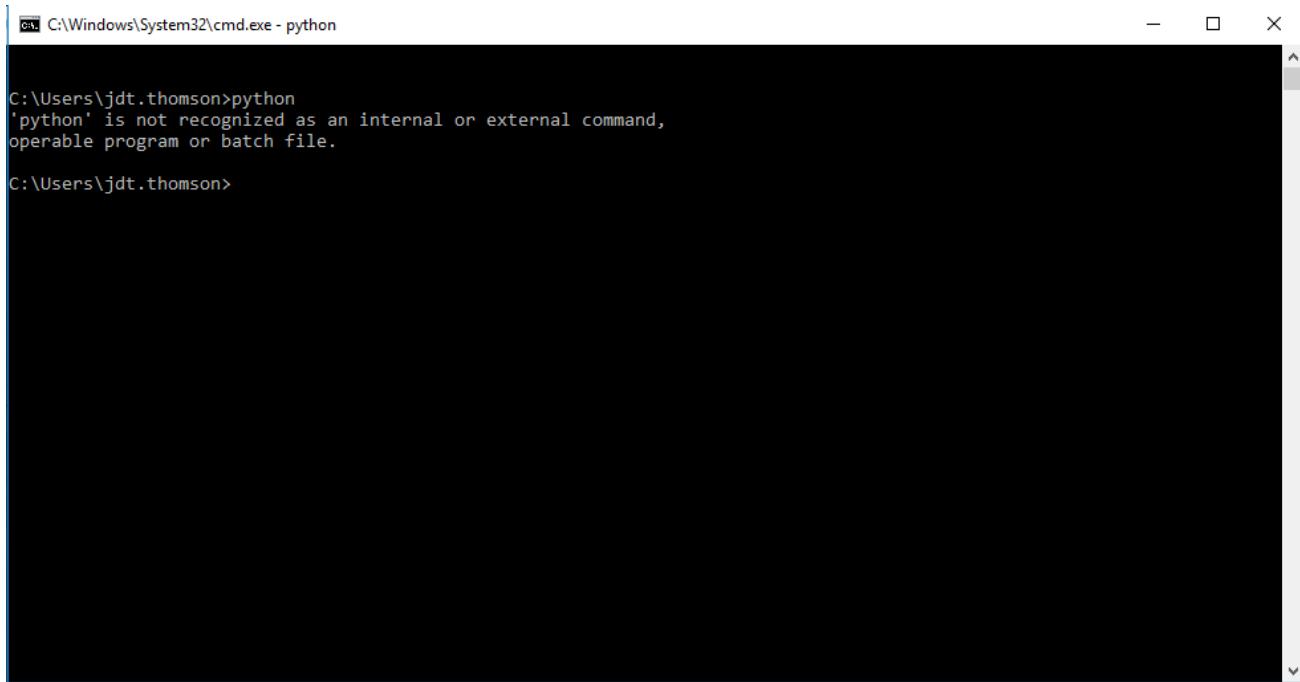
A screenshot of a Windows Command Prompt window titled "C:\Windows\System32\cmd.exe - python". The window is mostly empty, with only the command prompt line "C:\Users\jdt.thomson>python" visible at the top. The rest of the screen is black.

Als python geïnstalleerd is ziet het er zo uit:



C:\Windows\System32\cmd.exe - python  
Microsoft Windows [Version 10.0.15063]  
(c) 2017 Microsoft Corporation. Alle rechten voorbehouden.  
C:\WINDOWS\system32>python  
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>>

Als python niet geïnstalleerd is ziet het er als volgt uit:



A screenshot of a Windows Command Prompt window titled "C:\Windows\System32\cmd.exe - python". The window shows the following text:  
C:\Users\jdt.thomson>python  
'python' is not recognized as an internal or external command,  
operable program or batch file.  
C:\Users\jdt.thomson>

Als python al is geïnstalleerd kun je verder gaan met het installeren van Visual Studio Code.

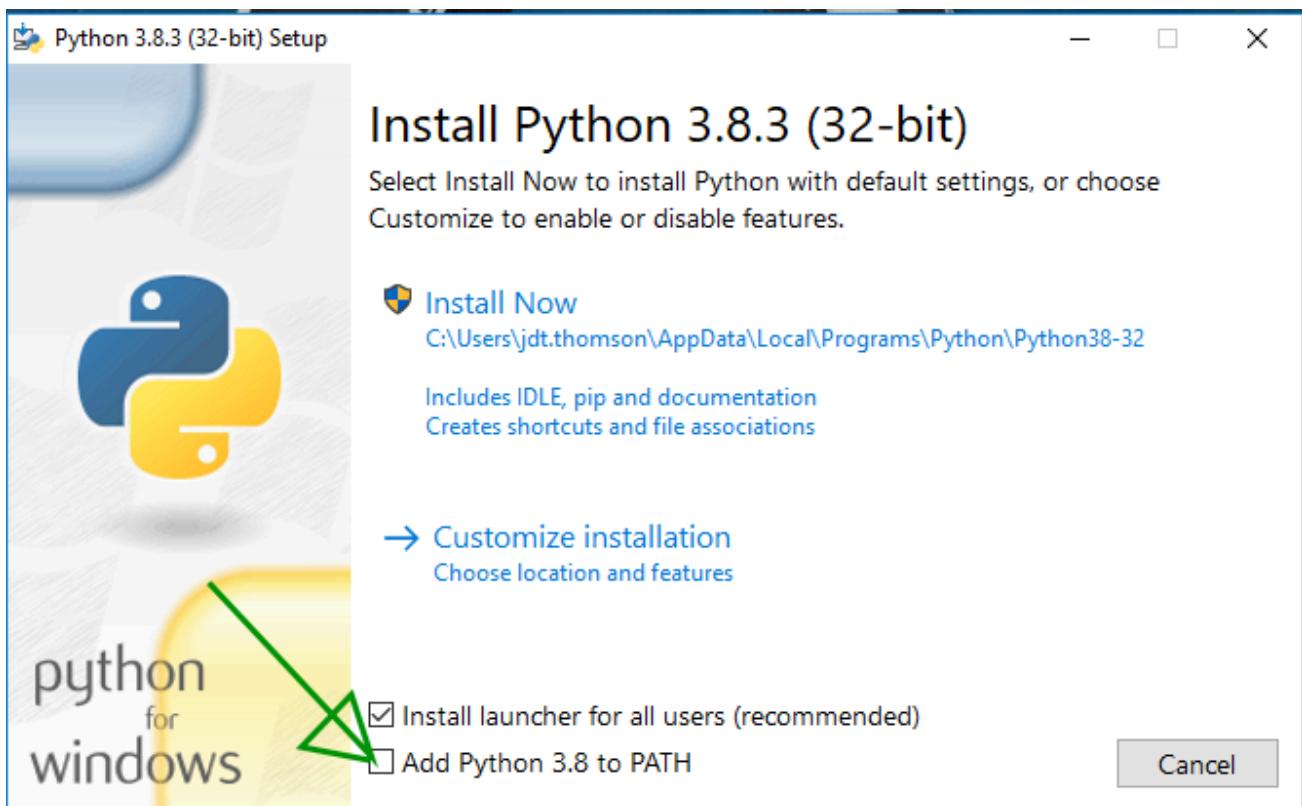
## ***Installeren***

Stap 1: ga naar <https://www.python.org/downloads/> en download Python 3.8.3 (of nieuwere).

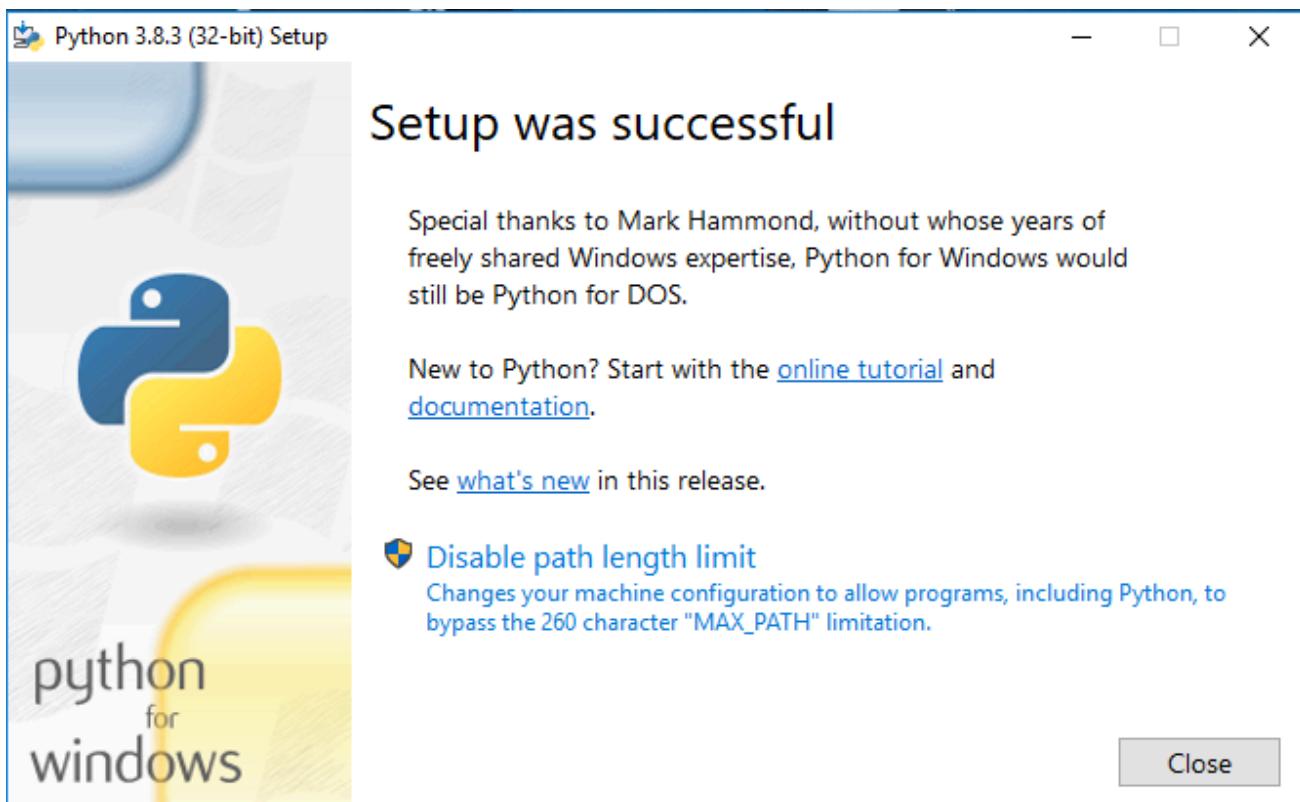
The screenshot shows the Python.org website's download section. At the top, there's a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, Community, and a search bar. Below the navigation is the Python logo. To the right of the logo are buttons for 'Donate', 'Search', 'GO', and 'Socialize'. A main heading says 'Download the latest version for Windows' with a 'Download Python 3.8.3' button. Below this, there are links for other OSes and Docker images. To the right is an illustration of two boxes with yellow and white parachutes falling from the sky. At the bottom left, there's a section for 'Active Python Releases'.

python-3.8.3.exe

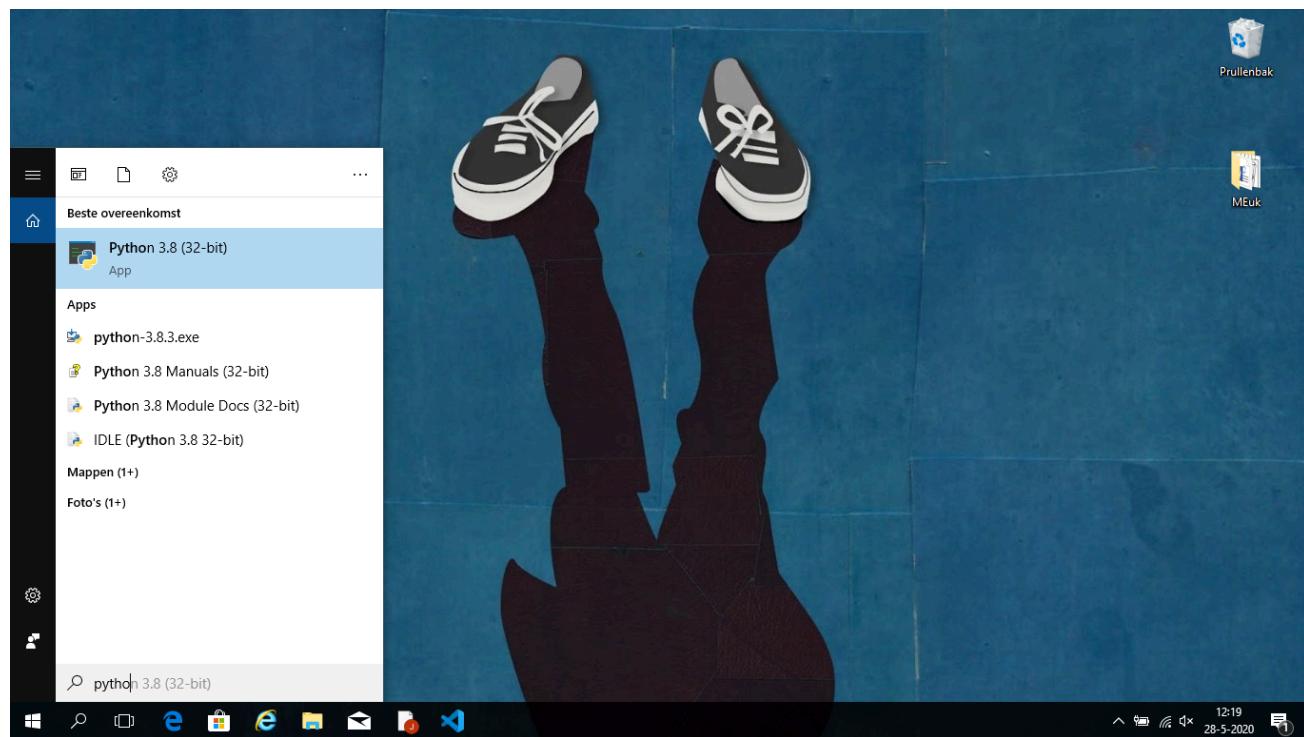
Stap 2: Installeer Python en let er op dat je Add Python 3.8 to PATH aanvinkt. Dit zorgt ervoor dat python in de hele pc terug te vinden is en niet alleen in de map waar je hem hebt aangemaakt.



Python is geïnstalleerd!



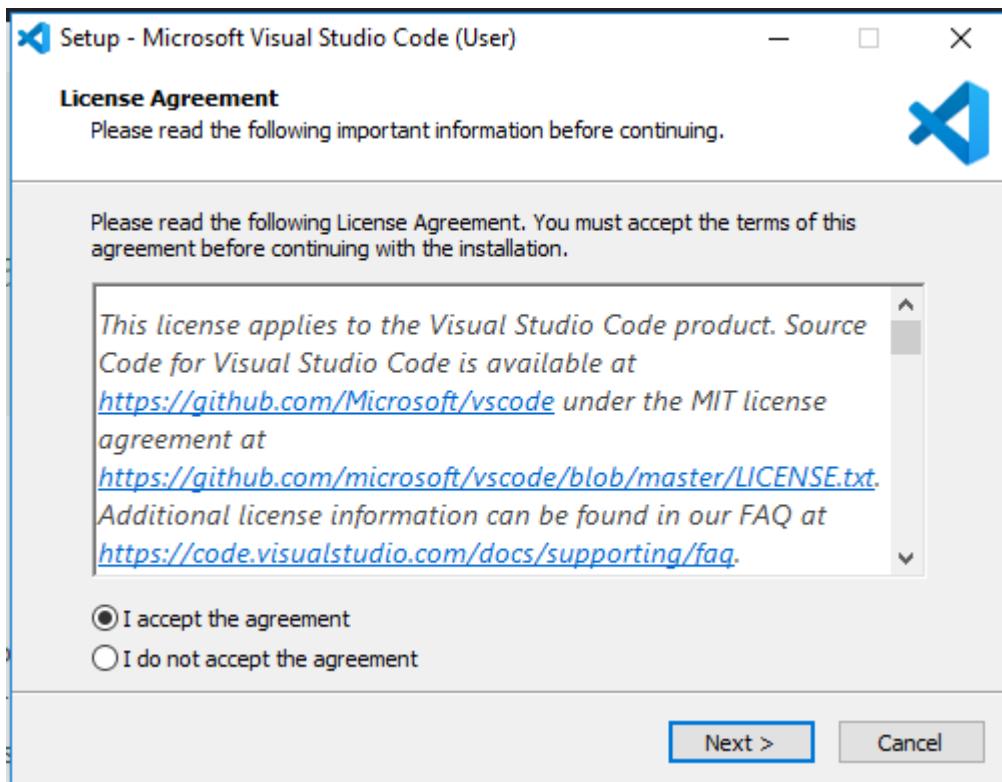
Stap 3: Controleer of je de Python app kunt vinden in je applicatie lijst.



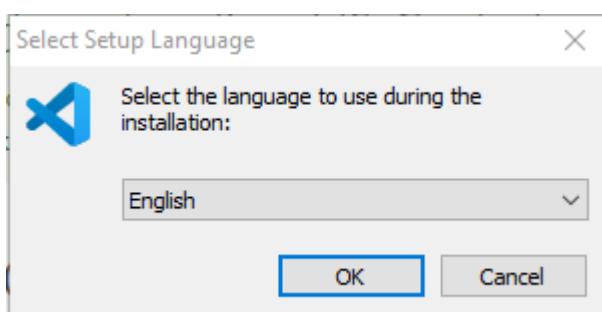
# Installeren van Visual Studio Code

Stap 1: Ga naar <https://code.visualstudio.com/docs> en download Visual Studio Code.

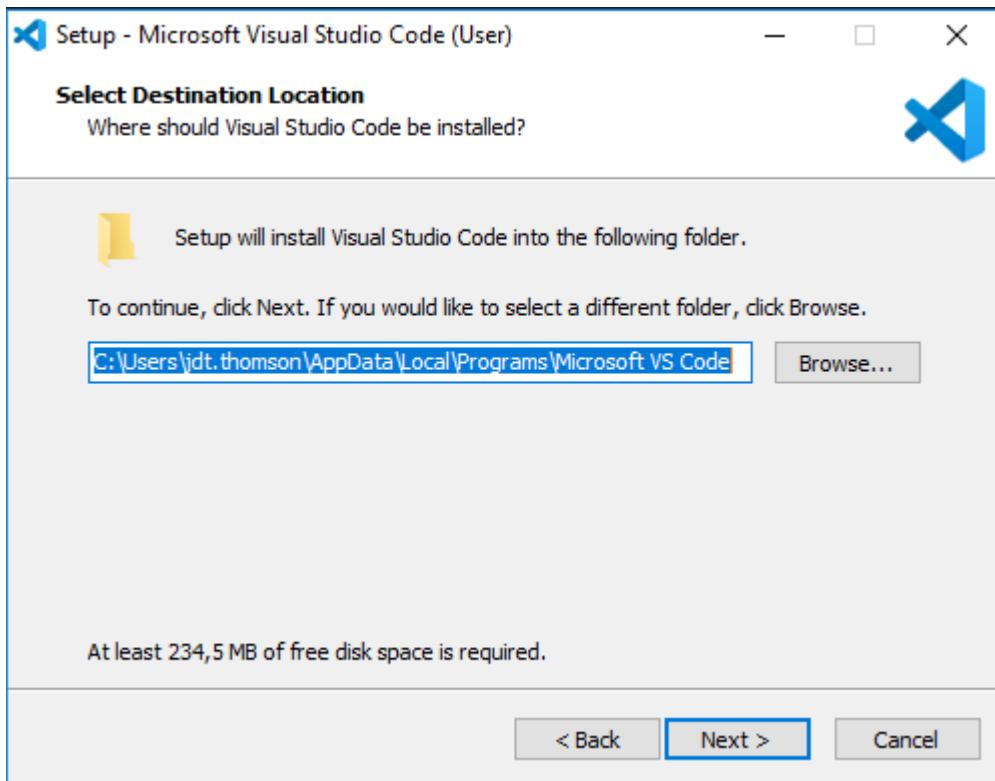
Stap 2: Accepteer de licentieovereenkomst en klik next



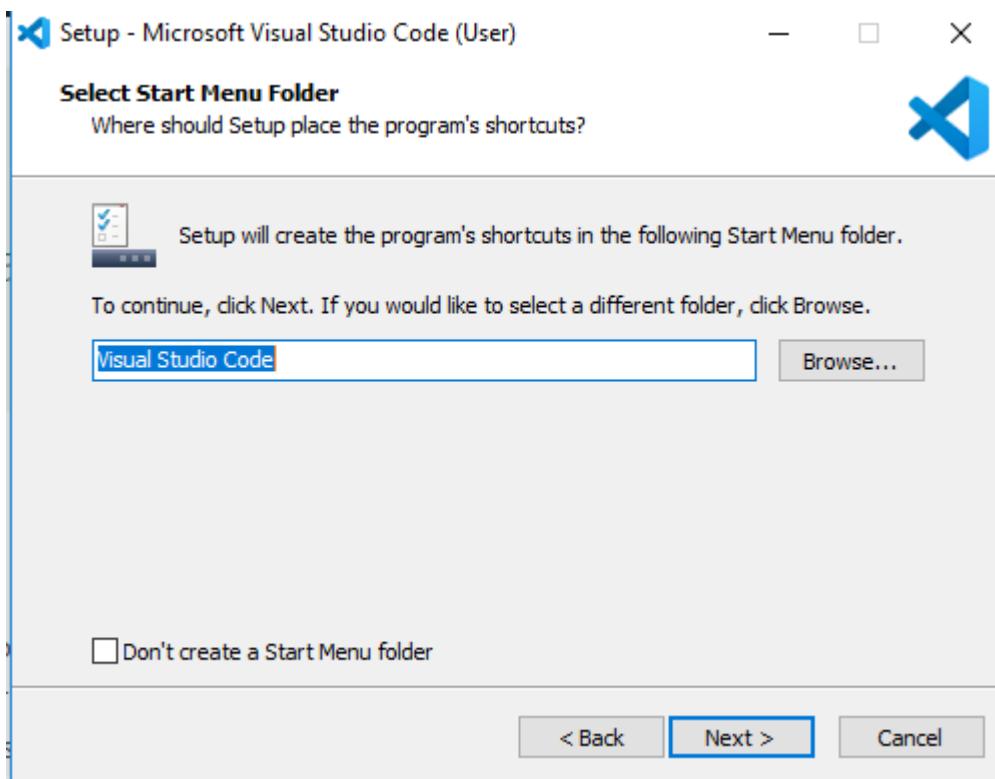
Stap 3: Selecteer de taal. Het is verstandig om Engels als de taal te kiezen, als je iets op moet zoeken op het internet krijg je veel meer informatie als je de Engelse termen gebruikt dan met de Nederlandse termen. Vind je Engels te moeilijk kun je natuurlijk Nederlands kiezen.



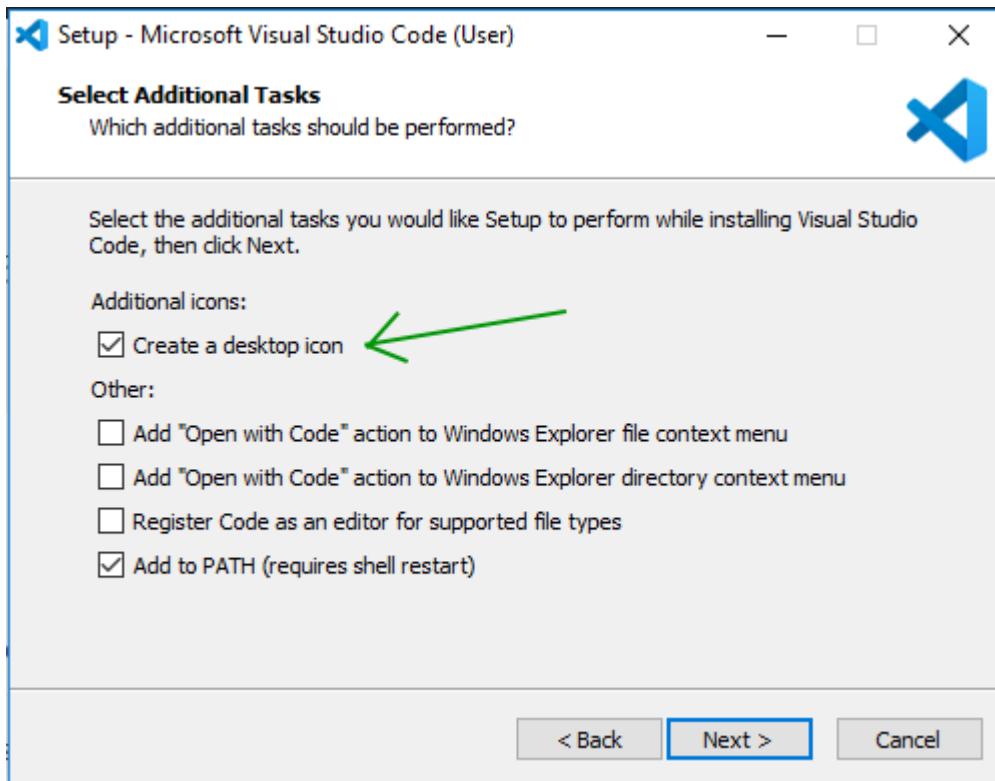
Stap 4: Kies een locatie om de installatie op te slaan.



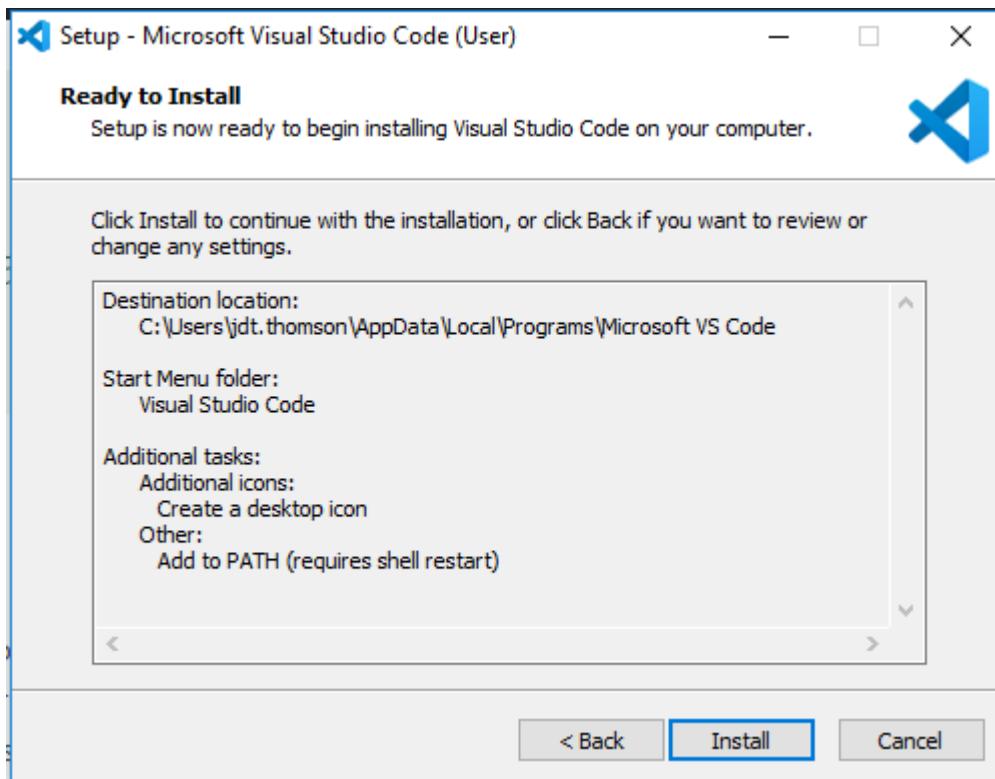
Stap 5: Er wordt een shortcut aangemaakt in het start menu met de naam "Visual Studio Code", zodat je hem makkelijk terug vindt.



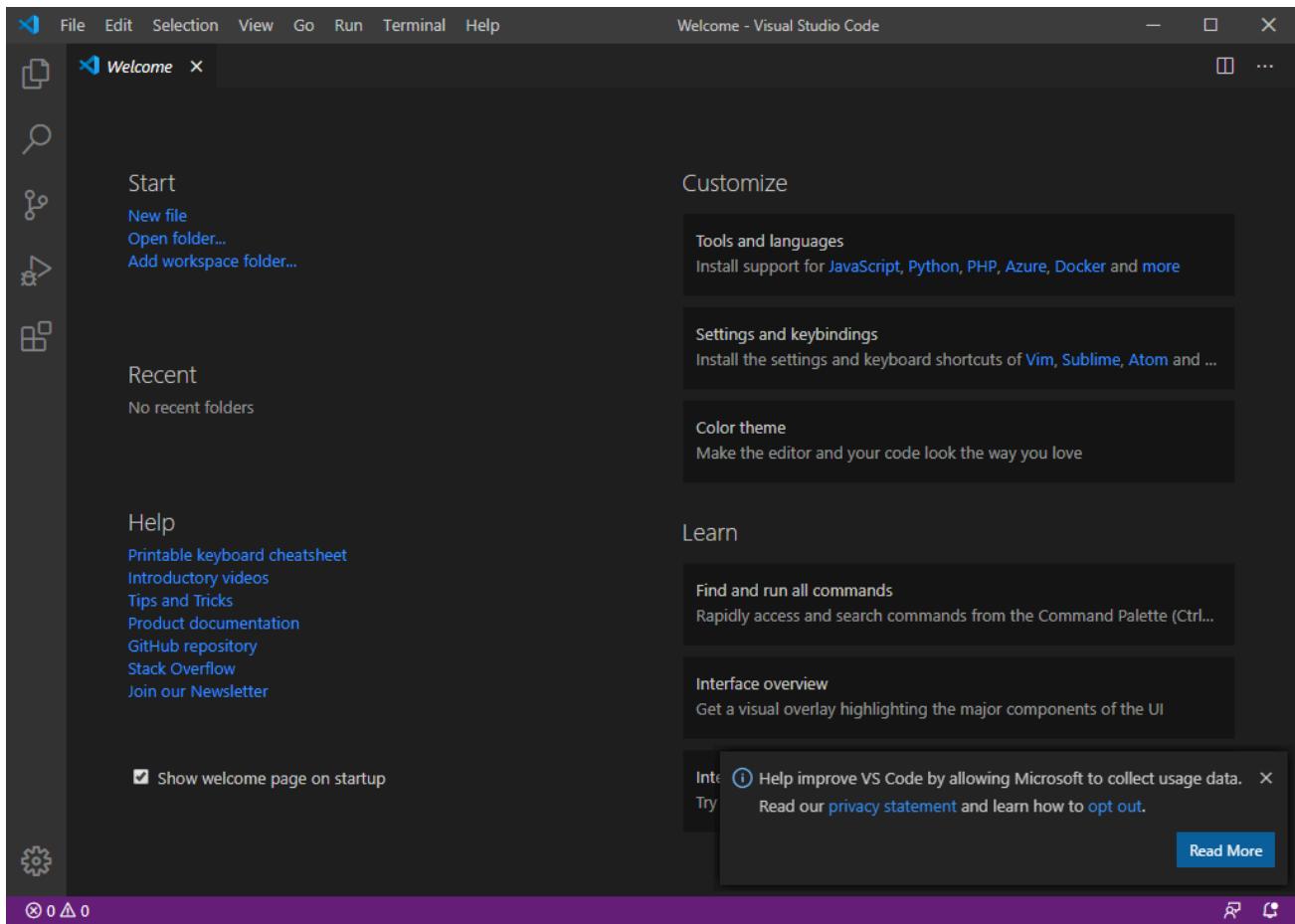
Stap 6: Vink Create a desktop icon aan als je Visual Studio Code makkelijk op je desktop wil terug vinden.



Stap 7: Visual Studio Code is nu klaar voor de daadwerkelijke installatie.

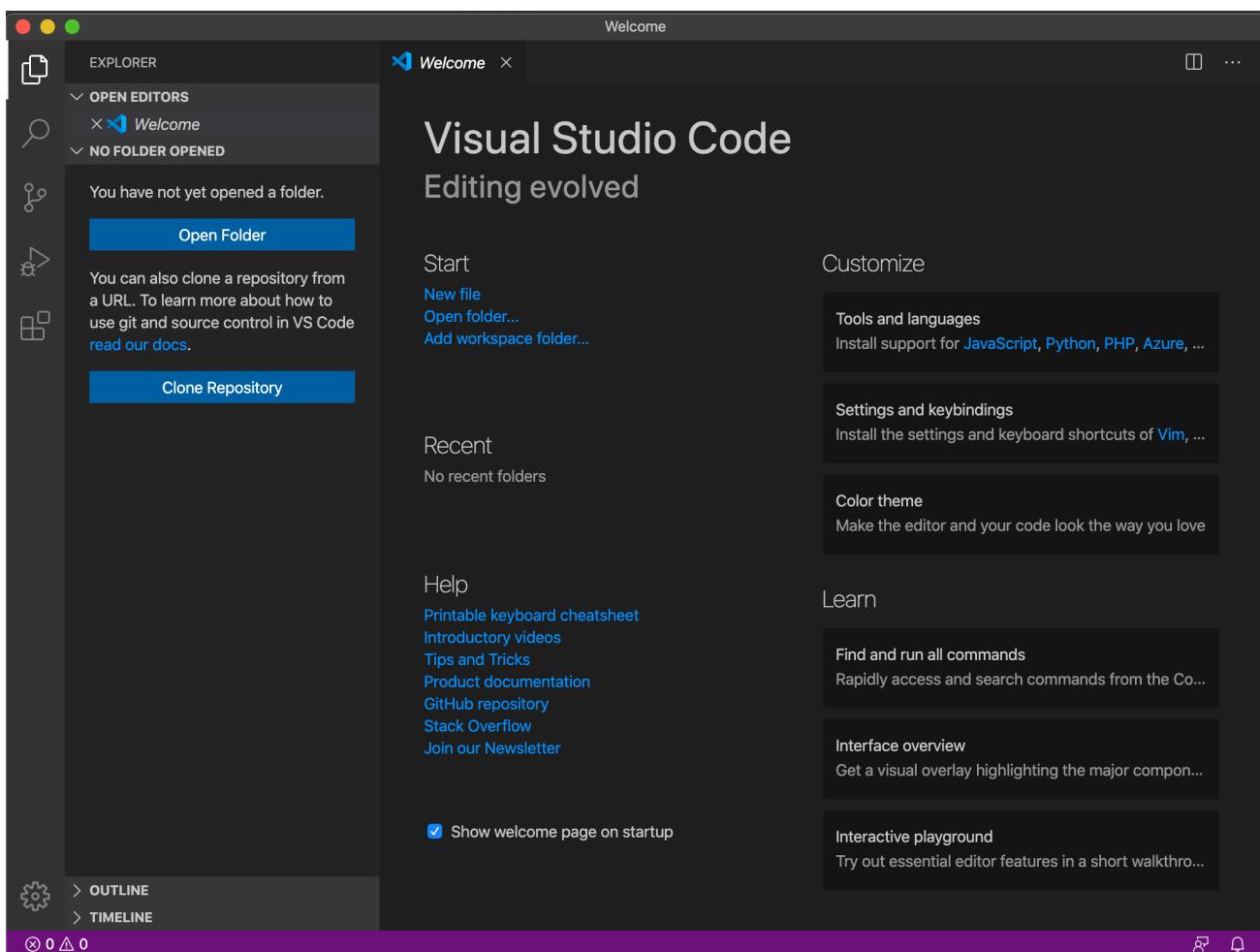


Gefeliciteerd je hebt Visual Studio Code geïnstalleerd op je PC! Als het goed is kun je het programma nu openen en zie je het volgende:

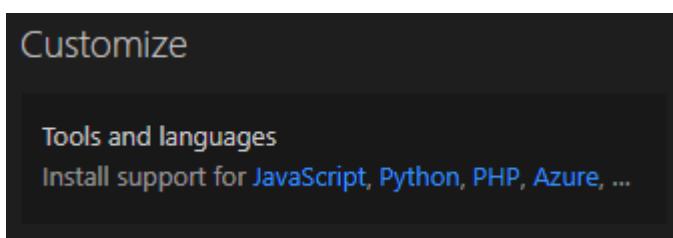


## Python en Visual Studio Code intergreren

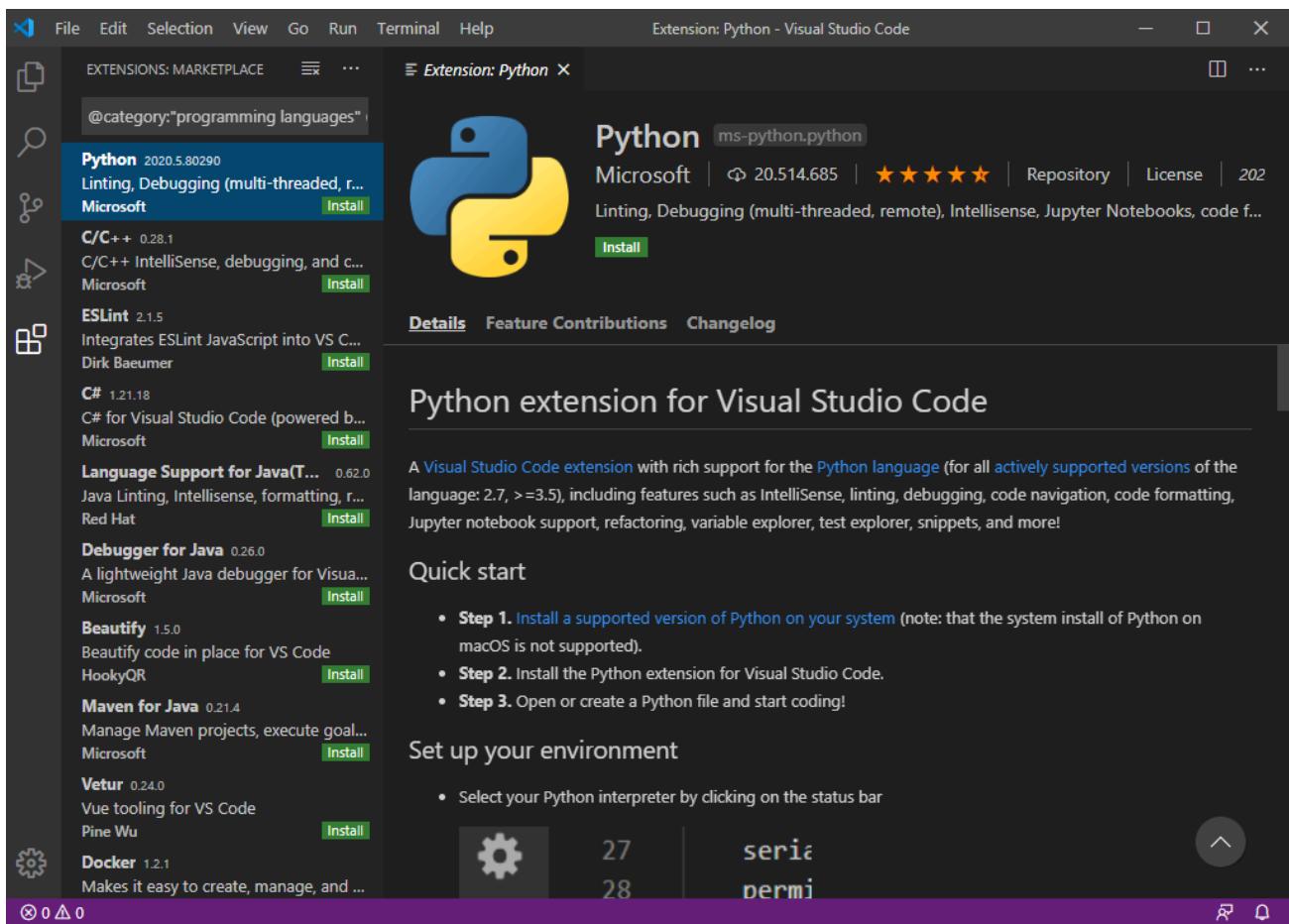
Stap 1: Open Visual Studio



Stap 2: Open het blok Customize



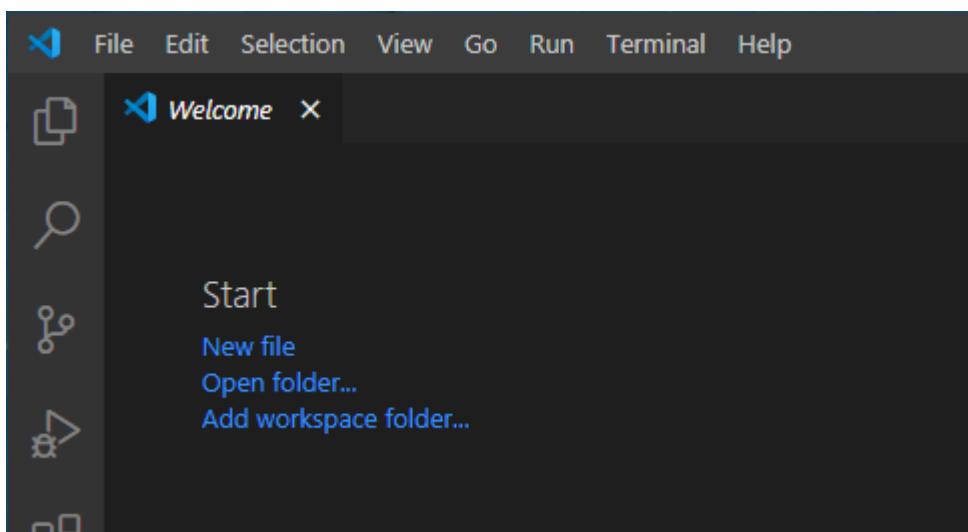
Stap 3: Installeer Python.

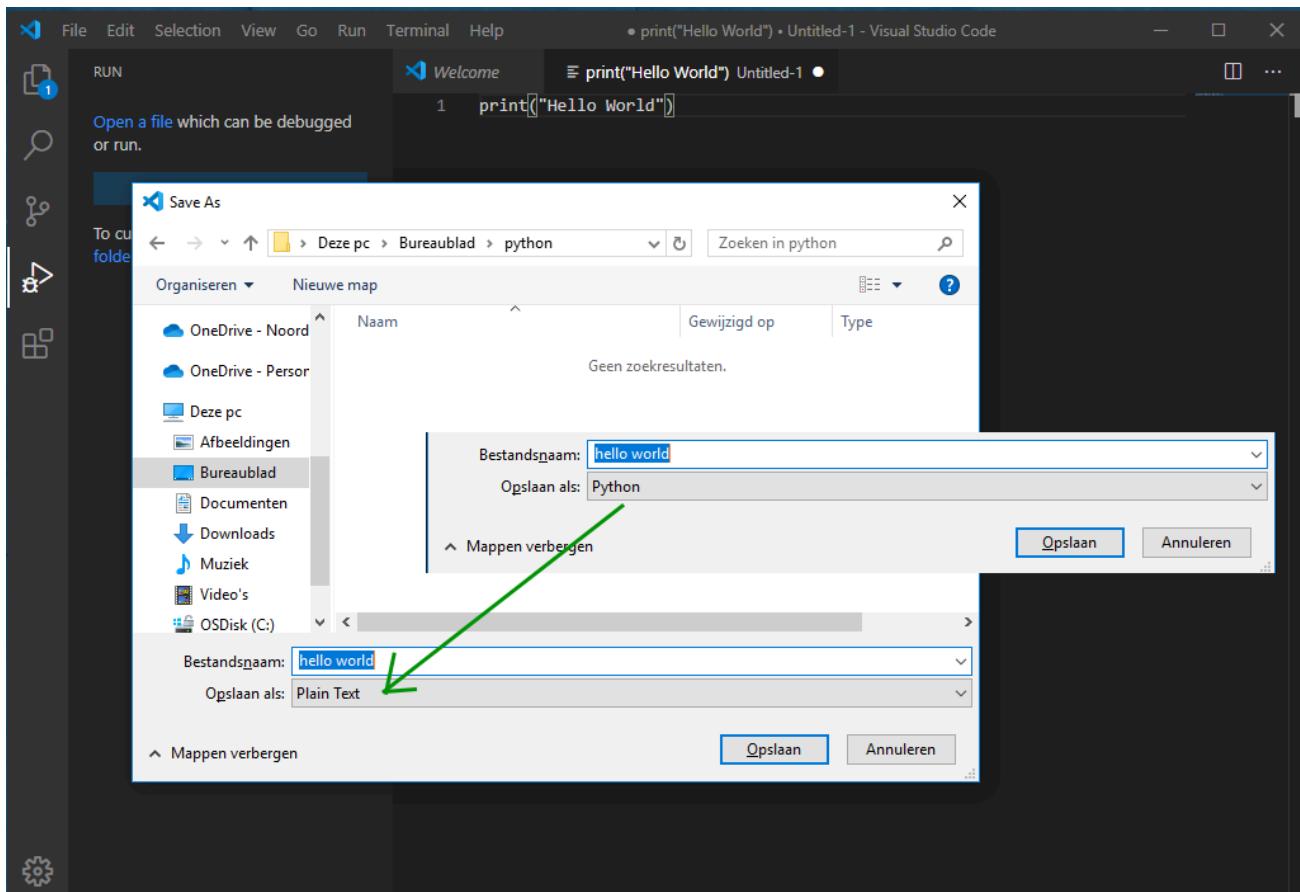


## Je eerste python code

Stap 1: Open een nieuw document en sla deze op met de .py extention. bijvoorbeeld HelloWorld.py

! Let op dat je Python klikt bij het opslaan als menu.





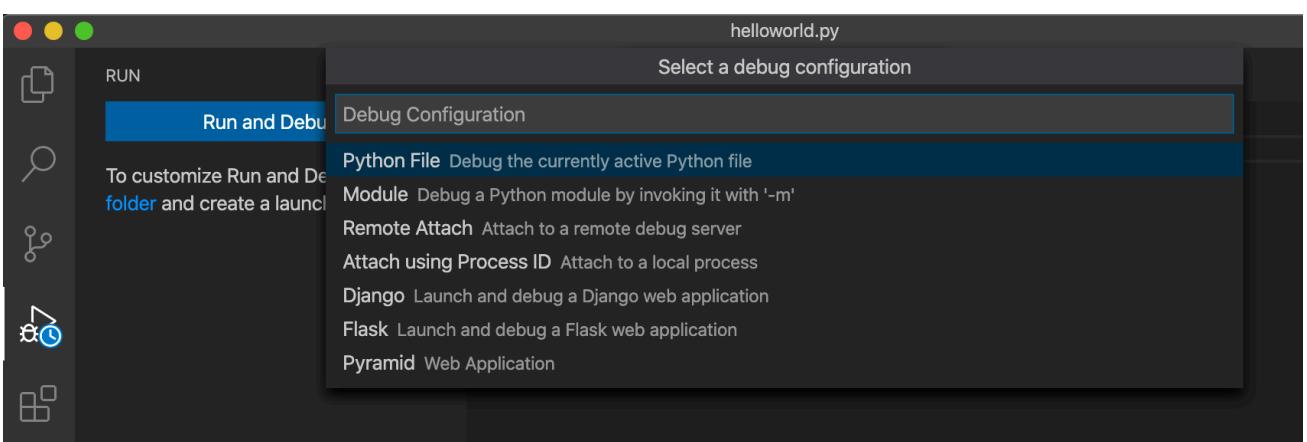
Stap 2: Zorg dat je Hello World print in python en run en debug dit bestand.

A screenshot of the Visual Studio Code interface. The title bar says "helloworld.py". The left sidebar has icons for file, search, and other tools. The main area shows a code editor with the following content:

```
1 print("Hello World")
```

The top navigation bar has tabs for "RUN" (which is highlighted in blue), "Extension: Python", and "helloworld.py". Below the tabs, it shows the file path: "Users > jadethomson-oddin > Desktop > python > helloworld.py". The bottom status bar shows "Python 2.7.16 64-bit" and "Ln 1, Col 21 Spaces: 4 UTF-8 LF Python".

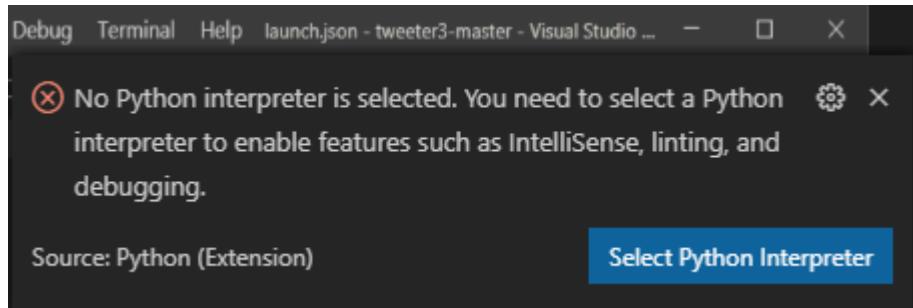
Als het goed is krijg je als je op run and debug klikt het volgende:



Je hebt het document op geslagen als en python document en daarom kun je ook "python file" kiezen.

Als Visual Studio Code de melding "No Python interpreter is selected" geeft kun je op Select

Python Interpreter drukken.



Als je run and debug hebt uit gevoerd heb je je eerste python geprint. Je krijgt dan Hello World in beeld.

A screenshot of the Visual Studio Code terminal window. The tab bar at the top shows 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is selected), and 'DEBUG CONSOLE'. The terminal window itself has a title '1: Python Debug Console'. The text in the terminal is as follows:  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit <https://support.apple.com/kb/HT208050>.  
bash-3.2\$ env python /Users/jadethomson-odding/.vscode/extensions/ms-python.python-2020.5.80  
290/pythonFiles/lib/python/debugpy/no\_wheels/debugpy/launcher 58117 -- /Users/jadethomson-odd  
ing/Desktop/python/helloworld.py  
Hello World  
bash-3.2\$

Gefeliciteerd je hebt python geïnstalleerd en Visual Studio Code!

Dit betekend dat jij :

- je eerste code hebt geschreven (Hello world)
- je software heb geïnstalleerd
  - Python
  - Visual Studio Code
  - Interpreter

## Links

1. Installatie Python : <https://www.python.org/>
2. Installatie Visual studio code: <https://code.visualstudio.com/docs>
3. Tutorial om te beginnen met Visual Studio Code: <https://youtu.be/Sdg0ef2PpBw>

## Huiswerk

Opdracht 1: Zorg dat alles geïnstalleerd is en werkt voor de volgende les.

## Opdracht 2: Termenlijst

Maak een termenlijst waarin je omschrijft wat de gebruikte termen betekenen.

In programmeren zijn veel termen en afkortingen die je vaak gebruikt die in het begin niet logisch zijn, daarom is het handig om een lijst bij de hand te hebben tijdens het programmeren.

Je mag creatief zijn met het maken van een termenlijst, je kunt bijvoorbeeld een website maken, een online boek of een .exe bestend.

Voor deze week bijvoorbeeld de volgende termen:

1. interpreter
2. debugger
3. print

Voorbeelden van een termenlijst:

<https://wiki.python.org/moin/PythonDebuggingTools>

<https://www.yumpu.com/nl/document/read/20104041/begrippenlijst-klassieke-kring>

Een .exe bestand als template:



[Termenlijst programmeren](#)

# Turtle

## Omschrijving

Programmeren is doen! Als je programmeren goed onder de knie wilt krijgen, zul je er uren in moeten steken. Je zult moeten zoeken naar projecten die jij leuk vindt en die iets toevoegen voor jou.

## Leerdoelen:

1. Libraries importeren
2. Werken met de turtle library
3. Een vierkant maken met turtle
4. Een driehoek maken met turtle
5. Werken met vormen en kleuren.

## Theorie

Turtle is een Python library die in het standaard pakket van Python libraries is ingebouwd. Turtle maakt het mogelijk om afbeeldingen en vormen te gebruiken in Python. Deze library is heel divers en laat zien hoe je een library gebruikt. Turtle wordt ook gebruikt om 2D games te maken. Om beter te begrijpen wat turtle doet zijn de termen [library](#), [functie](#) en [variable](#) hieronder uitgewerkt.

## Library

Library is het Engelse woord voor bibliotheek, maar omdat het een programmerterm is gebruiken we altijd library en niet bibliotheek. Een library is net als een bibliotheek een locatie waar verschillende soorten boeken worden bewaard. Je kunt alle informatie die jij nodig hebt altijd bekijken in de library. In programmeren is de library een groep van belangrijke functies en methoden. Deze functies en methoden kun je zien als de boeken van de bibliotheek. Deze functies maken programmeren een stuk makkelijker.

Om een library aan te roepen gebruik je:

```
import turtle
```

Import staat altijd bovenaan je programmerprogramma, omdat elk stuk code dan gebruik kan maken van deze library. Er is een aantal onderlinge afspraken tussen programmeurs, zodat de code van een ander makkelijker te lezen is mede hierom staan de libraries altijd boven aan.

# Functies

Functies maken het makkelijker om te programmeren. Tijdens programmeren gebruik je vaak dezelfde code om taken uit te voeren. Functies zijn de bouwblokken van code. In een functie staat code die vaak her en der gebruikt wordt één keer goed omschreven, zodat je het altijd kunt aanroepen.

Programmeertalen hebben vaak al veel ingebouwde functies zoals `input()` en `output()`. Die vooraf zijn gemaakt door andere programmeurs. Het is ook mogelijk om zelf functies te definiëren, als je zelf een functie hebt gedefinieerd kun je deze zo vaak oproepen als je wilt binnen dezelfde code.

Er is wel een belangrijk verschil tussen ingebouwde functies en zelf gedefinieerde functies: ingebouwde functies zijn aan te roepen in elk stuk code. Zelf gedefinieerde functies zul je in elk stuk code dat je gebruikt opnieuw moeten definiëren.

# Variable

Variables worden in programmeren gebruikt om informatie op te slaan die later in de code gebruikt kan worden. In andere programmeertalen is er een specifiek commando om een variable te declareren (vast te stellen) dit hoeft niet in Python.

Een variable wordt gecreeërd op het moment dat je er een waarde aan koppelt.

```
x = 4  
y = "Jan"  
  
print(x)  
print(y)
```

Je kunt verschillende soorten waarde koppelen. Je hebt verschillende soorten waardes:

- number: bestaat enkel uit cijfers.
- string: kan uit letters en/of nummers en/of leestekens bestaan.
- list: een verzameling van verschillende waardes.
- dictionary: koppelt sleutels aan waardes.

In het voorbeeld is `x` een variabale met een *number* waarde en `y` een variable met een *string* waarde.

Je initialiseert een variable wanneer je een startwaarde koppelt aan de variable. Een variable kan (meerdere keren) van waarde veranderen tijdens het uitvoeren van de code.

!LET OP: als je een variable een naam geeft moet je een logische naam kiezen die makkelijk te onthouden is en waarin de kans dat je fouten maakt klein is. Dit is vooral belangrijk als je de variable vaak gaat gebruiken. In de komende praktijkopdracht kom je bijvoorbeeld een variable met de waarde `turtle.getscreen()` tegen. Een lange naam zoals `turtle_screen_naam` is

moeilijk om vaak foutloos te typen en een naam of een willekeurige letter is niet logisch om te onthouden. Kies daarom een logische naam of letter. Hier zou je kunnen kiezen voor *screen* als variable naam of de letter *s* de eerste letter is makkelijk te onthouden.

## Praktijk

!Let op: Als je het document de zelfde naam geeft als een van je libraries, leest je programma de library niet in

### De *turtle library*

Als je naar projecten zoekt waarin turtle is gebruikt kom je veel verschillende projecten tegen, sommige zullen simpel lijken maar er zijn hele leuke dingen mogelijk met turtle. Ter inspiratie een presentatie van een jongen die zichzelf leren programmeren:



<https://www.youtube.com/embed/OXi4T58PwdM>

Als je niet het hele filmpje wil zien kun je ook naar de volgende tijden in het filmpje gaan: 4:46, 6:16, 6:40, hoewel hij aan het begin verteld waar je zijn projecten terug kunt vinden.

### Library aanroepen

Om gebruik te kunnen maken van de functies van een library moet je hem eerst importeren, zodat het Pythonscript weet welke [functies](#) Python moet aanroepen.

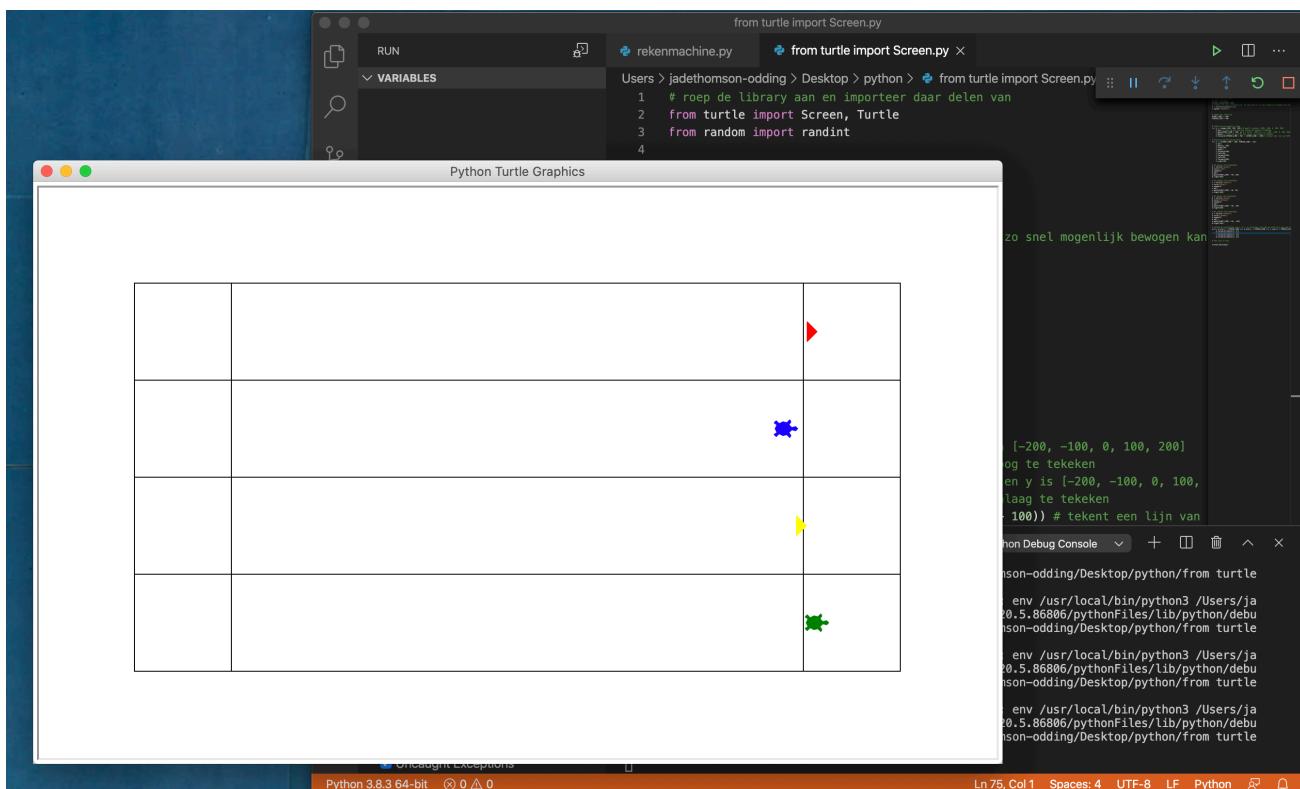
Het aanroepen doen we als volgt:

```
>>> import turtle
```

De Python turtle library bevat [methodes](#) en functies die je nodig hebt om afbeeldingen te creëren.

Turtle is een grafische library, dit betekent dat we een raamwerk nodig hebben om binnen te tekenen. Dit raamwerk moet in een apart venster (screen) aangeroepen worden. In dit venster worden alle commando's die je in de code geeft uitgevoerd.

Dit ziet er als volgt uit:

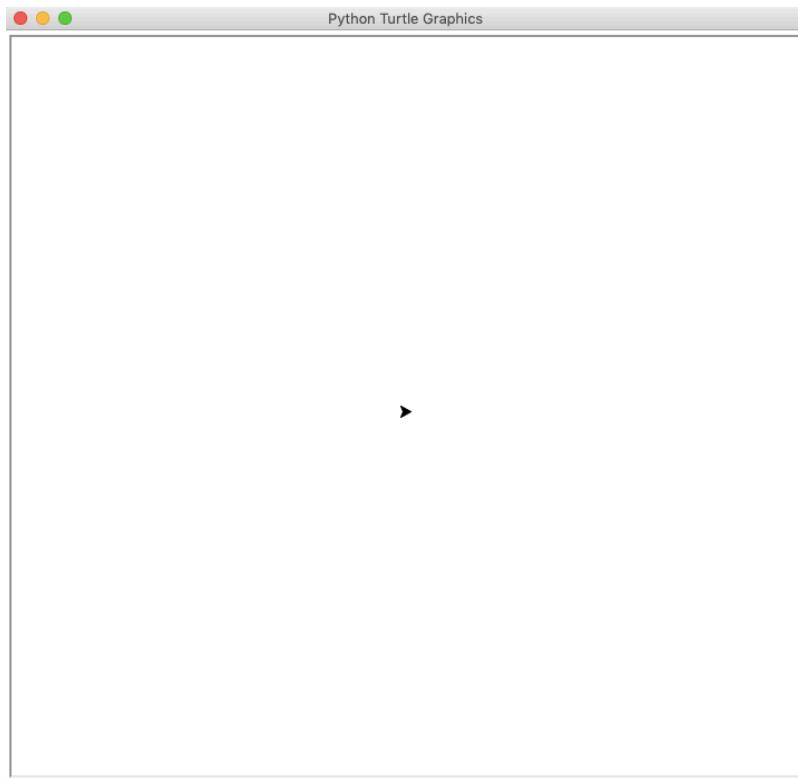


Je kunt dit venster aanroepen door een variabele te maken voor dit venster.

Om het turtle venster aan te roepen gebruiken we de volgende code:

```
>>> s = turtle.getscreen()
```

De `s` is de variabele, deze krijgt de waarde die wordt terug gegeven bij het aanroepen van `turtle.getscreen()`. Vanaf nu is de variabele `s` bezet. Als je de code draait gaat variabele `s` gaan nu een venster aanroepen die ziet er als volgt uit:



Op dit venster genaamd screen kan je de output van je code zien. In het midden zie je een driehoek deze venster is de turtle.

Als je wil dat dit scherm lang in beeld blijft kun je onder in je scherm de volgende regel zetten:

```
>>>  
turtle.exitonclick()
```

Dit zorgt er voor dat het scherm zich sluit als je op het scherm klikt en het niet direct na het uitvoeren van de code snel typbaar te houden gaan we voor het driehoekje in het venster (turtle) een variabele t maken.

```
>>> t = turtle.Turtle()
```

We hebben nu je canvas (het venster) en de pen (turtle). Nu kan de turtle bewegen over het venster,

## Bewegen

Het hebben van een driehoek in je venster is leuk, maar nu willen we er ook wat mee gaan doen. Turt

De turtle kan

- Vooruit .forward()
- Achteruit .backward()
- Links .left()
- Rechts .right()

Je kunt turtle laten bewegen door deze comando's te geven in combinatie van het aantal graden dat je

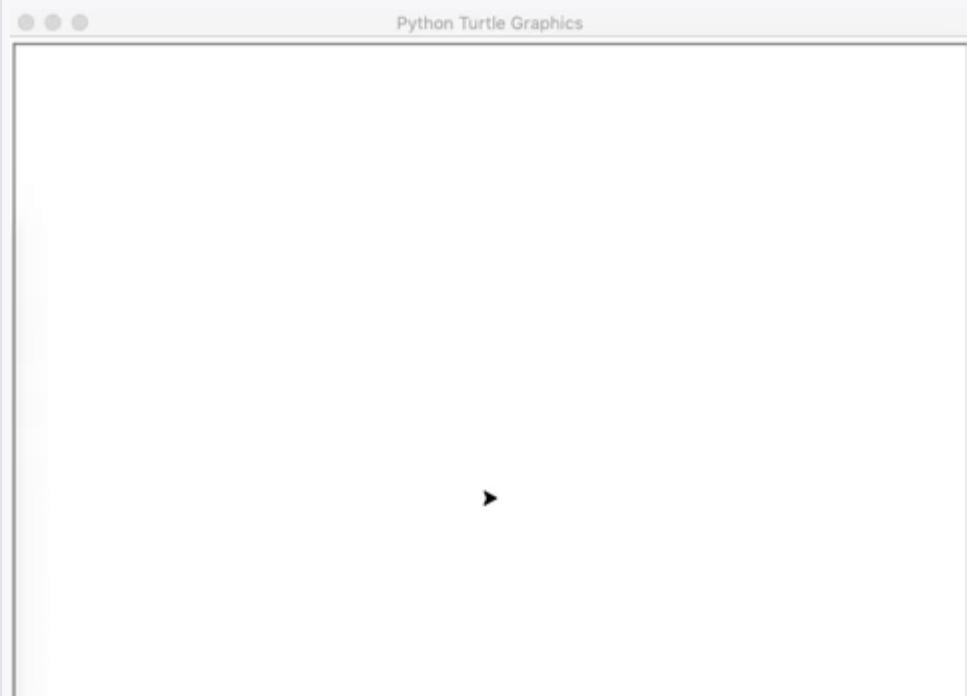
We hebben de variabele t gemaakt waar turtle in zat, daarom kunnen we nu t koppelen aan de actie die

```
>>> t.right(90)
```

Vervolgens willen we dat de turtle 100 pixels vooruit gaat.

```
>>> t.forward(100)
```

Als je deze code hebt uitgevoerd zie je de lijn naar beneden gaan zoals in dit voorbeeld:



## **Preset figuren**

Er zijn ook vooraf geprogrammeerde figuren die in de library te vinden zijn. Je zou bijvoorbeeld niet handmatig het hele scherm willen doorlopen met turtle als je een gekleurde achtergrond wil. Ook is het maken van een cirkel lastiger dan nodig is.

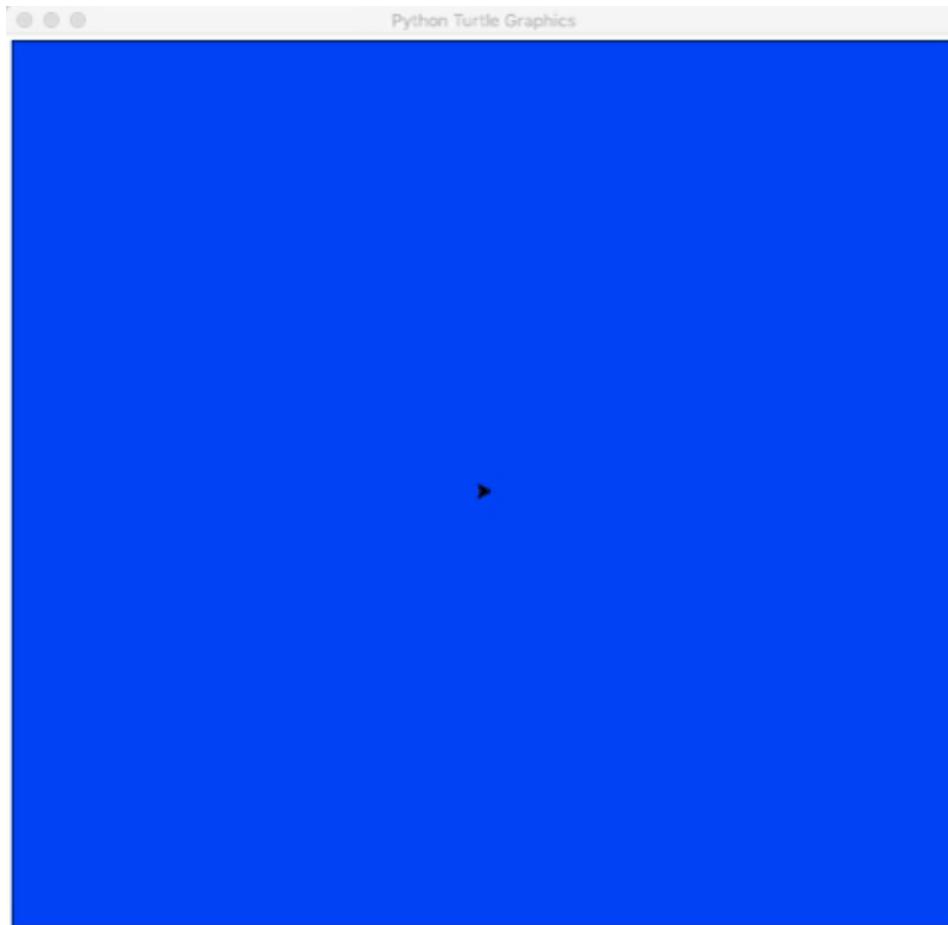
### **Achtergrond**

Turtle opent altijd een scherm met een witte achtergrond, deze kleur kun je makkelijk aanpassen met `turtle.bgcolor() # kleuren`, net als andere voorafgedefinieerde settinges zijn in het engels.

Om de achtergrond kleur naar blauw te veranderen typ je het volgende:

```
>>> turtle.bgcolor("blue")
```

Je kunt wat binnen de ("") valt vervangen met alle kleuren die je wilt. Ook kun je ze achter elkaar laten veranderen. Dan krijg je het volgende:

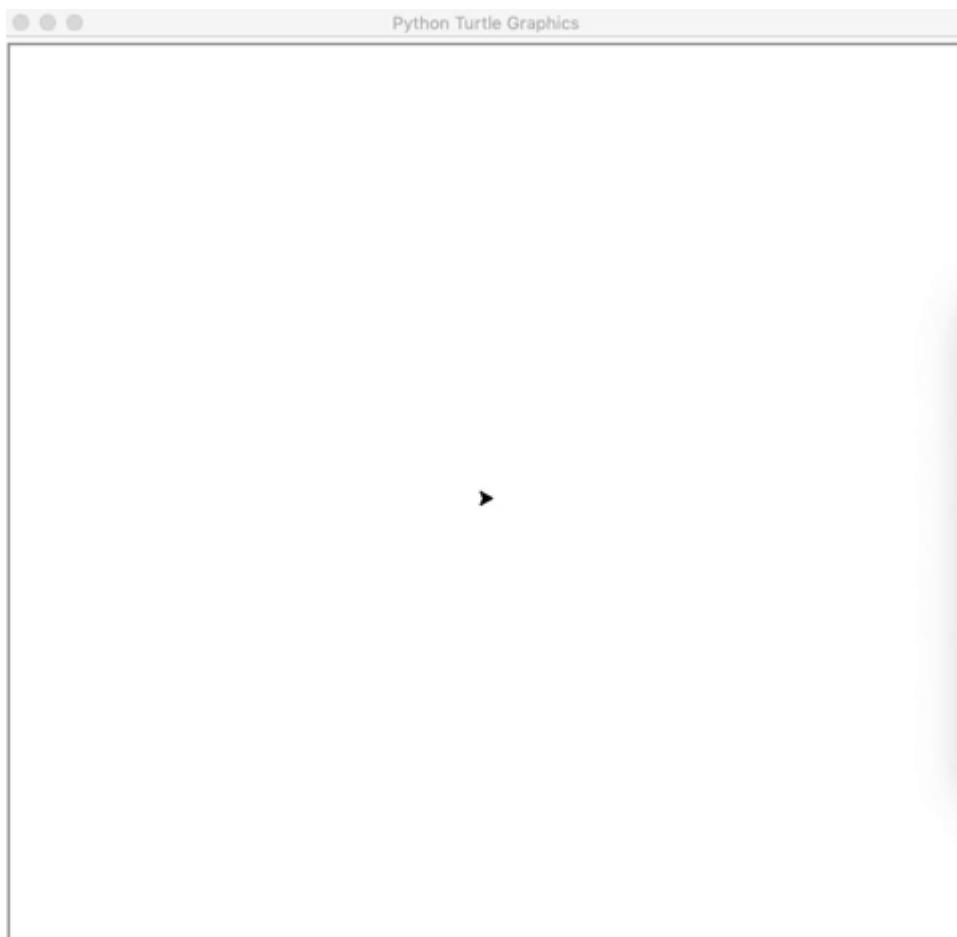


### **Cirkel**

Het berekenen van een cirkel is extra werk, dus ook hier is een preset voor.

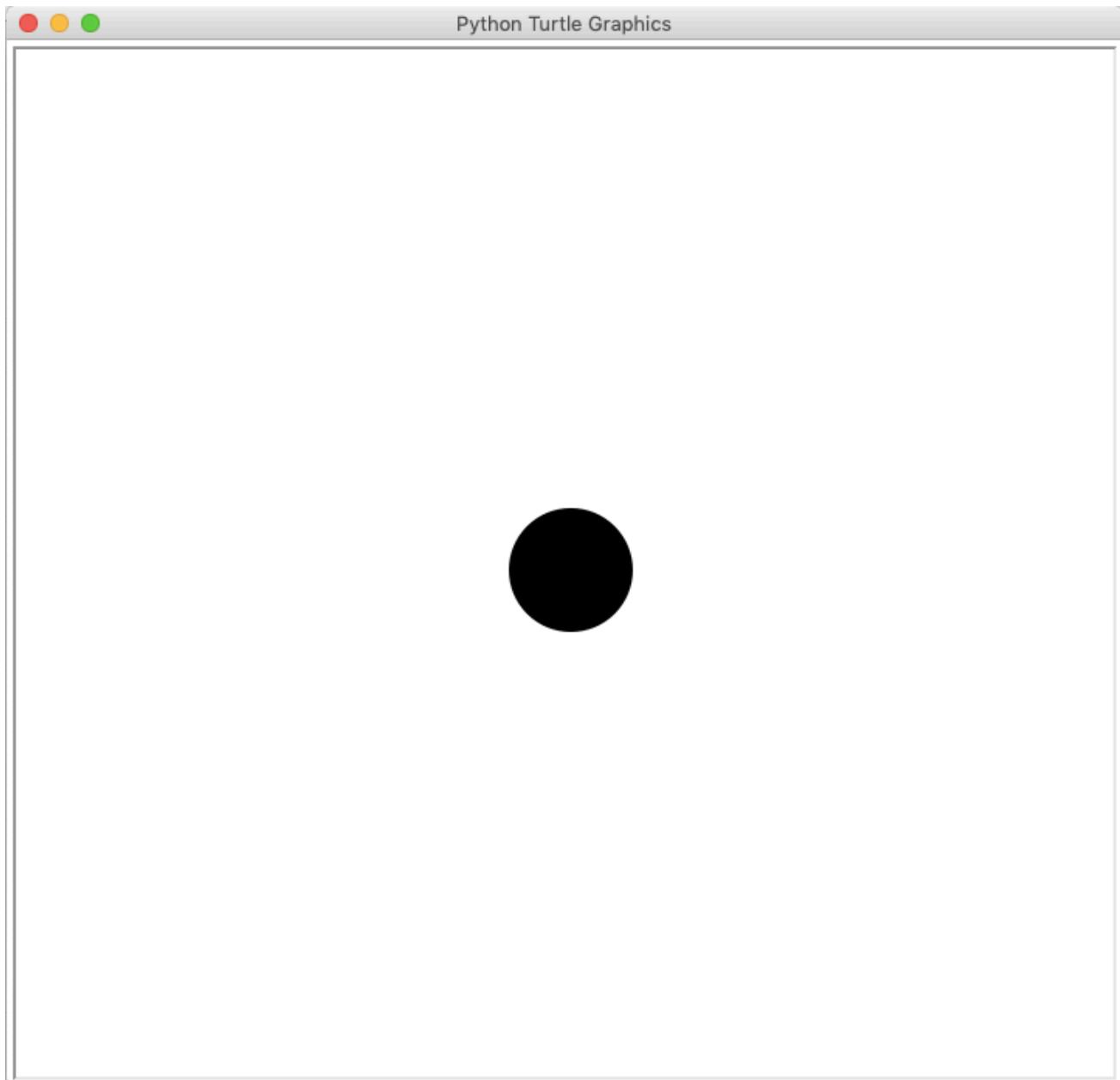
```
>>> t.circle(60)
```

Het voorbeeld spreekt redelijk voor zich, turtle krijgt een cirkel met de radius van 60 pixels, dat ziet er als volgt uit:



Als je een gevulde cirkel wilt kun je dit met .dot() doen:

```
>>> t.dot(20)
```



### Turtle formaat

Om het formaat van de turtle pijl te veranderen kun je gebruik maken van de functie `.shapesize()`. Deze functie verandert alleen het formaat van je pijl aan en doet niets met andere objecten die je mogelijk ook in beeld hebt. De nummers binnen de haakjes geven het formaat wat de turtle moet aannemen. (lengte,breedte, outline)

```
>>> t.shapesize(1,5,10) >>> t.shapesize(10,5,1) >>> t.shapesize(1,10,5) >>> t.shapesize(10,1,5)
```

Als je het voorbeeld over typt krijg je de output die je in de afbeelding ziet:



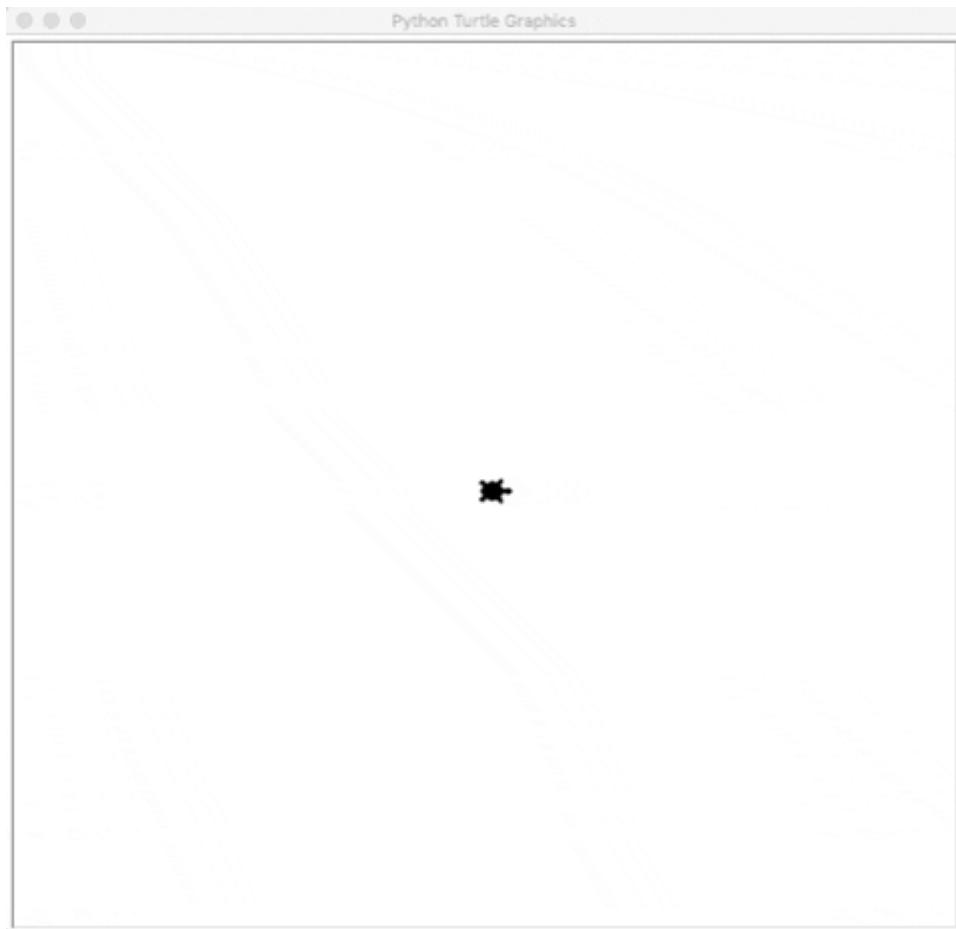
## Turtle shape

Je hebt het formaat van de turtle aangepast, maar je kunt hem ook veranderen. De turtle library heeft verschillende vormen, waaronder de vorm van een turtle. Met de functie `.shape()` kun je de turtle aanpassen. Er zijn veel preset formen beschikbaar en aantal hiervan zijn:

- Square
- Arrow
- Circle
- Turtle
- Triangle
- Classic

```
>>> t.shape("turtle") >>> t.shape("arrow") >>> t.shape("circle")
```

Als je de code hierboven intypt krijg je het volgende:

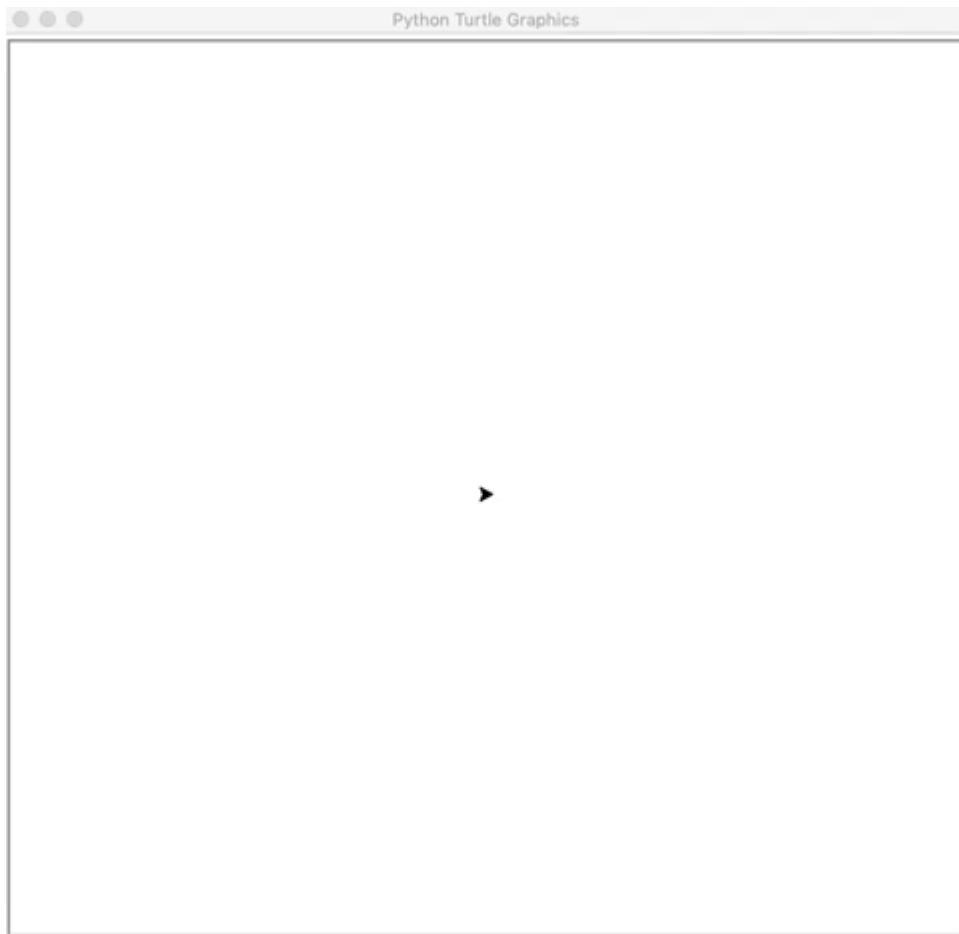


### **Snelheid van de turtle**

Als je vind dat de turtle te snel of langzaam gaat kun je dit ook aanpassen typ je het volgende:

```
>>> t.speed(1) >>> t.forward(100) >>> t.speed(10) >>> t.forward(100)
```

Deze code heeft de snelheid in eerste instantie op normale snelheid en versneld de turtle als hij de bocht om is.



The speed can be any number ranging from 0 (the slowest speed) to 10 (the highest speed). You can play around with your code to see how fast or slow the turtle will go.

## Opdracht: Maak een cirkel die van kleur veranderd.

Pas de achtergrond aan.

Extra: Pas de lijn en vulling apart aan. Kijk hier voor naar de volgende [link](#).

Extra: laat de cirkel bewegen over het scherm.

### Turtle / pen kleur

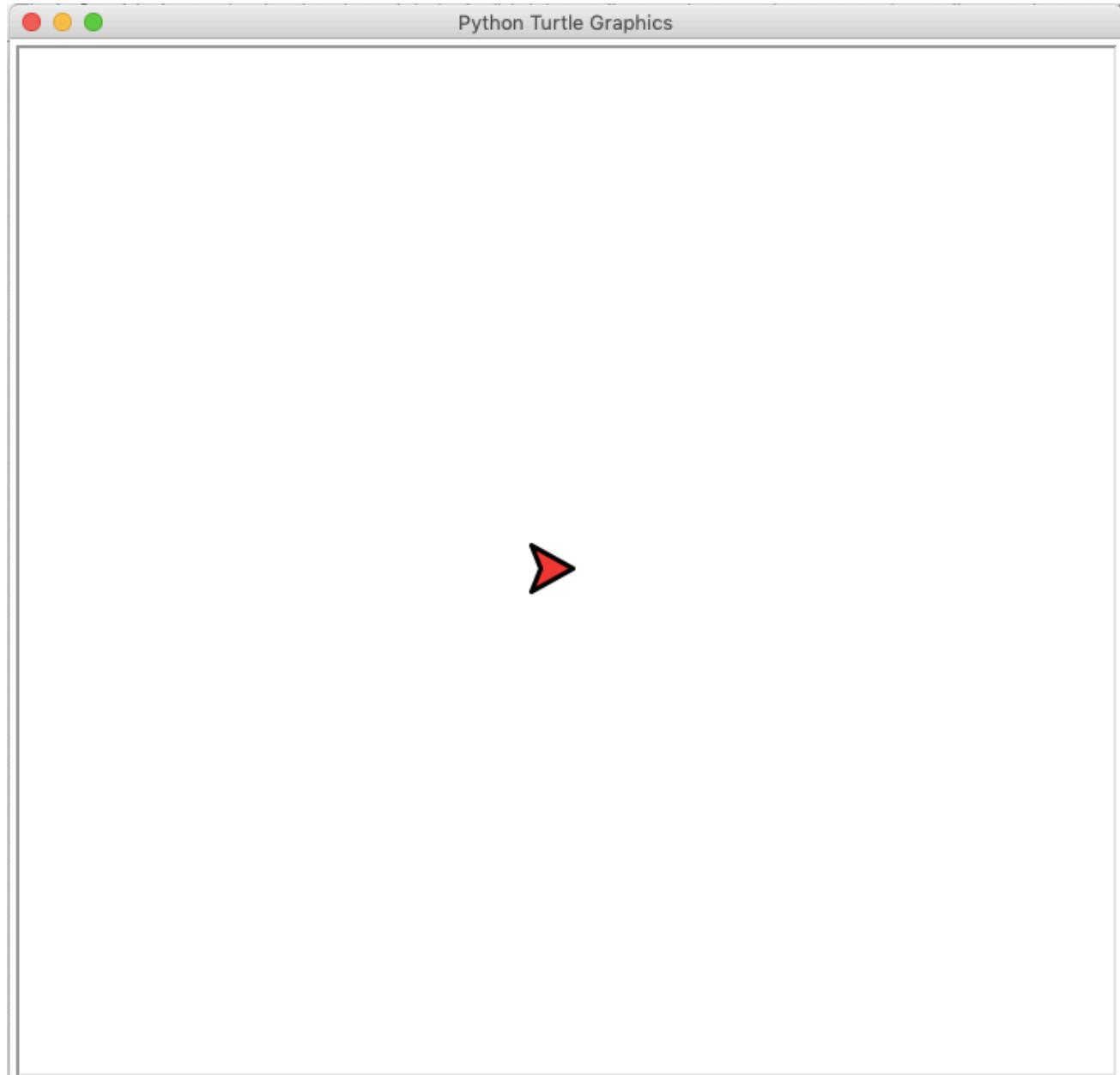
De turtle pijl bestaat uit twee onderdelen die je kunt aanpassen. De buitenste lijn en de vulling.

!Let op: als de pijl te klein is veranderd hij niet van kleur, dus de pijl moet eerst groter gemaakt zijn.

Om de vulling te veranderen gebruiken we `.fillcolor()`, dit kleurt enkel de vulling en niet de buitenstelijn.

```
>>> t.fillcolor("red")
```

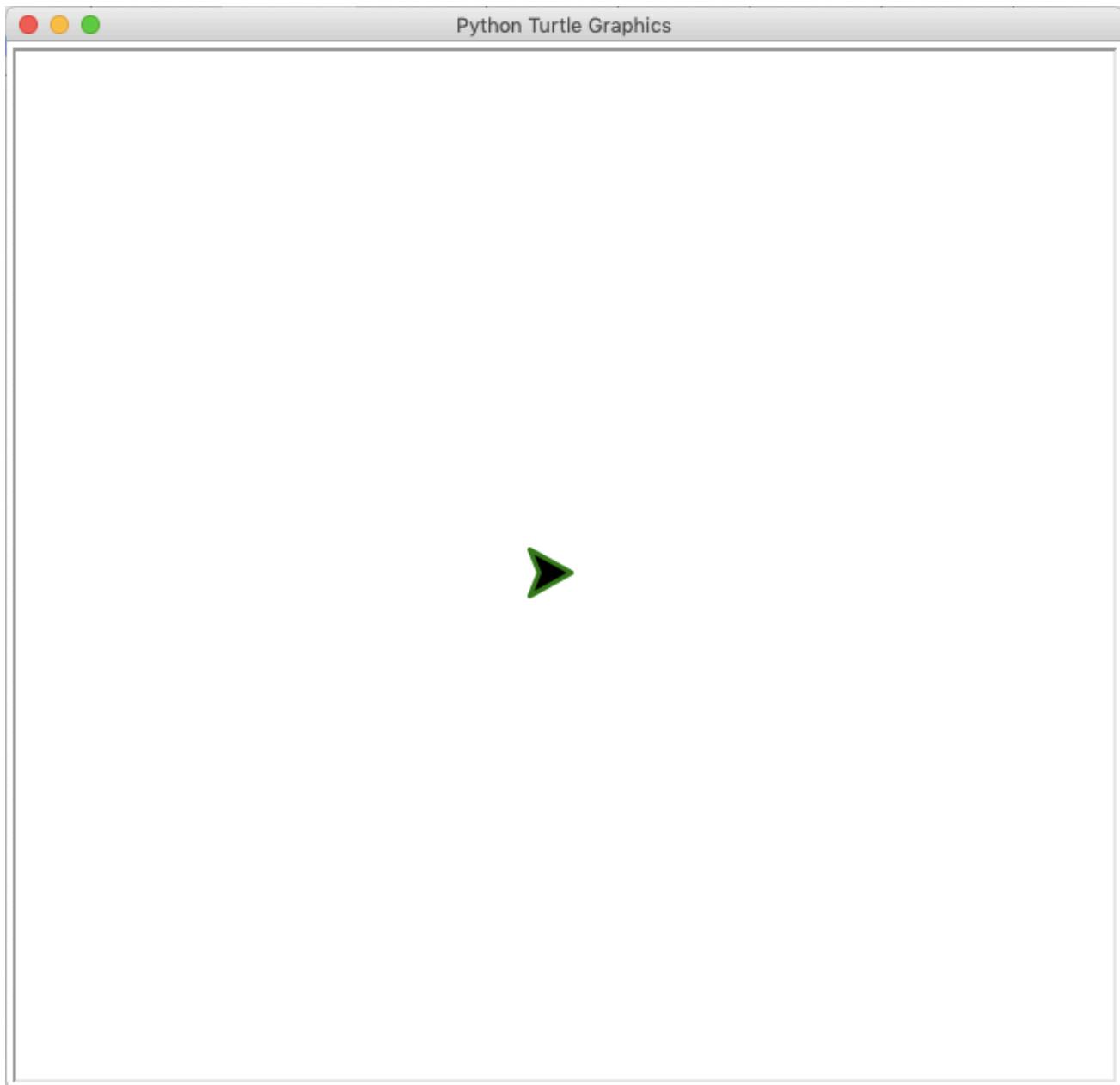
De code hierboven ziet er als volgt uit:



Om de buitenste lijn te veranderen wordt de variabele `.pencolor()` gebruikt:

```
>>> t.pencolor("green")
```

Deze code is groen, je kunt hem naar alle kleuren aangepast worden en komt er als volgt uit zien:



Als je dit allemaal hebt gedaan betekent dat dat je

- Een scherm kunt genereren
- Achtergronden kunt aanpassen
- Vormen kunt programmeen

Dit is een goede basis voor het maken van een spelletje.

## ***Links***

1. <https://realpython.com/beginners-guide-python-turtle/>

# Huiswerk

## ***Opdracht 1: Termenlijst***

Breid je termenlijst uit met termen die deze les zijn langs gekomen.

## ***Opdracht 2: Opdrachten + experimenteren***

Zorg dat alle opdrachten gedaan zijn en maak zelf iets creatiefs met turtle.

# Loops

## Omschrijving

Vaak wil je een opdracht vaak uitvoeren en hiervoor is programmeren uitermate geschikt voor. Om een herhalende opdracht uit te voeren gebruiken we loops. Binnen loops moeten we vaak vergelijken dit doe je met operatoren.

In dit hoofdstuk gaan we while loops, for loops en operatoren behandelen en maken we een turtle wedstrijd.

## Leerdoelen:

1. Operatoren begrijpen
2. While loop begrijpen
3. For loop begrijpen
4. Werken met een while loop

## Theorie

## Operatoren

Operatoren ken je vast nog wel van wiskunde. In programmeren gebruiken we verschillende soorten operatoren.

- Rekenkundige operatoren: plus, min
- Vergelijkings operatoren: groter dan en kleiner dan
- Logische operatoren: and or not

## Rekenkundige operatoren

Operator	Omschrijving	Voorbeeld
+	Optellen	$6 + 3 = 9$
-	Aftrekken & Negatief maken	$6 - 3 = 3$
*	Vermenigvuldigen	$6 * 3 = 18$
/	Delen	$6 / 3 = 2$

%	<a href="#">Restwaarden</a>	$7 \% 3 = 1$ $(3 + 3 = 6 \text{ dus er is } 1 \text{ over})$
**	Machtsverheffing	$6 ** 3 = 216$ $(6 * 6 * 6)$

## Vergelijgings operatoren

Operator	Omschrijving	Voorbeeld	Uitwerking
==	Gelijk aan	<code>a == b</code>	<code>a = 5 b = 3  print(a == b)  # geef False, a is niet gelijk aan b</code>
!=	Niet gelijk aan	<code>a != b</code>	<code>a = 5 b = 3  print(a != b)  # geef True, a is niet gelijk aan b</code>
>	Groter dan	<code>a &gt; b</code>	<code>a = 5 b = 3  print(x &gt; y)  # geef True, a is groter dan b</code>

<	Kleiner dan	$a < b$	<pre> a = 5 b = 3  print(x &lt; y)  # geef False, a is niet # groter dan b </pre>
$\geq$	Groter of gelijk aan	$a \geq b$	<pre> a = 5 b = 3  print(x \geq y)  # geef True, a is groter # of gelijk aan b </pre>
$\leq$	Kleiner of gelijk aan	$a \leq b$	<pre> a = 5 b = 3  print(x \leq y)  # geef False, a is niet # groter of gelijk aan b </pre>

## Logische operatoren

Operator	Omschrijving	Voorbeeld	Uitwerking
and	Geeft True terug als beide statements waar zijn	$a < 5 \text{ and } b < 10$	<pre> a = 5  print(a &gt; 3 and a &lt;       10)  # geeft True omdat # 5 groter is dan 3 # AND 5 is kleiner # dan 10 </pre>

or	Geeft True terug als een of beide statements waar zijn	$a < 5 \text{ or } a < 4$	$a = 5$  print( $a > 3 \text{ or } a < 4$ )  # geeft True omdat 5 groter is dan 3
not	Draait het andwoord omen geeft False als het andwoord True is.	not( $a < 5 \text{ and } a < 7$ )	$a = 7$  print(not( $a > 3 \text{ and } a < 7$ ))  # geeft False niet wordt gebruikt om het andwoord om te keren.

Deze operatoren zijn handig als je iets wil berekenen, zoals het aantal bezoekers van je website. Maar zijn ook handig om te gebruiken om te kijken of iemand een taak goed heeft uitgevoerd om vervolgens verder te mogen in je programma. Stel je moet in een spelletje 10 sterren hebben gehaald om naar de volgende level te gaan. Elke keer dat de speler een ster haalt kijk je of het gehaalde aantal sterren groter of gelijk aan ( $\geq$ ) 10 sterren is, zo ja, dan mag hij door naar de volgende level. Die *zo ja*, noemen we *if statements*

## Loops

In programmeren gebruiken we [loops](#). Er zijn twee loop commands die veel gebruikt worden.

- [while loops](#)
- [for loops](#)

Met een while statement kun je net als met een for loop code herhaaldelijk uitvoeren.

Een for statement voert de loop alleen uit zolang een bepaalde conditie vervuld is.

## While Loop

De while loop voert een actie of statement uit als de conditie waar is.

Bijvoorbeeld: Print de waarde x uit zolang x kleiner is dan 8

```
x = 1
```

```
while x < 8:  
    print(x)  
    x += 1
```

De output van dit programma ziet er als volgt uit:

The screenshot shows a dark-themed code editor interface. At the top, there's a file tab labeled "while uitleg.py". Below it, the file path is shown as "Users > jadethomson-oddung > Desktop > python > while uitleg.py > ...". The code editor displays the following Python script:

```
1 x = 1  
2 while x < 8:  
3     print(x)  
4     x += 1
```

Below the code editor, there are tabs for "PROBLEMS", "OUTPUT", "TERMINAL", and "DEBUG CONSOLE". The "TERMINAL" tab is selected, indicated by a blue underline. To the right of the tabs, there's a dropdown menu labeled "1: Python Debug Console" with a downward arrow, and a set of icons for creating a new terminal, closing, minimizing, maximizing, and refreshing.

The terminal window below shows the command "homson-oddung/Desktop/python/while uitleg.py" followed by the output of the script:

```
1  
2  
3  
4  
5  
6  
7
```

The terminal window has a black background and white text. The path and command are in white, and the output numbers are also in white. The file name "while uitleg.py" is in blue.

Als de loop geen `+= 1` krijgt blijft hij voor altijd draaien, want er is geen waarde gegeven die onwaar is.

Om zeker te weten of de lus is beëindigd kun je onder de lus nog een print maken:

```
print("Lus beëindigd.")
```

The screenshot shows a code editor window with a dark theme. At the top, there's a file tab labeled "while uitleg.py". Below it, the code is displayed:

```
1 x = 1
2 while x < 8:
3     print(x)
4     x += 1
5 print("Lus beëindigd.")
```

At the bottom of the code editor, there's a navigation bar with tabs: PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is currently selected. In the terminal pane, the output of the script is shown:

```
/Users/jadethomson-oddung/Desktop/python/while uitleg.py"
1
2
3
4
5
6
7
Lus beëindigd.
bash-3.2$
```

In sommige gevallen kan het natuurlijk gewenst zijn om een oneindige lus te maken, een voorbeeld hiervan is een webserver, deze heeft een oneindige lus waarin hij een connectie van een client aanvaardt en afhandelt, hierna accepteert hij een nieuwe connectie.

## For Loops

Een andere manier om een lus te maken is het for statement.

Het volgende voorbeeld geeft dezelfde output als het voorbeeld van het for statement:

A screenshot of a terminal window from a code editor. The title bar shows the file path: "Users > jadethomson-oddung > Desktop > python > for uitleg.py > ...". The code in the editor is:

```
for x in range(1,8):
    print(x)
print("Lus beëindigd.")
```

The terminal tab is selected at the bottom, showing the output:

```
1: Python Debug Console
```

```
" /Users/jadet
homson-oddung/Desktop/python/for uitleg.py"
1
2
3
4
5
6
7
Lus beëindigd.
bash-3.2$
```

De eerste regel maakt gebruik van de range(start,einde) functie. Dit genereert een array getallen:

```
>>> print(*range(1,8))
1 2 3 4 5 6 7
```

De startwaarde wordt wel uitgelezen de eindwaarde wordt niet uitgelezen.

```
for x in range(1,8):      # voor elk element in de verzameling()
    print(x).            # print de waarde x
print("Lus beëindigd.") # print de zin "lus is beëindigd."
```

De eerste regel kun je lezen als for variabele x in de lijst(1,8).

Hij leest het eerste cijfer uit de lijst uit en schrijft deze weg in de variabele x, dus de x veranderd elke keer dat hij de loop uitvoert. Tijdens de 1e loop wordt de waarde x 1, tijdens de 2e loop wordt de waarde x 2 etc. tot hij bij de 8 komt. De 8 mag niet geprint worden uitvoeren en het programma gaat door naar de volgende stap van de code: print("Lus beëindigd.").

In dit geval zal het geïndenteerde blok opgeroepen worden voor elk element in de lijst, maar zal variabele gelijkgesteld worden aan het huidige element in de lijst (dus de eerste keer zal

variabele het eerste element uit de lijst bevatten, de tweede keer het tweede element en zo verder). Dit is een extra voorbeeld van hetzelfde principe, alleen wordt de lijst hier expliciet aangemaakt:

## Praktijk

**!Let op:** Als je het document dezelfde naam geeft als een van je libraries, leest je programma de library niet in.

### Turtle race

We gaan verder werken met turtle. We gaan een turtle race maken. In de race gaan de 4 racers per stap willekeurig tussen 1 en 6 stappen naar voren. Wie het eerst over de finishlijn is wint.

We beginnen met het aanroepen van de libraries. Je kunt een hele library aanroepen, maar je kunt ook delen van de library aanroepen. We roepen twee libraries aan Turtle en random:

```
# roep de library aan en importeer daar delen van
from turtle import Screen, Turtle
from random import randint
```

Note: Nu is duidelijk aangekondigd waar random vandaan komt. Er hoeft nu niet nog randint.random gebruikt te worden.

De volgende stap is om een venster te maken.

```
# venster instellingen
screen = Screen()      # er wordt een variabele screen gemaakt.
screen.setup(1000, 600) # de maten van het venster worden bepaald.
```

Standaard wordt turtle.turtle weergegeven, het pijltje, maar voor deze race willen we die niet zien dus zetten we deze uit.

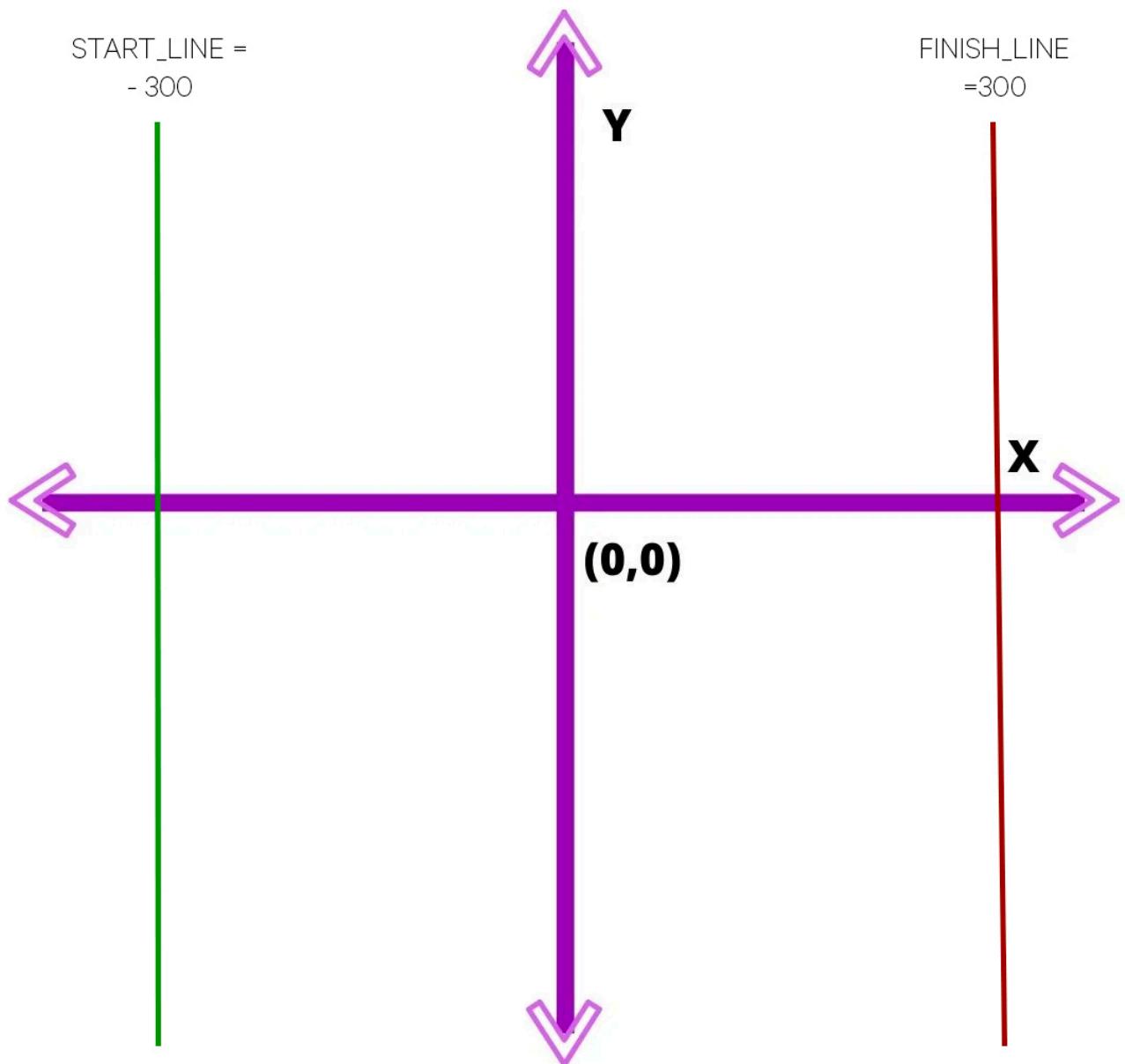
```
# Pijlcocon en snelheid
t = Turtle(visible=False) # maakt een variabele met turtle en maakt het pijlcoontje onzichtbaar
t.speed('fastest')         # zorgt dat er zo snel mogelijk bewogen kan worden
```

We maken ook alvast een variabele die aangeeft wat de locatie is waar de start begint en waar het einde eindigt.

```
# maak een startline en een finishlijn variabele.
START_LINE = -300
```

```
FINISH_LINE = 300
```

Deze lijn is niet zichtbaar, maar kondigt wel aan waar de turtles beginnen en eindigen, zoals je kunt zien in de afbeelding hieronder:



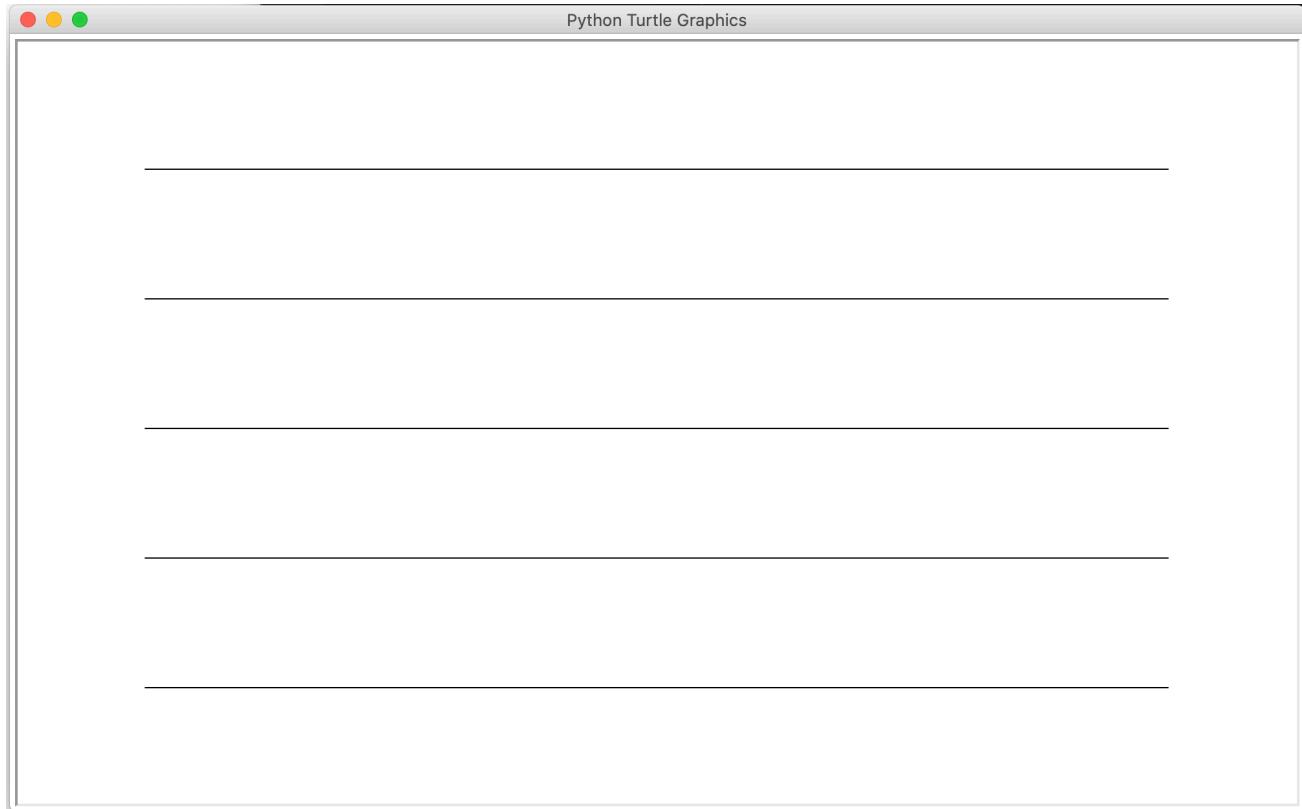
Nu het venster getekend is en de pijl verborgen, is het tijd om de race lijnen te tekenen.

Hiervoor gebruiken we een for loop.

```
# teken de horizontale lijnen
for y in range(-200, 300, 100): # vanaf -200 op de lijn y
    t.up()
    t.goto(START_LINE - 100, y)
```

```
t.down()  
t.forward((FINISH_LINE + 90) - (START_LINE - 100))
```

Deze code schrijft de horizontale lijnen:



We gaan nu de Startlijn en de Finishlijn tekenen.

Dit doen we met een for-loop, voor alle variabelen x die tussen -100 en -10 vallen

```
# Starting and Finishing Gates  
for x in [START_LINE - 100, FINISH_LINE - 10]:  
    t.up()  
    t.goto(x,200)  
    t.right(90)  
    t.down()  
    t.forward(400)  
    t.left(90)  
    t.forward(100)  
    t.left(90)  
    t.forward(400)  
    t.right(90)
```

## Opdracht: Teken de eerste race kandidaat.

De eerste kandidaat noemen we "d" deze moet de volgende waardes hebben;

- het moet een rode pijl
- de begint coordinaten zijn x -50(breed), y150(hoog) zodat hij mooi binnen de lijntjes van het start blok getekend worden.
- het moet een snelheid hebben.
- het blok heeft als waardes; breedte op -400 tot -300 en hoogte op -200 en -100.

De volgende kandidaten kunnen worden toegevoegd.

```
# de tweede race kandidaat
c = Turtle('turtle')
c.color('blue')
c.speed(5)
c.up()
c.goto(START_LINE -50, 50)
c.right(360)
```

## Opdracht: Teken de 3e en 4e kandidaat

Teken de 3e kandidaat heet b en is een pijl in het geel, gebruik hier voor arrow.

Zet hem op de lijn op -50 -50

De 4e kandidaat heet a en wordt een groene turtle.

Deze komt op de lijn op -50 -150.

Nu gaan we werken met een While statement.

De while staat voor *zolang*. Dus zolang de waardes van deze acties waar zijn worden ze uitgevoerd.

De *a*, *b*, *c* en *d* komen van de kandidaten waarna we een variabele *xcor* gebruiken dit doen we door de *a* en *xcor()* te koppelen met een punt dus *a.xcor()*.

```
while a.xcor() < FINISH_LINE
```

Dit zelfde gaan we doen voor b en c.

```
and b.xcor() < FINISH_LINE and c.xcor() < FINISH_LINE:
```

## Opdracht: Zorg dat d er ook tussen komt te staan.

d moet *ook* kleiner zijn dan finisch\_line

Nu willen we dat a vooruit gaat met een willekeurige hoeveelheid stappen wisselend tussen 1 stap en 6 stappen.

weer gebruiken we *a* om de kandidaat aan te roepen en nu gebruiken we de variabele *forward* om a vooruit te laten gaan.

variabele *forward()* krijgt een willekeurige waarde tussen 1 en 6 dus we roepen *randint()* aan uit de library en geven daarin aan tussen welke waarden er random gekozen mag worden.

```
a.forward(randint(1, 6))
```

We hebben nu de variabele *randint()* in de variabele *forward()* gestopt:

*forward(randint(1,6))*

Dit gaan we nu ook doen voor b, c en d (**let op!** deze moeten allemaal op hun eigen regel staan!)

We eindigen met de volgende regel:

```
screen.mainloop()
```

## Opdracht: Maak de race af.

zorg dat alle kandidaten een random waarde hebben die ze vooruit gaan en zorg dat er nette aantekeningen staan bij je code.

# Huiswerk

## Opdracht 1: Termenlijst

Breid je termenlijst uit met termen die deze les zijn langs gekomen. Zoals While; For; Rekenkundige operatoren etc.

***Opdracht 2: Opdrachten + experimenteren***

Zorg dat alle opdrachten gedaan zijn.

# If/Else

## Omschrijving

In turtle hebben we een library geïmporteerd en kennis gemaakt met variabelen. Andere belangrijke onderdelen van programmeren zijn de [operatoren](#) en [if/else statements](#).

## Leerdoelen:

1. Werken met if/else.
2. Begrijpen wat operatoren zijn en er mee kunnen werken.

## If statements

Nu we de operatoren weer vers in het geheugen hebben, kunnen we er een if statement van maken. Een if statement voert iets uit, als er aan een bepaalde voorwaarde voldaan wordt.

Stel je moet in een spelletje 10 sterren hebben gehaald om naar de volgende level te gaan. Elke keer dat de speler een ster haalt, kijk je of het gehaalde aantal sterren groter of gelijk is aan ( $\geq$ ) 10 sterren, zo ja, dan mag je door naar de volgende level. Die 'zo ja', noemen we *if statements*

```
a = 44
b = 250
if b > a:
    print("b is groter dan a")
```

In dit if statement kijken we of de variabele b groter is dan variabele a. Als dat het geval is, komt er in beeld te staan **b is groter dan a**. Wordt er niet aan deze voorwaarde voldaan, dan gebeurt er niets.

## Else

Als je, zoals in het voorbeeld hieronder, een if hebt uitgevoerd en de uitkomst is niet wat binnen de if valt (in dit geval is 44 niet groter dan 250) doet het programma niets.

```
a = 44
b = 250
if a > b:
    print("b is groter dan a")
```

Om er voor te zorgen dat je toch weet dat het programma werkt, is het verstandig om een

else toe te voegen.

```
a = 44
b = 250
if a > b:
    print("b is groter dan a")
else:
    print("b is niet groter dan a")
```

Else wordt uitgevoerd als het if statement False is. Dus elke andere uitkomst dan  $a > b$  valt onder else. Als je in je spelletje nog geen 10 sterren hebt en je had in je if statement aangegeven dat hij verder naar de volgende level moet, zal hij dus stoppen. Echter als je in je else aangeeft dat hij verder moet in dit level, omdat het aantal sterren kleiner is dan 10, zal het spel niet stoppen.

## Elif

Stel dat je in datzelfde spel altijd gele sterren verzamelt, maar als de speler een blauwe ster tegen komt gaat hij naar een tussenlevel, dan moet je dit in een tussenstap doen. De speler heeft nu nog geen 10 sterren gehaald (het if statement), maar hij moet ook niet door spelen in dit level (het else statement).

Deze stap tussen if en else noemen we *elif*. Je kunt maar één *if* gebruiken en je gebruikt de *else* voor alle overgebleven mogelijkheden. Elif is Pythons manier van zeggen "als de vorige condities niet waar zijn, probeer de volgende conditie."

```
a = 250
b = 44
if b > a:
    print("b groter dan a")
elif a == b:
    print("a is gelijk aan b")
else:
    print("a is groter dan b")
```

In het voorbeeld hierboven kijkt het programma of  $b$  groter is dan  $a$ . Zo niet, dan gaat hij kijken of  $a$  gelijk is aan  $b$ . Als  $b$  niet groter is dan  $a$  én  $a$  is ook niet gelijk aan  $b$  dan is  $a$  groter dan  $b$ . De code kijkt dus stap voor stap wat er mogelijk is en als er niets anders mogelijk is gaat hij naar else.

Anders dan bij if en else mag de elif zo vaak als nodig gebruikt worden.

Aan het einde van deze les heb je:

- libraries geïmporteerd
- het turtle scherm aangeroepen
- achtergonden bewerkt

- vormen gemaakt
- if en else statements gebruikt
- operatoren gebruikt

## Praktijk

We gaan met turtle if statements maken.

Om te beginnen roepen we de turtle en de time library aan. Vervolgens maken we de variable s die het venster aanroeft, waarna we ook de achtergrond kleur kunnen bepalen.

```
import turtle  
import time  
  
s = turtle.Screen()  
s.bgcolor("blue")
```

We maken een invoerlijn, waarbij er staat dat je een nummer moet invoeren.

```
number = int(input('Enter your a number: '))
```

Nu gaan we werken met het if statement.

De cijfers 0 tot 3 tekenen een cirkel. Als dit uitgevoerd is komt er 5 seconden pauze waarna het venster sluit.

```
if number < 3:  
    turtle.circle(60)  
    time.sleep(5)
```

Ook willen we, dat als het nummer tussen de 2 en 5 valt, er een gevulde cirkel wordt getekend. En de achtergrond verandert in geel.

```
elif number >= 3 and number < 5:  
    turtle.dot(20)  
    turtle.title("My yellow turtle")  
    turtle.bgcolor("yellow")  
    time.sleep(5)
```

Als geen van de statements waar is printen we de tekst "Er is iets fout gegaan".

## Opdracht: Maak het if statement af

- Als geen van de statements waar is printen we (onderaan) de tekst "Er is iets fout gegaan".
- Als het nummer groter of gelijk is aan 5 en kleiner of gelijk is aan 7 maak je een blauw vierkant en wordt de achtergrond rood (met de code uit je vorige projecten).
- Als het nummer 8 is, teken je een gele driehoek met een groene achtergrond (met de code uit je vorige projecten).
- Anders teken je een cirkel met een rode achtergrond.
- Extra: zorg dat je vaker een nummer kunt voeren door het een loop te maken

## Extra opdracht:

- Kijk wat [deze](#) code doet. Beschrijf in je aantekeningen wat het doet en experimenteer met de kleuren. (om er in te kunnen werken moet je het opslaan als .py niet .txt)
- Probeer er voor te zorgen dat de lijnen elkaar niet raken. Stop de code als dit wel zo is (je hebt iets wat al op snake gaat lijken.)
- Kijk of je snake kunt maken: <https://www.edureka.co/blog/python-turtle-module/>

Na deze opdrachten kun je:

- Werken met loops.
- Vormen maken.
- Tijden instellen.
- De libraries turtle en time gebruiken.
- Achtergronden veranderen.
- Venster namen bewerken.

# Calculator

## Omschrijving

Als je in de ICT aan het werk wil, heb je vaak te maken met programmeren. Sommige bedrijven vragen je zelfs om eerst een opdracht uit te voeren voor je door bent in je sollicitatieprocedure. Een van de basisopdrachten die programmeurs maken om te kijken of ze een programmeertaal een beetje begrijpen, is het maken van een calculator. Daarom gaan wij dit ook doen.

In dit hoofdstuk gaan we een begin maken met het maken van een calculator. Hierbij stellen we de eis, dat er enkel getallen ingevoerd kunnen worden. Dit kun je instellen met data types. We beginnen we daarom het doornemen van de data types.

## Leerdoelen:

1. Data Type begrijpen
2. Cijferinvoer
3. Maken van invoervelden
4. Maken calculator

## Theorie

### Data Type

Er zijn veel verschillende soorten data types.

Nederlandse benaming	Engelse benaming	Type
Tekst type	Text Type:	str
Numeriek type	Numeric Types:	int, float, complex
Reeks types	Sequence Types:	list, tuple, range
Map types	Mapping Type:	dict
Vast gestelde types	Set Types:	set, frozenset
Boolean type	Boolean Type:	bool
Binair / tweeledige types	Binary Types:	bytes, bytearray, memoryview

Voorbeeld	Data type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apel", "banaan", "kers"]	list
x = ("mango", "wortel", "peer")	tuple
x = range(6)	range
x = {"name" : "Peter", "age" : 33}	dict
x = {"komkommer", "citroen", "tomaat"}	set
x = frozenset({"appel", "banaan", "peer"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

## Int() function

De int() functie verandert nummers in gehele getallen.

Voorbeeld:

```
x = int(3.5)
```

The screenshot shows a dark-themed instance of Visual Studio Code. At the top, there's a tab bar with the file name "int voorbeeld.py". The main editor area contains the following Python code:

```
1 x = int([4.6])
2
3 print(x)
```

Below the editor, the bottom navigation bar has tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is currently selected, indicated by a solid underline. To the right of the tabs is a dropdown menu set to "1: Python". Further right are icons for creating a new terminal ("+"), closing the terminal ("X"), and other terminal-related functions.

The terminal window below displays the output of the Python code execution:

```
""/Users/jadethomson-oddung/Desktop/python/int
voorbeeld.py"
4
jade-mbp:~ jadethomson-oddung$
```

Syntax: int(*value*, *base*)

*Parameter waarden:*

Parameter	Omschrijving
<i>value</i>	Een nummer of string die naar een geheel nummer wordt veranderd
<i>base</i>	Een nummer dat het format bepaalt: Default value: 10

## Str()

`Str()` converteert specifieke waarden in een string.

### Voorbeeld:

```
x = str("3.5")  
print(x)
```

The screenshot shows the Visual Studio Code interface. On the left is the sidebar with icons for file operations, run and debug, and terminal. The main area has two tabs open: 'str.py' and 'Untitled-1'. The 'str.py' tab contains the following code:

```

1 x = str(3.5)
2
3 print(x)

```

The terminal at the bottom shows the output of running the script:

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\jdt.thomson\OneDrive - Noorderpoort\2020-2021\AB\Code>
3.5

```

Syntax: `str(object, encoding=encoding, errors=errors)`

Parameterwaarden:

Parameter	Omschrijving
<code>object</code>	Elk object. Geeft aan dat dit object in een string wordt veranderd
<code>encoding</code>	De codering van het object. Default is UTF-8
<code>errors</code>	Specificeert wat de code doet als het omzetten verkeerd gaat

## Float

De float() functie converteert de specifieke waarde in een decimaal getal.

The screenshot shows a code editor interface with several tabs at the top: 'les 3 Loops - race.py', 'a = 12 Untitled-1', 'bul.py', 'float.py' (which is the active tab), and another unnamed tab. Below the tabs, a terminal window displays the following command and output:

```
C: > Users > jdt.thomson > OneDrive - Noorderpoort > 2020-2021 > AB > Code > float.py
1 # for integers
2 print(float(10))

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\jdt.thomson\OneDrive - Noorderpoort\2020-2021\AB\Code>
10.0
```

Syntax: `float(value)`

Parameterwaarden:

Parameter	Omschrijving
<code>value</code>	Een nummer of een string die wordt veranderd in een decimaal getal

## Bool

De `bool()` functie geeft de boolean (True/False) waarde van een specifiek object

Voorbeeld:

```
print(5>4)
print(5==4)
print(5<4)
```

The screenshot shows a code editor interface with three tabs at the top: 'les 3 Loops - race.py', 'a = 12 Untitled-1', and 'bul.py'. The 'bul.py' tab contains the following Python code:

```
C: > Users > jdt.thomson > OneDrive - Noorderpoort > 2020-2021 > AB > Code > bul.py
1 print(5 > 4)
2 print(5 == 4)
3 print([5] < 4)
```

Below the tabs is a navigation bar with 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected and displays the following output from a Windows PowerShell:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\jdt.thomson\OneDrive - Noorderpoort\2020-2021\AB\Code>
[REDACTED]
True
False
False
```

De boolean geeft altijd True terug. Er zijn uitzonderingen:

- Als het object leeg is [], (), {}
- Het object is False
- Het object is 0
- Het object is None

Syntax: `bool(object)`

Parameterwaarden:

Parameter	Omschrijving
<i>object</i>	Elk object, zoals String, List, Number etc.

## Praktijk

Het eerste wat je nodig hebt voor een rekenmachine is een inputregel om je cijfers in te kunnen voeren waar mee je wilt rekenen.

hier voor wil je de functie input gebruiken een voorbeeld hier van is:

```
number_1 = input('Typ je eerste nummer hier: ') # een variable waar een nummer ingevoerd ka
```

We hebben number\_1 de waarde input gegeven, dit betekent ook dat we letters kunnen invoeren. Dit is lastig voor het rekenen. Als je in Python 3 programmeert, zul je het volgende zien in je berekening:

The screenshot shows a code editor interface with a dark theme. At the top, there's a file tab labeled 'rekenmachine.py ×'. Below it is a code editor window containing the following Python code:

```
Users > jadethomson-oddin > Desktop > python > rekenmachine.py > ...
1  number_1 = input('Enter your first number: ')
2  number_2 = input('Enter your second number: ')
3  print(number_1, "+", number_2, "=", (number_1 + number_2))
```

Below the code editor is a terminal window titled '1: Python Debug Console'. It displays the following interaction:

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: Python Debug Console + ^ X
Enter your first number: 1
Enter your second number: 2
1 + 2 = 12
bash-3.2$
```

Daarom stellen we in dat een int moet zijn.

```
number_1 = int(input('Typ je eerste nummer hier: ')) # een variable die enkel gehele cijfer
```

Dit wil je ook doen voor nummer twee.

Nu willen we natuurlijk zien dat er iets is geprint, om de uitkomst van number\_1 + number\_2 te krijgen print je de twee variabelen uit tussen de haakjes.

```
print(number_1 + number_2)
```

Als je ook de berekening wil uitprinten, dan doe je dat als volgt:

```
print(number_1, "+", number_2)
```

Om het mooier te laten lijken, kun je ze ook samen op een regel printen. Dit doe je als volgt:

```
print(number_1, "+", number_2, "=", number_1 + number_2)
```

De regel print een opsomming met 5 waardes:

- de variabele number\_1 (in het voorbeeld hier onder is dat 1)
  - een string met de waarde "+"
  - hierna weer een variabele number\_2
  - een string "="
  - een expressie met twee variabelen en de operator +

The screenshot shows a code editor window with the following details:

- Title Bar:** The file name is "rekenmachine.py".
- Code Area:** The code is as follows:

```
1 number_1 = int(input('Enter your first number: '))
2 number_2 = int(input('Enter your second number: '))
3 print(number_1, "+", number_2, "=", number_1 + number_2)
```
- Bottom Navigation:** Buttons for PROBLEMS, OUTPUT, TERMINAL (underlined), DEBUG CONSOLE, and a Python Debug Console tab.
- Terminal Output:** The terminal shows the execution of the script:

```
Enter your first number: 1
Enter your second number: 2
1 + 2 = 3
bash-3.2$
```

De volgende stap is om te kunnen kiezen of je plus +; min -; keer \*; gedeeld door /, wilt

gebruiken om te berekenen.

De gebruiker mag kiezen tussen deze mogelijkheden en de ingetypte keuze wordt opgeslagen in de variabele berekening.

```
berekening = input('' Kies de berekenings methode: + - * / '')
```

De volgende stap is om een if te gebruiken: als berekening + is print

```
print(number_1, "+", number_2, "=", number_1 + number_2)
```

```
if berekening == '+':  
    print(number 1, "+", number 2, "=", number 1 + number 2)
```

Dit komt er als volgt uit te zien:

The screenshot shows a code editor with a dark theme. The file 'rekenmachine.py' is open, displaying the following code:

```
1 number_1 = int(input('Enter your first number: '))
2 number_2 = int(input('Enter your second number: '))
3 print(number_1, "+", number_2, "=", number_1 + number_2)
4
5 berekening = input('''
6 Kies de berekenings methode:
7 +
8 -
9 *
10 /
11 ''')
12
13 if berekening == "+":
14     print(number_1, "+", number_2, "=", number_1 + number_2)
```

Below the code editor is a terminal window showing the execution of the script:

```
Enter your first number: 1
Enter your second number: 2
1 + 2 = 3

Kies de berekenings methode:
+
-
*
/
+
1 + 2 = 3
```

*If* mag maar 1 keer in een loop gebruikt worden, dus de volgende stap moet met een *elif*.

Als alle mogelijkheden zijn geweest plaats je een else waarbij geprint wordt "u heeft geen geldige keuze gemaakt, begin opnieuw"

## Opdracht: Maak de calculator af

zorg dat het volgende er in zit:

- +
- -
- \*
- /
- "u heeft geen geldige keuze gemaakt, begin opnieuw"

Gefeliciteerd, je hebt een werkende calculator gemaakt.

je hebt gewerkt met:

- if statements
- het elif statement
- je hebt invoer velden gemaakt
- je hebt met variabelen gewerkt

Als je een uitgebreidere calculator wilt maken die visueel is kun je nog kijken naar de volgende links:

<https://www.youtube.com/watch?v=WRRigB8nMU0>

<https://medium.com/@adeyinkaadegbenro/project-build-a-python-gui-calculator-fc92bddb744d>

# Over dit lesmateriaal

## Colofon

Auteurs	Maaike van Kessel ; J.D.T. Thomson ; Maaike van Kessel
Team	Wikiwijs Maken Auteurs
Laatst gewijzigd	25 maart 2021 om 09:53
Licentie	De Internationale Creative Commons 4.0 licentie waarbij de gebruiker het werk mag kopiëren, verspreiden en doorgeven en afgeleide werken mag maken onder de voorwaarde: Naamsvermelding, zie <a href="http://creativecommons.org/licenses/by/4.0/">http://creativecommons.org/licenses/by/4.0/</a> . <a href="#"><u>Meer informatie over de CC Naamsvermelding 4.0 Internationale licentie licentie.</u></a>

## Aanvullende informatie over dit lesmateriaal

Van dit lesmateriaal is de volgende aanvullende informatie beschikbaar:

Eindgebruiker	leerling/student
Studiebelasting	4 uur en 0 minuten