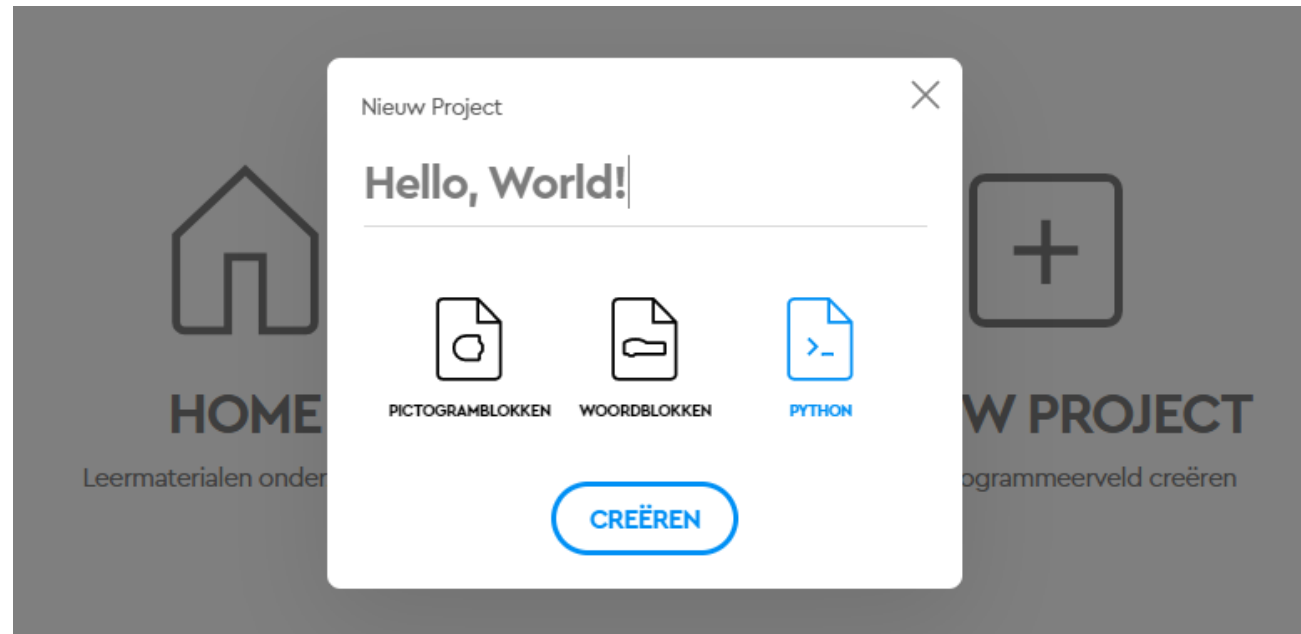


# Workshop

## Programmieren in Python

# Opstarten



Start een nieuw Python-project.  
Geef het project de naam “Hello, World!”


# Hello, World!

- Verwijder de code die in beeld staat en neem het volgende over:

```
from hub import light_matrix
from runloop import run

async def main():
    await light_matrix.write("Hi!")

run(main())
```

- Upload het programma naar je robot door op de  -knop te klikken.
  - Wat gebeurt er?
- Experimenteer en beantwoord de volgende vragen:
  1. Hoe kun je de robot iets anders laten zeggen?
  2. Maakt het uit in welke volgorde je de regels van het programma zet?
  3. Wat is de betekenis van een # aan het begin van een regel?

# Tellen met Spike (1)

- Maak een nieuw project aan. Geef het nu de naam “Tellen”.
- Neem onderstaand programma over voer het uit op de robot.

```
from hub import light_matrix
from runloop import sleep_ms, run

async def main():
    light_matrix.write("1")
    await sleep_ms(1000)
    light_matrix.write("2")

run(main())
```

1. Kijk goed naar het scherm van de robot! Wat doet dit programma?
2. Hoe kun je de robot sneller of langzamer laten tellen?
3. Verander het programma zodat de robot tot 5 telt.

# Tellen met Spike (2)

- Neem onderstaand programma over.
  - Dit programma werkt nog niet! Er staat nog een vraagteken...

Belangrijk: gebruik hier de TAB toets om de code te laten inspringen.

```
from hub import light_matrix
from runloop import sleep_ms, run

async def main():
    for i in range(1, ?):
        light_matrix.write(str(i))
        await sleep_ms(1000)

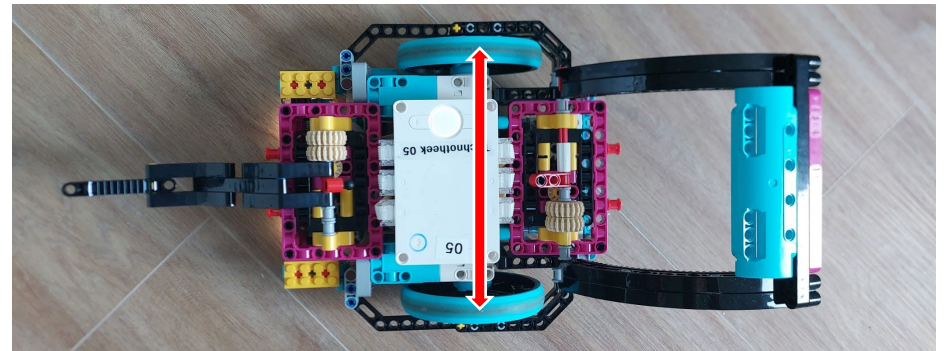
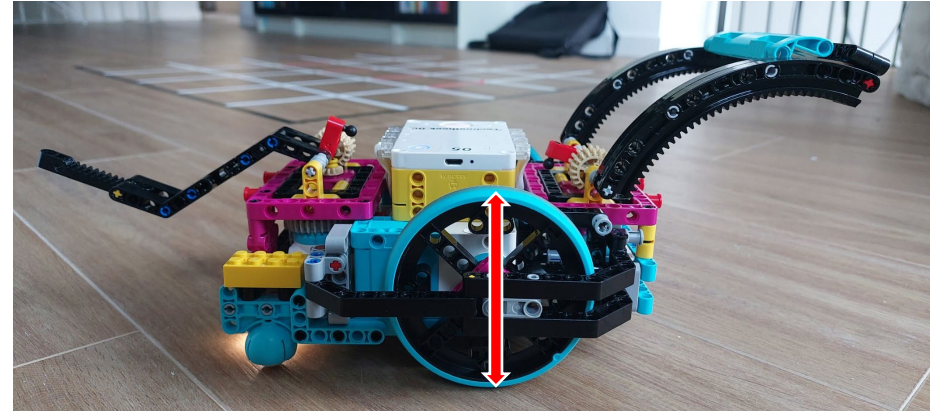
run(main())
```

Vergeet de dubbele punt niet!

1. Experimenteer: welk getal moet er op de plaats van het vraagteken komen te staan om Spike **tot en met 5** te laten tellen?
2. Verander het programma zodat ...
  - a. ... Spike van **1 tot en met 9** telt.
  - b. ... Spike van **4 tot en met 8** telt.
  - c. ... Spike twee keer zo snel telt.

# Rijden (1)

- Benodigdheden:
  - Schuifmaat/geodriehoek
  - Meetlint
  - Pen en papier
- Meet en noteer (in centimeter):
  1. De diameter van de wielen van de robot.
  2. De wielbasis van de robot.
    - De wielbasis is de afstand tussen de wielen, gemeten van midden tot midden.



## Rijden (2)

- Sluit je oude project en maak een nieuw project aan.
- De code die je nodig hebt om de robot te kunnen laten rijden, kun je downloaden via de volgende link:

<https://tinyurl.com/3dafxr3j>

- Kopieer de code uit de link naar je project.
- Schrik niet! De code die er staat doet de ingewikkelde berekeningen zodat je die niet meer zelf hoeft te doen ;-)

# Rijden (3)

- Eerst vertellen we de robot met welke poorten zijn motoren zijn verbonden, hoe groot zijn wielen zijn en hoe breed zijn wielbasis is. Deze gegevens heeft de robot nodig om te kunnen berekenen hoe ver en hoe snel zijn wielen moeten draaien.
- Zoek deze regel op en verander de vraagtekens naar de juiste letters/getallen:

```
robot = Robot(motor_links = port.?,  
              motor_rechts = port.?,  
              motor_arm_voor = port.?,  
              motor_arm_achter = port.?,  
              wielbasis = ?,  
              wieldiameter = ?)
```

- Vanaf nu kunnen we de robot commando's geven. Dit doen we in de main-functie:

Zonder “await” werkt  
het niet....

→ `await robot.vooruit(afstand = 100, snelheid = 50)`

- Meet hoe ver de robot gereden heeft en schat in hoe lang hij daar ongeveer over deed.
  - Wat betekenen de getallen in dit voorbeeld?
  - Laat de robot verder, minder ver, sneller en langzamer rijden.
  - Kan de robot ook achteruit?



# Rijden (4)

- We gaan nu de robot **fine-tunen**, zodat hij nauwkeuriger rijdt.
- Geef de robot het commando om 100 cm te rijden en meet daarna hoe ver hij daadwerkelijk gereden heeft. Onthoud dit getal!
- Laat nu de robot 360 graden om zijn as draaien en schat in hoe ver hij daadwerkelijk gedraaid heeft.
- Verander de volgende regels door je metingen in te vullen op de plek van de vraagtekens:

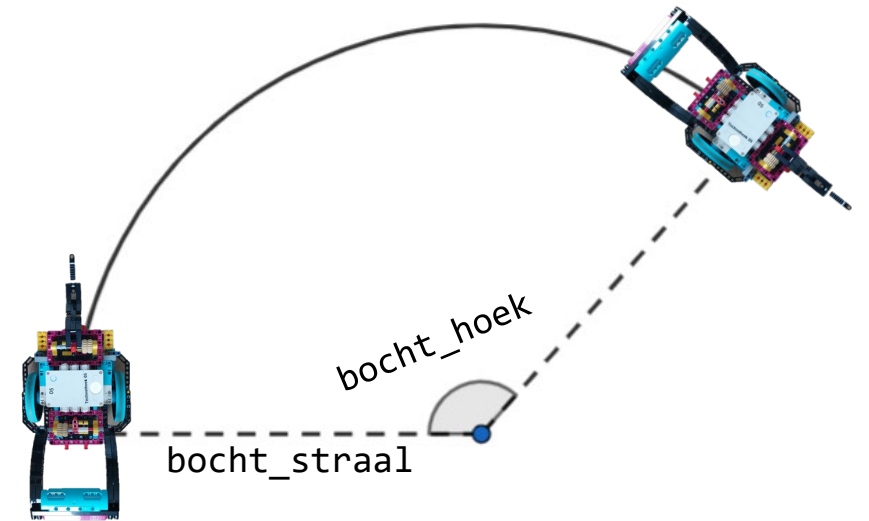
```
robot.corrigeer_wieldiameter(afstand_ingesteld = 100, afstand_gereden = ?)  
robot.corrigeer_wielbasis(hoek_ingesteld = 360, hoek_gedraaid = ?)
```

- Test nu of de robot nauwkeuriger geworden is!

# Rijden (5)

- Hieronder zie je de andere commando's die de robot uit kan voeren:

```
vooruit(afstand, snelheid)  
achteruit(afstand, snelheid)  
bocht_links(bocht_straal, bocht_hoek, snelheid)  
bocht_rechts(bocht_straal, bocht_hoek, snelheid)  
draai_links(hoek, snelheid)  
draai_rechts(hoek, snelheid)
```



- Bijvoorbeeld, om de robot 180 graden linksom te laten draaien:

```
await robot.draai_links(hoek = 180, snelheid = 20)
```

Vergeet "await" niet!

### Spiekbriefje:

```
vooruit(afstand, snelheid)
achteruit(afstand, snelheid)
bocht_links(bocht_straal, bocht_hoek, snelheid)
bocht_rechts(bocht_straal, bocht_hoek, snelheid)
draai_links(hoek, snelheid)
draai_rechts(hoek, snelheid)
```

## Rijden (6)

- Schrijf een programma die de robot het volgende laat doen:
  - Rijd eerst 1,50m recht vooruit.
  - Draai 180 graden.
  - Rijd weer 1,50m terug.
  - Draai weer 180 graden.
- Staat de robot weer op dezelfde plek als in het begin?

# Challenges

---

1. Laat de robot in een cirkel van 50 cm in diameter rijden.
2. Gebruik een for-loop om de robot 10x heen en weer te laten rijden.
3. Laat de robot in een vierkant rijden.
4. Gebruik een for-loop om de robot in een vierkant te laten rijden.
5. Laat de robot dansen!



# Armen Gebruiken (1)

- Voordat we de armen kunnen gebruiken, moeten we de robot vertellen bij welke motorstanden de armen omhoog/naar beneden staan.
- We maken eerst een testprogramma. Voeg de volgende regel toe aan main (haal alle oude regels eerst weg):

```
await robot.test_arm_voor()
```

- Start het programma en gebruik de pijltjestoetsen op de robot om uit te zoeken bij welk getal de arm helemaal omhoog en naar beneden staat. Noteer deze getallen!



Noteer dit getal!

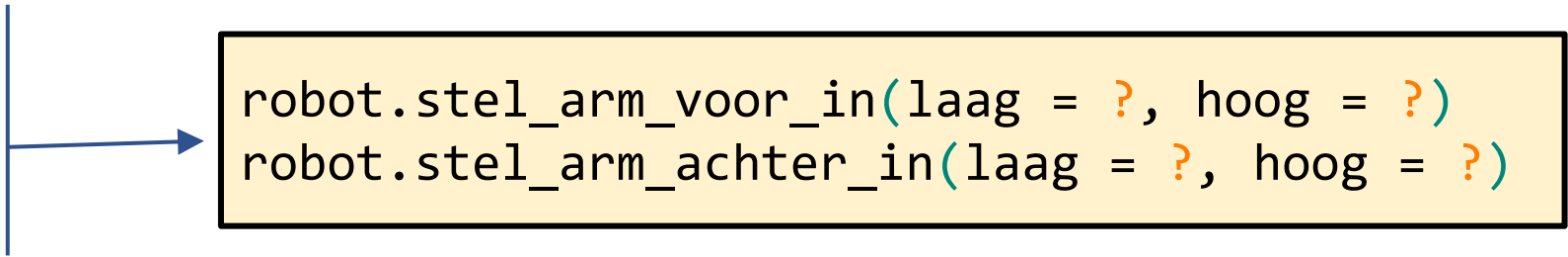
- Doe hetzelfde nog een keer, maar nu voor de achterste arm:

```
await robot.test_arm_achter()
```

# Armen Gebruiken (2)

- Voeg nu deze twee regels toe onder de corrigeer-regels (vlak boven main)
  - Verander de vraagtekens naar de getallen die je net gevonden hebt.

Deze commando's hebben geen "await" nodig omdat de robot niks doet waar we op hoeven te wachten.



```
robot.stel_arm_voor_in(laag = ?, hoog = ?)  
robot.stel_arm_achter_in(laag = ?, hoog = ?)
```

- Je kunt de armen nu besturen met deze commando's:

```
await robot.arm_voor_omhoog(snelheid)  
await robot.arm_voor_omlaag(snelheid)  
await robot.arm_achter_omhoog(snelheid)  
await robot.arm_achter_omlaag(snelheid)
```



# Overzicht & Challenge

## Challenge

Rijd een rondje om de doos en til dan met je arm het plastic bekertje op! Begin bij de startstreep.

```
vooruit(afstand, snelheid)
```

```
achteruit(afstand, snelheid)
```

```
bocht_links(bocht_straal, bocht_hoek, snelheid)
```

```
bocht_rechts(bocht_straal, bocht_hoek, snelheid)
```

```
draai_links(hoek, snelheid)
```

```
draai_rechts(hoek, snelheid)
```

```
arm_voor_omhoog(snelheid)
```

```
arm_vooromlaag(snelheid)
```

```
arm_achter_omhoog(snelheid)
```

```
arm_achtermomlaag(snelheid)
```