



# Python

---

Informatica

Renske Smetsers



# Leerdoelen

---

- Op een creatieve manier problemen leren oplossen
- Leren (goed) te programmeren
  
- Computational thinking (21 eeuwse vaardigheden):
  - Algoritmisch denken
  - Decompositie
  - Generalisatie
  - Abstractie
  - Evaluatie



# Woordje vooraf...

---

Programmeren is moeilijk

- ... het gaat nooit in 1 keer goed (ook bij mij niet)
- ... schets op papier voordat je achter de PC duikt (doe ik ook nog altijd!)
- ... het geeft veel voldoening (je maakt echt zelf iets)
- ... het geeft veel macht (je kan iets wat maar weinig anderen kunnen)
- ... help elkaar, daar leer je zelf ook veel van.
- ... geef niet op!

Gouden regel in de klas:

“Ask 3, then ask me!”

Kom je fouten in de opdracht tegen: meld ze!



# Omgeving

---

- ❑ Opstarten
- ❑ Inloggen
- ❑ Welke knoppen
- ❑ Welke opgaven
- ❑ Eerste programma runnen: Hello world!

- Omgeving met Theorie en Opgaven
  - ‘Complete hoofdstuk’ is een herhaling
- Je moet **alle** opgaven maken, uitvoeren en controleren
  - Zie planner!
  - OPTIONEEL opgaven mag je maken als je wilt
  - AFSLUITENDE OPGAVEN zijn verplicht als handelingsdeel (zie PTA)
- H5/V5 mag als de afsluitende opgaven zonder probleem te maken zijn, de opgaven in de theorie overslaan.



# Opstarten

---

- Volg de stappen op montiplaza > werkdocumenten:



- Elke les:
  - 1) Start Pycharm (desktop)
  - 2) Log in bij [stepik.org](https://stepik.org) & klik door



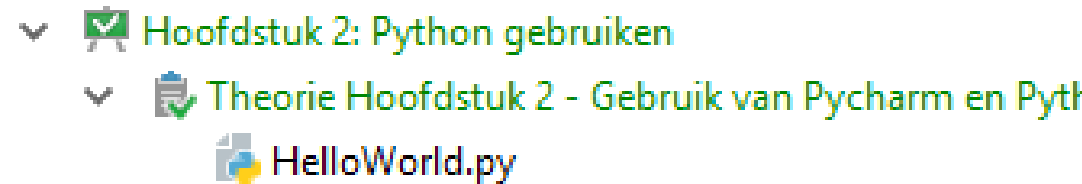
## Werkwijze

---

- Zie je een blauwe slide zoals deze?
- Dat betekent dat we met zijn allen dit samen gaan maken!
- Doe NU mee!

# Eerste programma

1. Ga naar
2. Tik in...



```
print( "Hello, world!" )
```

3. Code runnen:



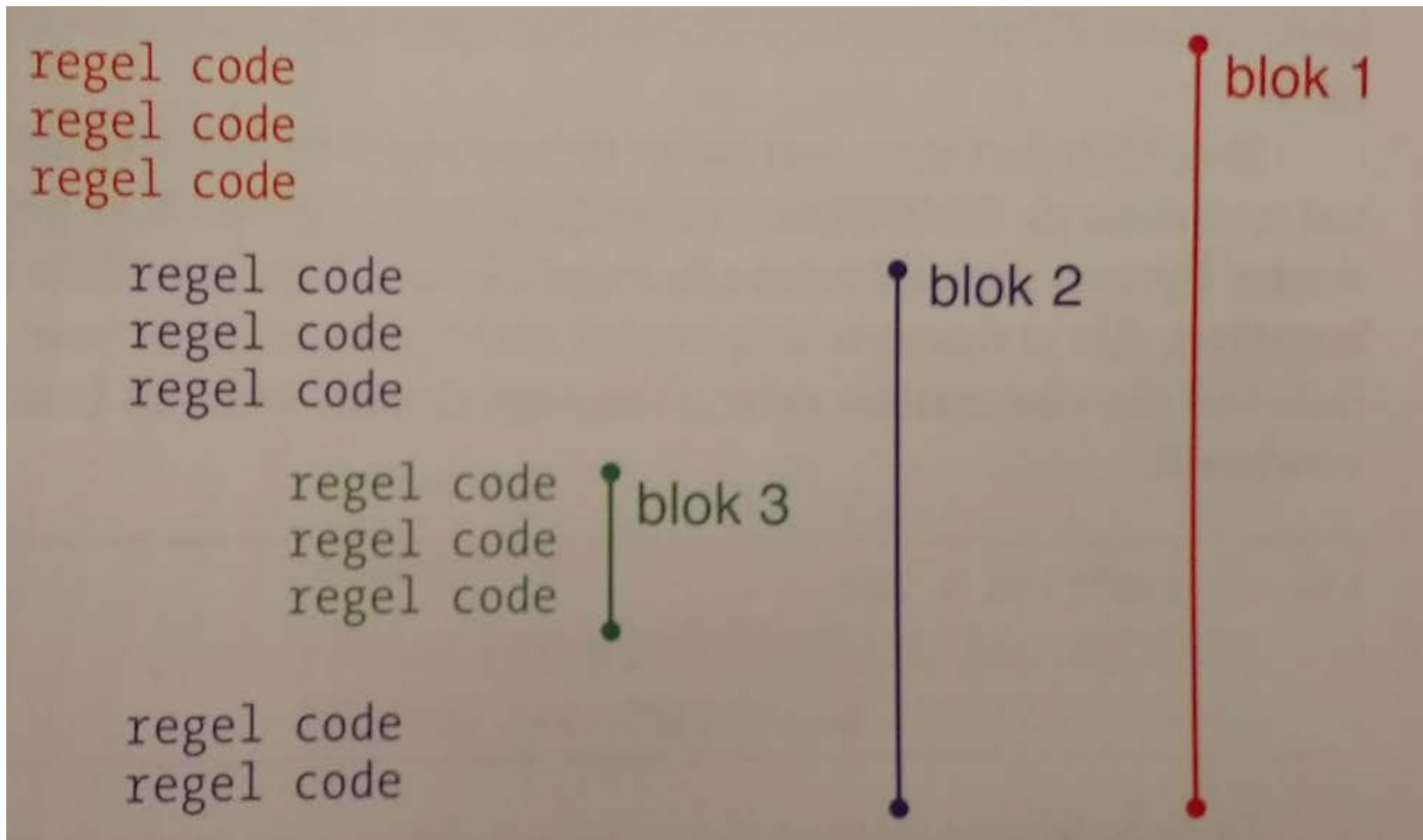
4. Check uitvoer (wat gebeurt er in de console?)
5. Controleer of je de taak goed hebt gemaakt:





# Inspringen

- ▣ Code direct onder elkaar hoort bij elkaar!
- ▣ Spaties/tabs links van de code hebben een betekenis.





# Inspringen

---


- ❑ Inspringen: code dat bij elkaar hoort naar rechts schuiven
- ❑ We leren later hoe dat te gebruiken.
- ❑ Voor nu: alle code moet helemaal naar **links** staan
- ❑ Als je code knipt-plakt kan die wel eens misgaan

**FOUT:**

```
print("De twee belangrijkste dingen te onthouden zijn:")  
    print("Help elkaar, daar leer je zelf ook veel van.")  
    print("... geef niet op!")
```

**GOED:**

```
print("De twee belangrijkste dingen te onthouden zijn:")  
print("Help elkaar, daar leer je zelf ook veel van.")  
print("... geef niet op!")
```



# Commentaar

---

```
# Dit programma drukt een groet af naar het scherm  
  
print("Hallo") # tekst afdrukken
```

Als je # gebruikt, negeert de computer alles wat erachter staat.

**Doel:** uitleggen hoe je code werkt

**Resultaat:** verhoogt de kwaliteit van je code

Maakt je code:

- ❑ leesbaar
- ❑ herbruikbaar
- ❑ onderhoudbaar

**Waarom?** Scheelt je straks tijd bij:

- ❑ debuggen (fouten opsporen)
- ❑ uitbreidingen maken



# Commentaar toevoegen

---

- ❑ Voeg commentaar toe
- ❑ Run je programma
- ❑ Test je programma



# Leerdoelen hoofdstuk 2 (terugblik)

---

- Het gebruik van de **print()** functie om zaken op het scherm te tonen
- Commentaar gebruiken om je code toe te lichten



# Turtle graphics

---

- Met Python kun je ook tekenen!
  - Hiermee oefen je belangrijke concepten voordat we verder gaan met de opdrachten.
- 
- We beginnen met (héél precies) instructies geven.

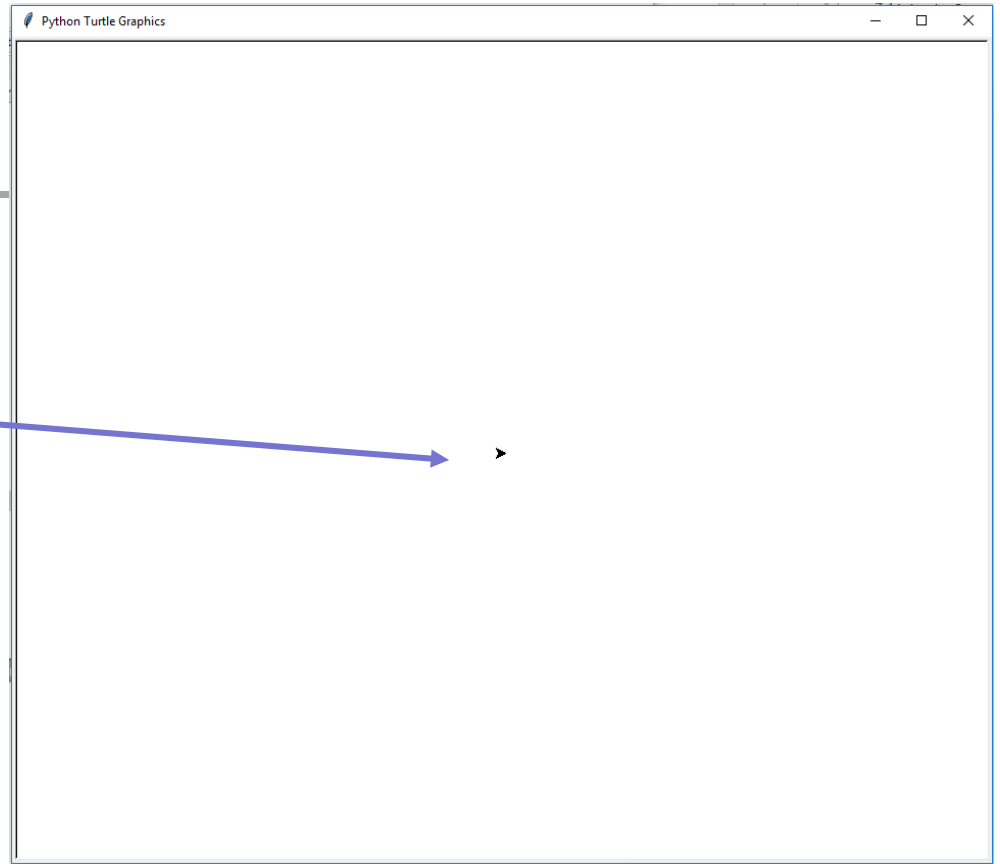


# Turtle's wereld

## Turtle

Begincoördinaat: (0,0)

Kijkrichting: rechts



800

900 px

```
pendown()    # Zet pen op papier
penup()       # Haal pen van papier

right(90)     # Draai 90 graden naar rechts
left(45)      # Draai 45 graden naar links
forward(10)   # Loop 10 stappen vooruit
```



# Turtle programmastructuur

---

```
import turtle          # importeer turtle graphics module

turtle.pendown()        # zet pen neer
turtle.forward(150)     # 150 stappen vooruit
turtle.right(90)        # draai 90 graden naar rechts

turtle.done()          # klaar
```



# Voorbeeld: hexagon

1) Maak een schets

- Lengtes
- Hoeken

2) Bedenk een strategie:

Gebruik een turtle  
Pen neerzetten  
Herhaal 6 keer:  
    Vooruit 100 px  
    Draai  
Klaar

3) Schrijf code & test:

```
import turtle

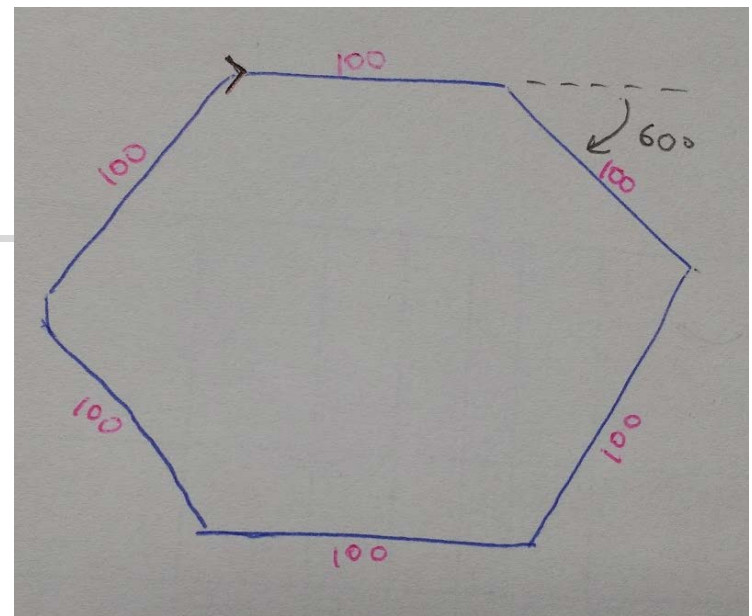
turtle.pendown()

turtle.forward(100)
turtle.right(60)

turtle.forward(100)
turtle.right(60)

...

turtle.done()
```





# Turtle instructies

---

```
pendown()    # Zet pen op papier, alle bewegingen hierna worden zichtbaar
penup()      # Haal pen van papier, bewegingen hierna worden niet zichtbaar
pencolor("red")  # Kleur van de pen.
                # Je kunt ook kiezen uit black, tan, gold, blue...

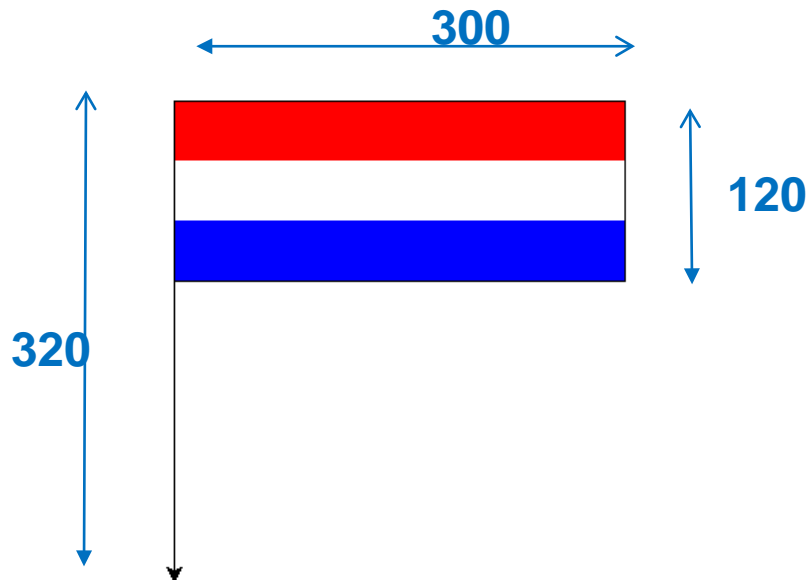
right(90)    # Draai 90 graden naar rechts
left(45)     # Draai 45 graden naar links
forward(10)  # Loop 10 stappen vooruit

setposition(40, 40)    # Zet schildpad op positie (x,y).
                        # (0,0) is midden van het scherm
setheading(90)         # Kijkrichting naar boven

fillcolor()  # Hiermee vertel je met welke kleur je de figuur wilt inkleuren.
begin_fill() # Begin van figuur die ingekleurd moet worden.
end_fill()   # Einde van figuur die ingekleurd moet worden.
```



# Nederlandse vlag tekenen



## Aanpak:

Maak een schets

Deel je problem op in kleinere delen

Pak ieder deelprobleem apart aan

- Schrijf een beetje code
- Test steeds kleine stukjes
- Herhaal totdat je klaar bent