
DE CREATIEVE STEM


Belgium
esero


INGEGNO

Mijn eerste mini-satelliet

Metingen tijdens een vlucht – voor absolute Pico beginners

Benny Malengier - M.Cristina Ciocci

Lesgevers

- B. Malengier. CTSE, MaTCh, FEA, UGent & De Creatieve STEM
  oa Python programmeur.
  github.com/bmcage

- M.Cristina Ciocci, Ingegno & De Creatieve STEM

DE CREATIEVE STEM

Rondetafel

Wat is je programmeerniveau?

1. Ik heb nog nooit geprogrammeerd
2. Ik doe wat formules in Excel, meer niet
3. Ik kan basis programmeren in een IDE (Arduino, C)
4. Ik kan goed programmeren, maar geen python
5. Ik ken python al (redelijk) goed

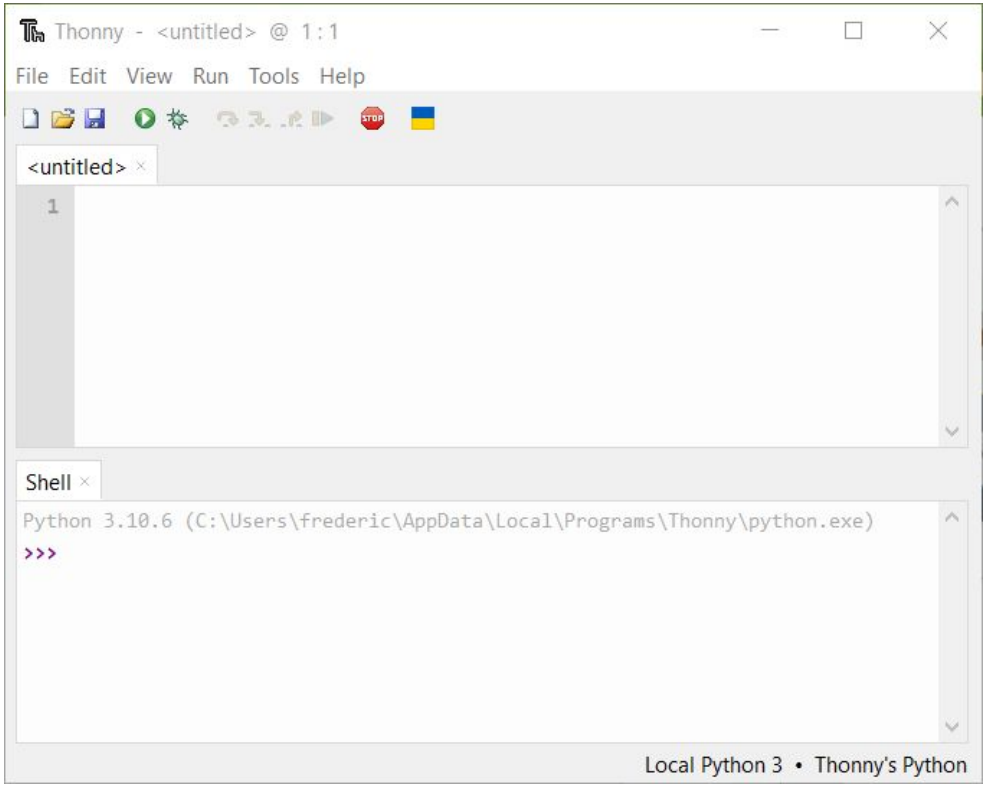
DE CREATIEVE STEM

Voor we beginnen: Installatie Thonny

Installeer Thonny versie 4.1.2 van thonny.org !

Voor linux: gebruik
pip3 install thonny

Voor windows:
Installer with 64-bit Python 3.10
Ben je python programmeur met Anaconda, wijzig in Thonny de pythn interpreter naar die van Anaconda



DE CREATIEVE STEM

Python - Wat is het?

Materials used from:

Python ? Uitspraak.

- Hoe uitspreken?
- https://www.google.com/search?q=python+uitspraak

Wat is Python ?

- Een programmeertaal
- Een manier van leven, …, en programmeren.
- Wat je maar wil dat het is: een rekenmachine, een scripting taal, een high end computing environment, en veel veel meer!

Users vs. Programmers

- **Gebruikers** zien computers als een **set van tools** - word processor, spreadsheet, map, todo list, etc.
- **Programmeurs** leren de computer dingen doen via een programmeertaal
- Programmeurs maken **nieuwe tools**
- Programmeurs schrijven tools voor veel gebruikers, maar soms schrijven ze kleine “hulptools” voor zichzelf om een taak te automatiseren

Wat is Code? Software? Een Programma?

- Een sequentie van opgeslagen instructies
  > een klein deel van **jouw** intelligentie in de computer
  > een klein deel van onze intelligentie die we anderen kunnen geven

  We bedenken iets en dan coderen we het en delen het om anderen de tijd en energie te besparen het uit te zoeken
- Een stukje creatieve kunst - vooral als we veel werk steken in de gebruikerservaring ⇒ je hebt **copyright**

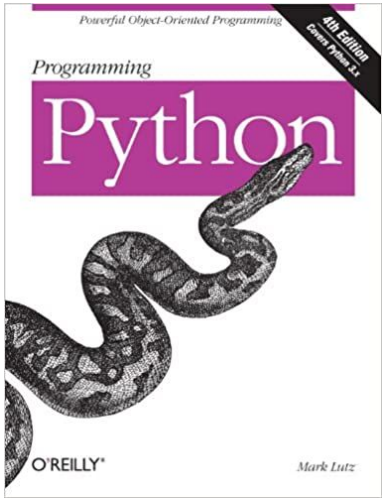
DE CREATIEVE STEM

Waarom werd Python gemaakt? Doelen:

- Een **gemakkelijke en intuïtieve** taal die net zo krachtig is als grote concurrenten
- **Open source**, dus iedereen kan bijdragen aan de ontwikkeling ervan
- Code die zo **begrijpelijk** is als gewone taal
- Geschikt voor alledaagse taken, waardoor **korte ontwikkeltijden** mogelijk zijn

Wat is Python ?

Python is the language of the **Python Interpreter** and those who can converse with it. An individual who can speak **Python** is known as a Pythonista. Nearly all known Pythonistas use software initially developed by Guido van Rossum.



Guido van Rossum was **BDFL** van Python.
Naast het aansturen van Python, werkte hij voor Google en Dropbox aan grote software projecten … geschreven in Python.

Hij trad af in juli 2018, en is nu in pensioen.

DE CREATIEVE STEM

Python Interpreter ?

Ja, een vertaler! Je praat ermee met commando's en het interpreteert je commando's om actie te ondernemen.

Hoe?

- Interactief: Je typt regel voor regel rechtstreeks naar Python en het reageert.
- Script: Je voert een reeks instructies (regels) in een bestand in met behulp van een teksteditor en vertelt Python om de instructies in het bestand uit te voeren. Python-bestanden gebruiken de extensie **.py**

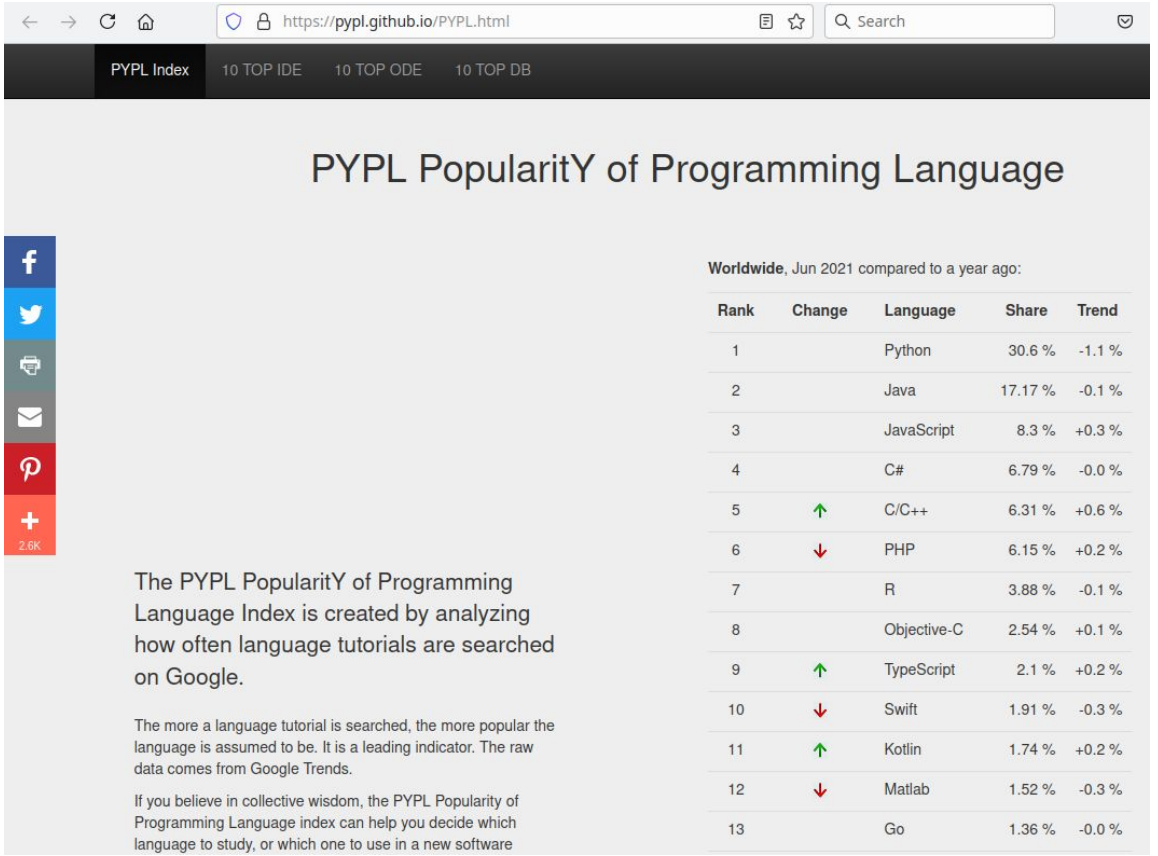
Python is een Scripting Language

- Een programmeertaal die niet is gecompileerd tot bytecode die een computerprocessor begrijpt, maar die in plaats daarvan wordt geïnterpreteerd als opdrachten die eraan worden gegeven, wordt een scripttaal genoemd: Perl, PHP, Javascript, Python
- **Evenwel:** indien gewenst, kunt u met python bytecode maken om de interpretatie te versnellen. Je kunt zelfs compileren als dat nodig is ... maar dat is voor een cursus voor gevorderden

Waarom zou ik Python moeten kunnen?

http://pypl.github.io/PYPL.html

Juni 2021:



DE CREATIEVE STEM

Python - Setting it up

Materials used from:

Welke Python?

- Er zijn verschillende versies van python …
- Versie Python 2.x is niet langer ondersteund vanaf 2021
  ⇒ **GEBRUIK DIE NIET**

- **GEBRUIK versie Python 3.5** of hoger !

- OPGELET!, je wil **Python**, niet Jython (interpreter in Java), cython (compiled C versie), of PyPy (JIT compiler)

Let's start: locale installatie ?

- Je moet de **Python Interpreter** installeren als je python lokaal wil gebruiken
- Beter dan wij kunnen uitleggen:
  https://realpython.com/installing-python/

- **Evenwel**:
  - Makkelijker als je volledige OMGEVINGEN gebruikt, zoals https://www.anaconda.com/
  - Visual Studio Community (Microsoft) https://visualstudio.microsoft.com/vs/features/python/
  - **WIJ:** enkel Thonny, met basis python inbegrepen.

DE CREATIEVE STEM

IDE?

Werk je lokaal, installeer dan een IDE om in te programmeren

- **Spyder**: https://www.spyder-ide.org/
  (deel van Anaconda)

Alternatief:

- Pycharm: https://www.jetbrains.com/pycharm/
- Eric: https://eric-ide.python-projects.org/
- **Wij**: Thonny

Or the big boys like Eclipse, Visual Studio, …, **notepad++**

DE CREATIEVE STEM

De Cloud

Voor basiscursus: **Gebruik python vanuit de browser**

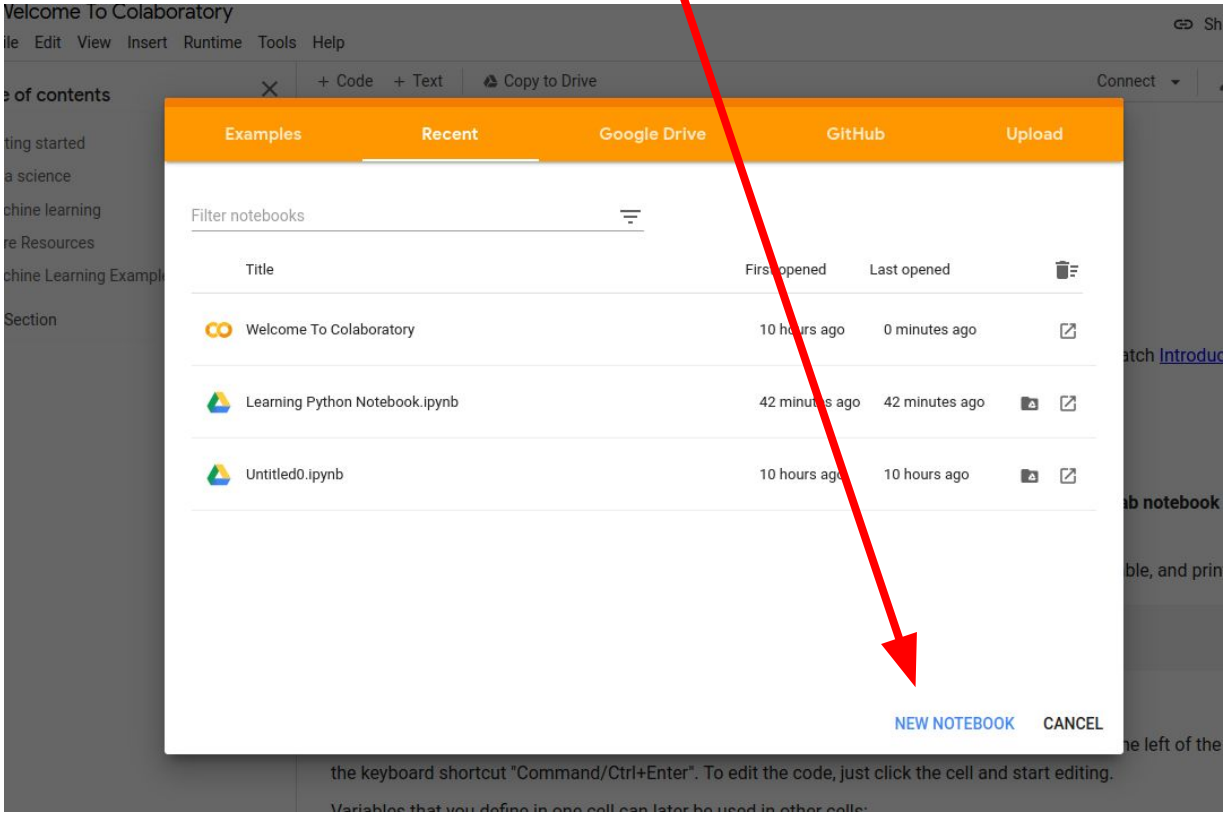
De Interactieve Notebook ! Wordt ondertussen ook voor cursussen gebruikt.
https://colab.research.google.com/

Log in, en start een python notebook!
Alternatief: https://cocalc.com/

DE CREATIEVE STEM

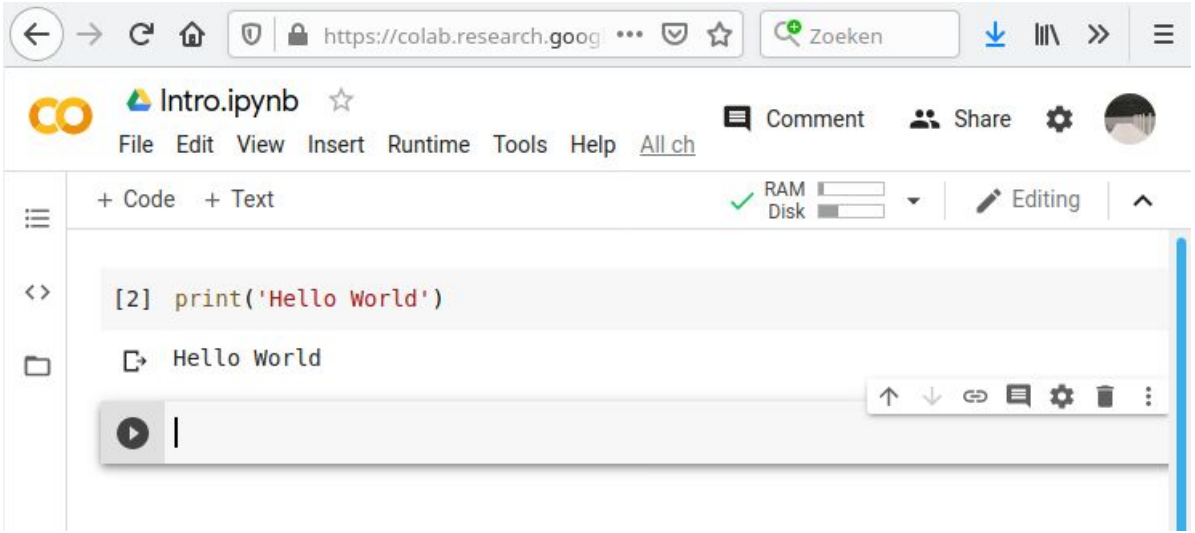
Create Notebook - Hello World

Select bottom right: New Notebook



DE CREATIEVE STEM

1. Geef de notebook een naam, eg Intro
2. Geef tekst: `print('Hallo Wereld')`
3. Press SHIFT+ENTER



JE EERSTE PYTHON CODE !

DE CREATIEVE STEM

Oefening 1

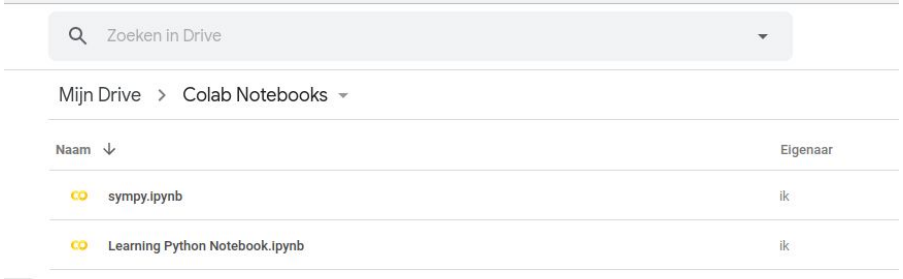
Kopieer volgend voorbeeld in je Notebook, en voer het uit!

```
adjectief = input("Geef een adjectief: ")
naamwoord = input("Voer een zelfstandig naamwoord in meervoud in: ")
werkwoord = input("Voer een werkwoord in verleden tijd in : ")
print("Je Zotte Ingave:")
print("De", adjectief, naamwoord, werkwoord, "over de luie bruine hond.")
```

DE CREATIEVE STEM

Python Notebook

- Orde van uitvoeren telt, niet orde van de cellen!
- *Text* cellen worden niet uitgevoerd
- Een fout in een cel stopt de uitvoering
- Notebooks worden opgeslagen in je Google Drive, in map **Colab Notebooks**



Oefening 2

Pas je Oefening 1 aan

```
adjectief = input("Geef een adjectief: ")
naamwoord = input("Voer een zelfstandig naamwoord in meervoud in: ")
werkwoord = input("Voer een werkwoord in verleden tijd in : ")
print("Je Zotte Ingave:")
print("De", adjectief, naamwoord, werkwoord, "over de luie bruine hond.")
```

Vraag type dier, wijzig dan de output zodat in plaats van ‘hond’ er het gegeven type dier staat.

The Zen of Python

There should be one-- and preferably only one --obvious way to do it.

In je notebook, voer volgende code uit:

```
import this
```

Structuur

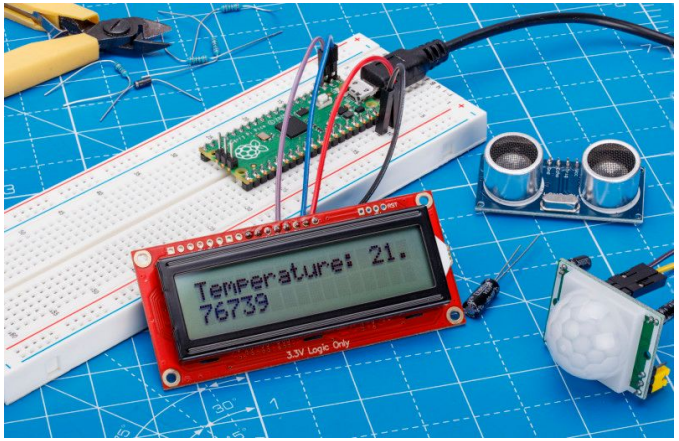
Python is loved by orderly people because **it forces you as much as possible to write orderly, readable, nice code**.

Rossum 1999: "Computer Programming for Everybody"

Gevolg: Sommige mensen HATEN python

Physical Computing en de Raspberry Pi Pico

Physical Computing



- Wat je maakt is tastbaar
- MAAR … je moet wel programmeren om resultaat te bekomen
- Het mengt activatievormen, maar zo krijg je meer leerlingen mee in het verhaal. Je kan werken naar ieders specialisatie en interesse

DE CREATIEVE STEM

Satelliet maken

- We maken een eerste eenvoudig satellietje dat de temperatuur en luchtdruk kan meten. Vervolgens ‘lanceren’ we het met een drone of raket.

- Voor absolute beginners

- Opstap naar CanSat en andere ESA programma’s

DE CREATIEVE STEM

Project	Beschrijving	Wanneer	Teams
ASGARD	Je experiment bereikt met een stratosfeerballon een hoogte van +/- 30 km. Dat is boven de Ozonlaag, de omstandigheden zijn vergelijkbaar met het oppervlak van Mars.	Elk schooljaar (inschrijven voor 11 nov)	●Max 5 leerlingen ●12-20 jaar ●1 leraar (max 2) ●Voor scholen uit alle landen
Bifrost	Je monteert je experiment in een klein vliegtuig, waarin je zelf ook meevliegt. Er worden meerdere parabolen gevlogen waarin je telkens +/- 20 s gewichtloos bent.	Om de twee jaar	●16-20 jaar ●1 leraar ●Voor Belgische scholen
CanSat Belgium	Je lanceert je satelliet ter grootte van een 33 cl blikje met een kleine raket naar een hoogte van 1 à 2 km. Je recupereert de satelliet via een veilig landingssysteem, en houdt radiocontact tijdens de vlucht.	Om de twee jaar	●18 teams (12 lanceringen) ●4 tot 6 leerlingen ●14-20 jaar ●1 leraar ●Voor Belgische scholen
CanSat Europe	De nationale winnaars komen ergens in Europa samen om hun CanSat te lanceren in een Europese competitie.	Einde van elk schooljaar (Belgische deelname om de twee jaar)	●1 team per land ●4 tot 6 leerlingen ●14-20 jaar ●1 leraar

DE CREATIEVE STEM

Het Lesmateriaal

- 1 pico H
- 6 schakeldraadjes
- USB A naar USB micro kabel
- 1 schakelbord
- 1 x 4.7 kOhm weerstand
- 3 x 220 Ohm weerstand
- 1 x DS18B20 Temperatuur sensor
- 1 x BME/P 280 Temp en Baro sensor
- 3 x LED
- 1 x SD kaart 4 GB
- 1 x SD kaart lezer 6 pins

Optioneel: een powerbank om onafhankelijk je opstelling van stroom te kunnen voorzien

DE CREATIEVE STEM

Satelliet onderdelen

- payload ⇒ Wat wij maken. Autonoom metingen kunnen doen

- De bus ⇒ Waar de payload in zit, bv de drone

- Ground Control ⇒ overzicht, uitwisselen data, besturen drone

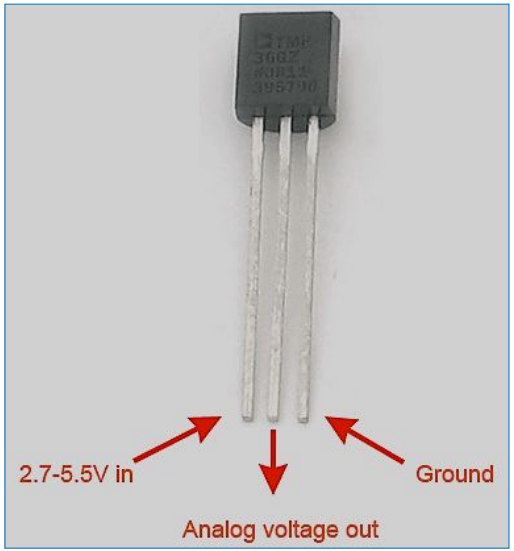
Temperatuur sensoren

NTC thermistor

Het woord thermistor is een samenvoeging van de Engelse woorden ‘thermo’ en ‘resistor’ (weerstand). De NTC is namelijk een elektrische weerstand waarvan de waarde afhankelijk is van de temperatuur.

Temperatuur sensoren

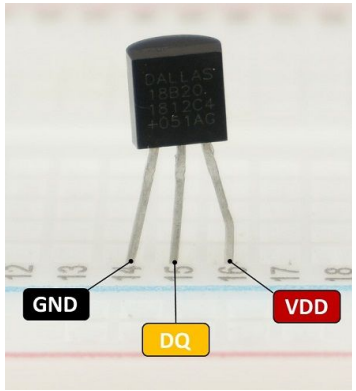
TMP36 Sensor

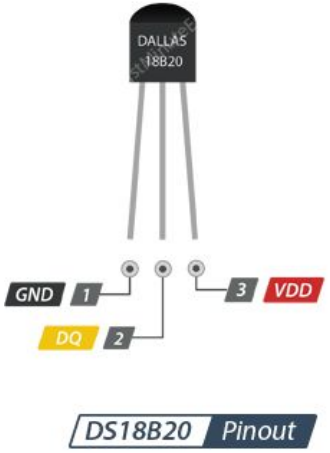


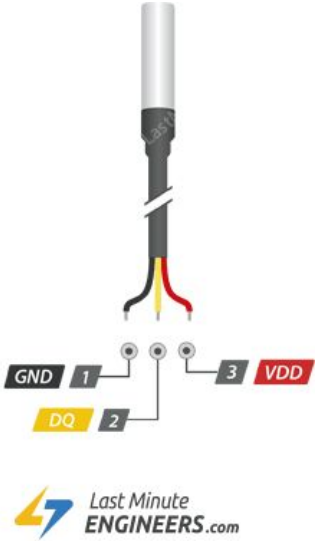
Bevat interne hardware die meting omzet in een analoog uitgangssignaal

Temperatuur sensoren

DS18B20 Sensor







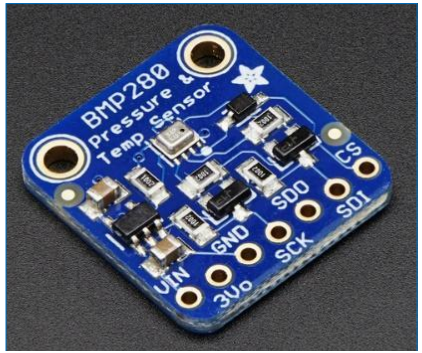
Bevat interne hardware die meting omzet in een digitaal uitgangssignaal.

Raadpleeg het DS18B20 datasheet

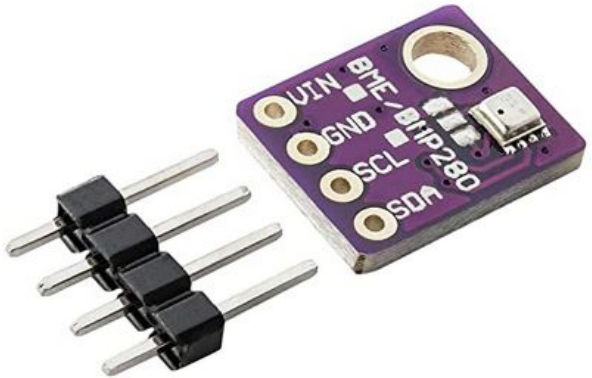
DE CREATIEVE STEM

Druksensor

BMP280. Bestaat in meerdere behuizingen gebouwd rond dezelfde chip.



BME280.

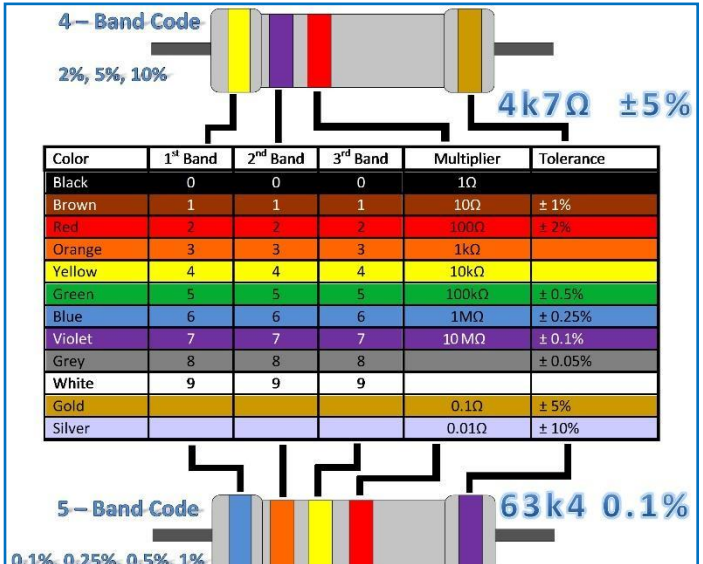


DE CREATIEVE STEM

Weerstand

Een elektrische weerstand beperkt de doorgang van elektrische stroom en veroorzaakt ter plekke een gewenste vermindering van het geleidingsvermogen. Hoe hoger de weerstandswaarde hoe sterker de stroomdoorgang wordt beperkt.

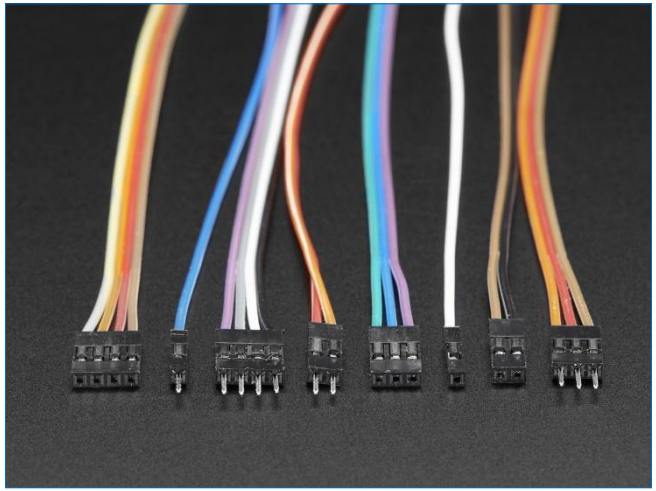
Weerstand R, de eenheid ohm wordt ook wel met het Ω (ohm) teken aangegeven, bijv. R = 512 Ω.



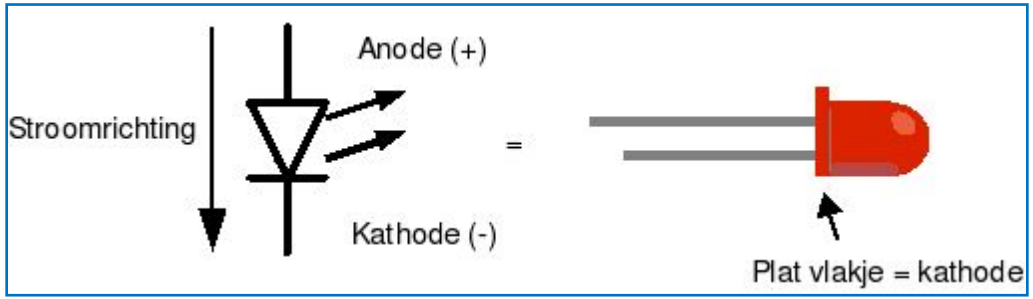
Color	1st Band	2nd Band	3rd Band	Multiplier	Tolerance
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1%
Red	2	2	2	100Ω	± 2%
Orange	3	3	3	1kΩ	
Yellow	4	4	4	10kΩ	
Green	5	5	5	100kΩ	± 0.5%
Blue	6	6	6	1MΩ	± 0.25%
Violet	7	7	7	10 MΩ	± 0.1%
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5%
Silver				0.01Ω	± 10%

DE CREATIEVE STEM

Kabels

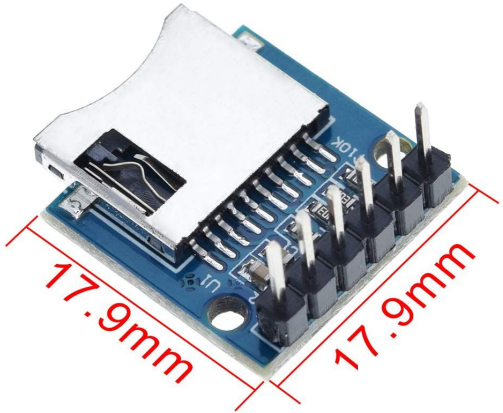


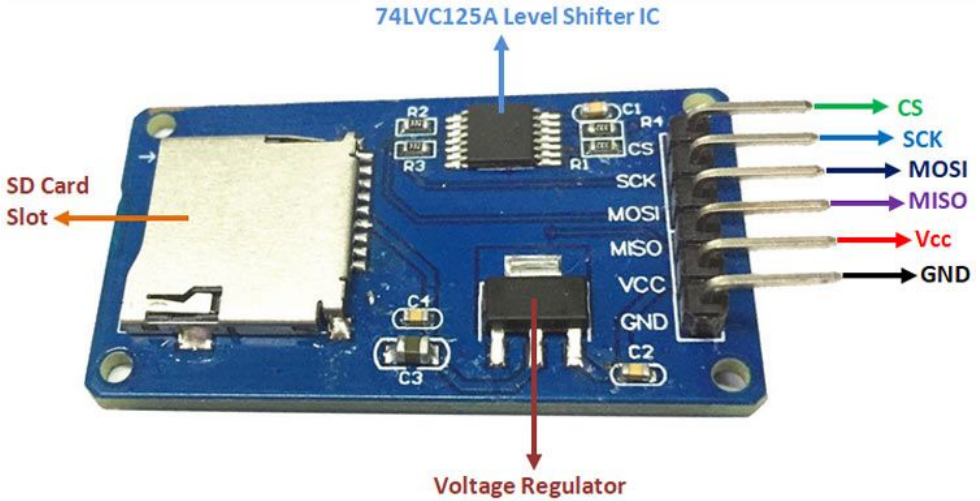
LED - Light Emitting Diode



SD Card Shield

- Data **persistent** opslaan op SD kaart



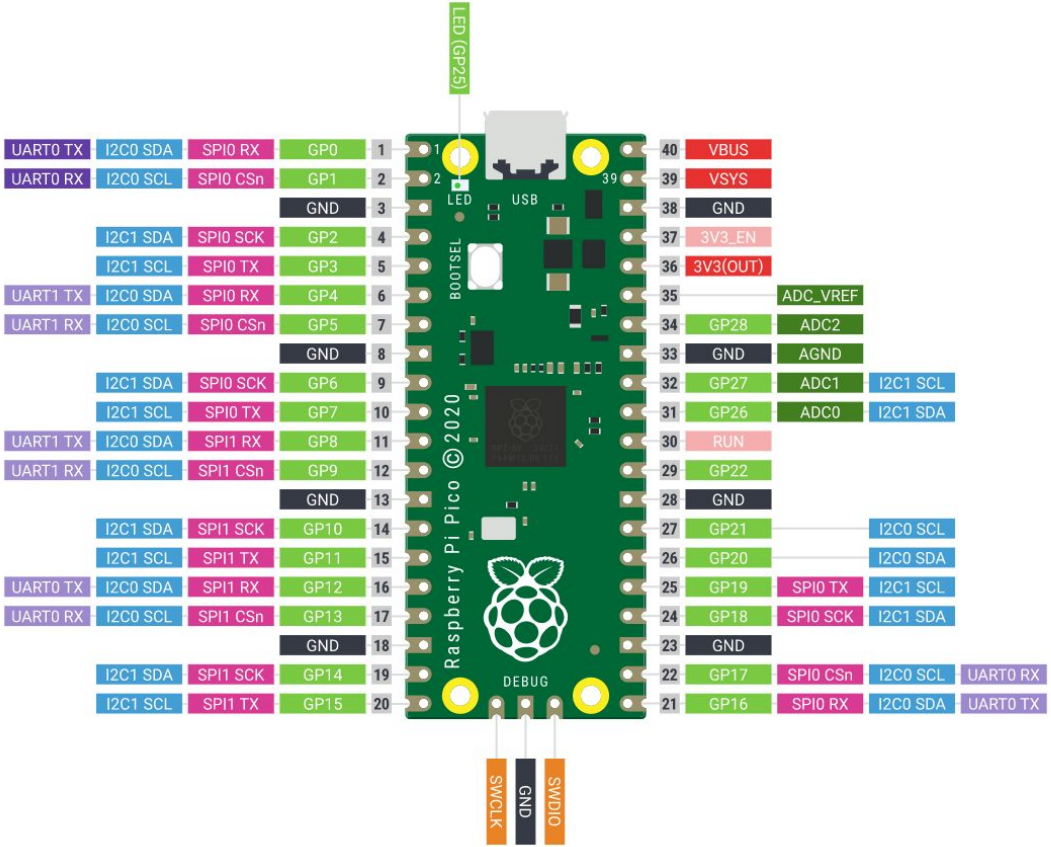


Raspberry Pi Pico

- Een 133MHz processor met 264 kilobytes aan RAM geheugen
- 2 megabytes aan bestandsopslag
- 2 x 20 pinnen: sommige voor stroom, andere voor data

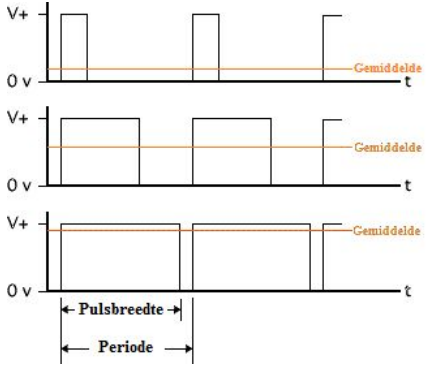
DE CREATIEVE STEM

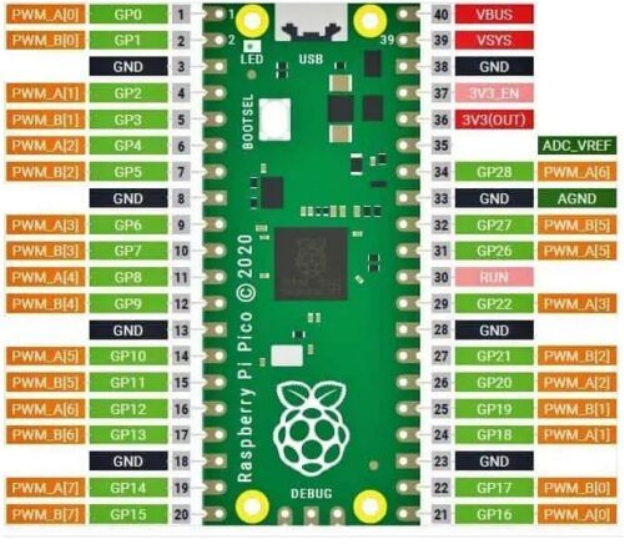
De Pico Pinnen - zie: de interactieve website



De Pico Pinnen

- 3 analoge pinnen. Lees analoge sensoren (max 3V) = ADC pinnen
- 26 digitale pinnen (aan (3V) of uit (GND))
- 16 PWM generatoren. Kun je koppelen aan de pinnen. PWM is Pulsbreedtemodulatie (PBM) in het Nederlands





DE CREATIEVE STEM

De Pico Pinnen

- TX en RX pinnen. Kun je gebruiken voor seriële communicatie.
- GND, 3.3V en 5V pinnen
  - **GND** is de grond. Geen USB kabel? Dan GND batterij aan **pin 38**!
  - **VBUS** is de spanning van de USB. Normaal 5 V vanuit je PC. Wil je pico stroom op de USB laten geven, dan 5 V input op VBUS
  - **VSYS** heeft waarde van VBUS. Je kan + van batterij erop aansluiten met waarde van 3.5 V tot 5.5 V
  - **3V3** om externe componenten te voeden met 3.3 V. Max 300 mA stroom evenwel!

De Pico Pinnen

- Je kan stroom voorzien via de digitale pinnen.
- Dit noemt sourcen of sinken.
- De digitale pin is dus deel van een circuit als stroombron.
- OPGELET: **max 50 mA** aan stroom zo leveren (ipv via GND pin en 5V/3.3V), en **max 12 mA per pin** !

- Pysical Computing = objecten kunnen stuk gaan !

DE CREATIEVE STEM

Opdracht 1

- Neem de Pico en een LED. Connecteer via de micro-USB de pico aan je PC.
- Teken circuit hoe je LED zou doen branden via de Pico.
- Bereken de weerstand die je nodig hebt
- Controleer met de lesgever.
- Maak je circuit

DE CREATIEVE STEM

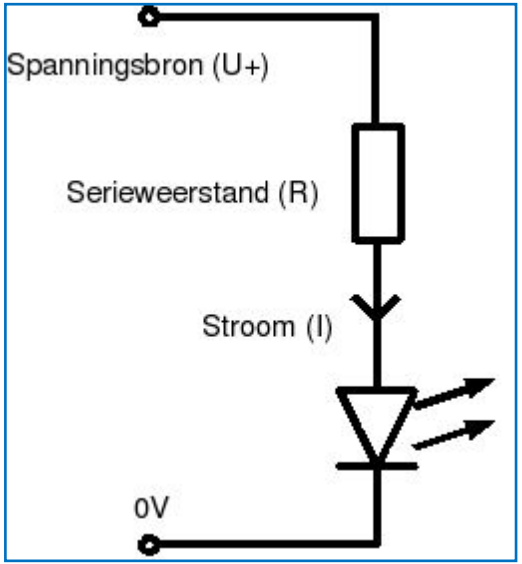
Oplossing

- De Wet van Ohm is van toepassing

  U = I x R of I = U / R of R = U / I

  U is de spanning in volt, I is de stroom in ampère, R is de weerstand in ohm.

- Het circuit is



Oplossing

- Rode LED: 1,9 V nominale spanningsval. Gewenste stroom 12 mA (het maximum toegelaten)
- Over de 3.3V pin of een digitale pin, hebben we 3,3 V, de weerstand die we nodig hebben is:

$$R = (U - U_{led}) / I$$

- Dus: R = (3,3 – 1,9) / 0,012 = 1,4 / 0,012 = 167 ohm
- Opgelet met LED die nominale spanning 3V of meer hebben!

DE CREATIEVE STEM

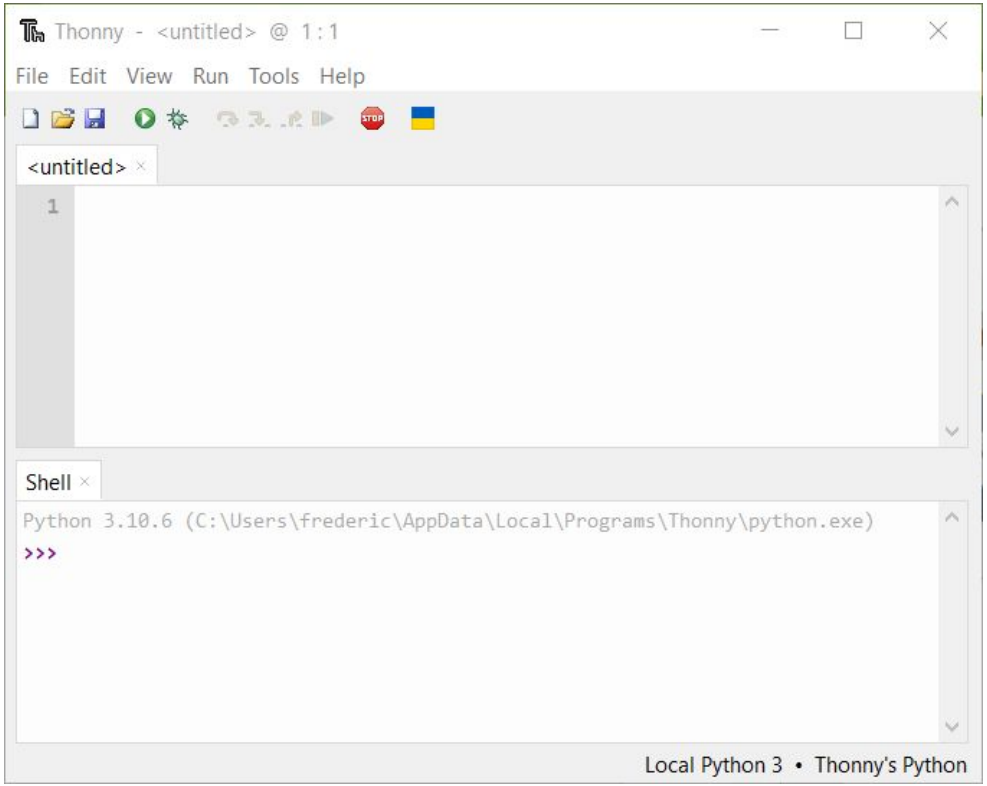
Software: Thonny

Thonny

Installeer Thonny versie 4.1.2 van thonny.org !

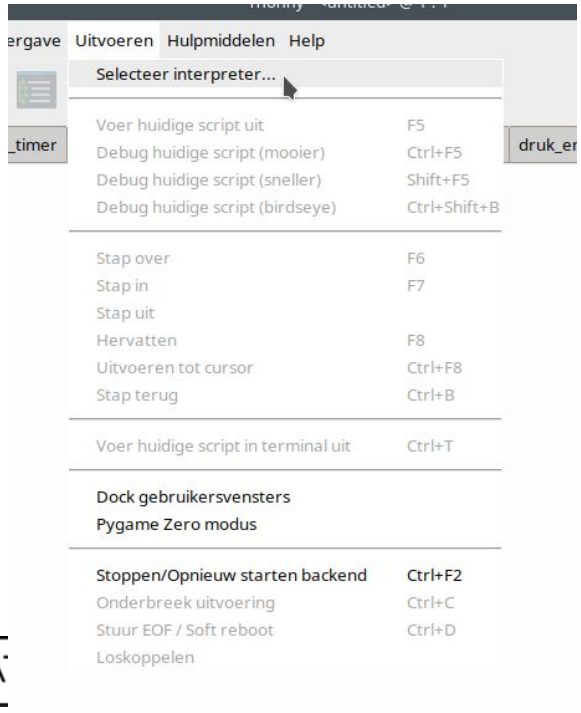
Voor linux: gebruik

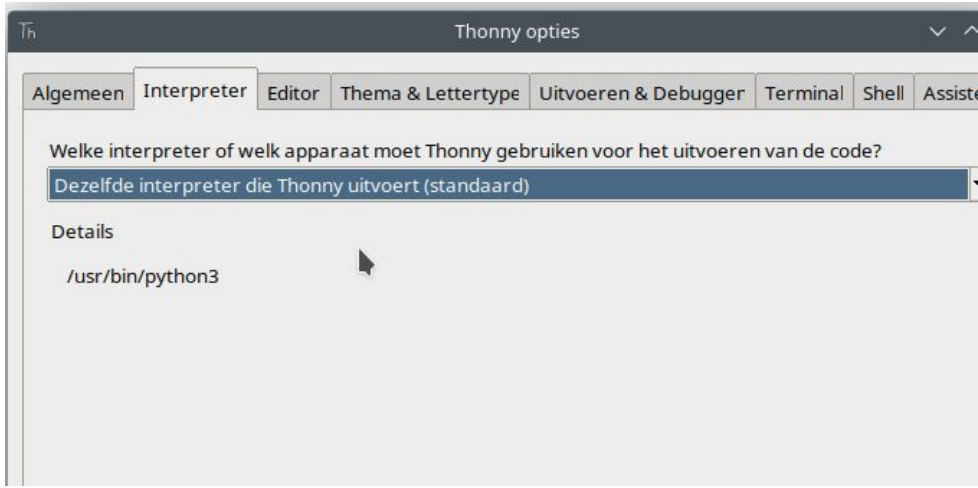
pip3 install thonny



Python interpreter selecteren

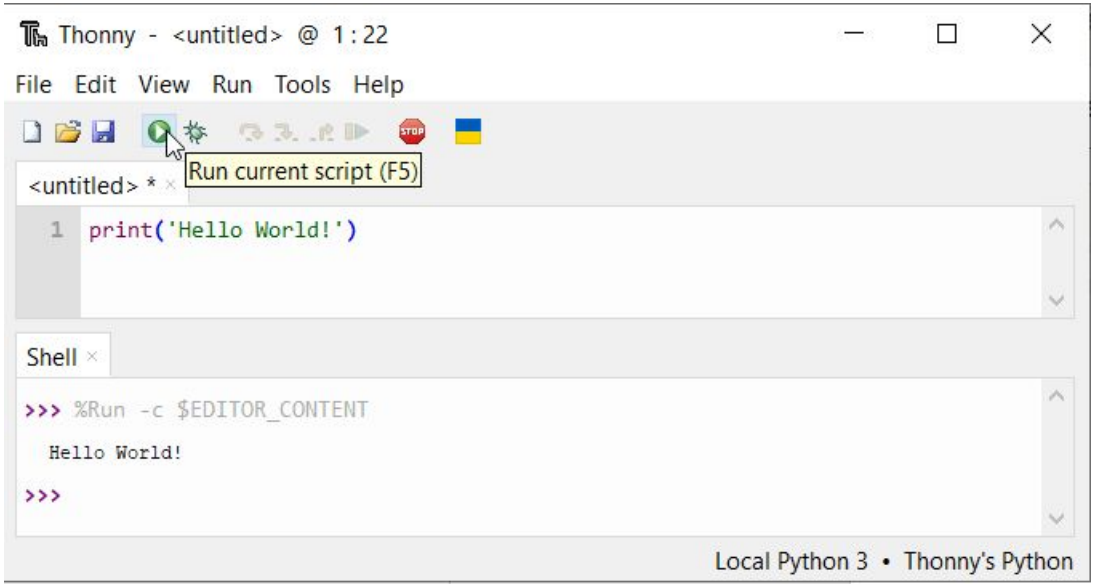
In menu “Uitvoeren”, selecteer Interpreter… ⇒ kies optie “dezelfde die Thonny uitvoert”





Opdracht: Hello World

Voer het hello world programma uit zoals hieronder:



Opdracht: Voeg commentaar toe

Commentaar in code kan via # of over meerdere regels via """

```
# Dit is commentaar
```

of:

```
Code
"""
Dit is commentaar
geschreven over
meerdere lijnen
"""
Nog meer code
```

DE CREATIEVE STEM

Python - Language Syntax

Materials used from:

Oefening: Turtle - import - variable - for loop

In https://trinket.io/ run

```
import turtle
t = turtle.Turtle()
for x in range(100):
  t.forward(x)
  t.left(90)
```

We leggen elke lijn van de code uit …

DE CREATIEVE STEM

Oefening

Oefening:

- Wijzig de hoek waarover je draait.

- Wijzig de afstand die gezet wordt.

Oefening: range

Laat ons kijken hoe range werkt.

Zie: https://docs.python.org/3/library/stdtypes.html#range

Oefening

- Voer in *pynb* de voorbeelden uit de doc van range uit
- In trinket, pas nu je code aan om van je kennis van range gebruik te maken

Wait, what is that *, import, :, for, …. ????

Heb je gelet op de structuur van het programma?

Indentaties worden gebruikt, en niet haakjes zoals { … }

Herinner: orderly, readable, nice code

De python documentatie

De documentatie is je vriend!

Bookmark:

- https://docs.python.org/3/library/index.html
- https://docs.python.org/3/library/functions.html
- https://docs.python.org/3/py-modindex.html
- https://docs.python.org/3/tutorial/index.html

Oefening

Maak een tekstvariabele (string):

```
achternaam = 'Peeters'
```

Zoek nu een manier in de documentatie om dit om te zetten in

1. PEETERS
2. P33t3rs

Merk op, er geen functie om sreteeP te bekomen … of toch? We komen hierop terug later!

DE CREATIEVE STEM

Core Electronics

Python Quick Reference

Variable

```
# Assign an number
x  = 5
# Assign String variable
maker  = “I am a maker ”
# Make a number into a string
str(x)
# Combine strings
newString = maker + “ More string”
```

Strings

```
# Output a string
print("Hello World ")
# output a number
print(x)
# output string and number
print("hello "+str(x)+ " world")
#input
myVariable = input(“message”)
#to input number eval output
myVariable = eval(myVariable)
```

Lists

```
# Make a list
myList = ["item0 ", "item1 ", "item2 "]
# Add item to list
myList.append(“item3”)
# Select first item in a list
myList[0]
# Select last item in the list
myList[-1]
```

Import

```
import time
# Call function from time
time.sleep(1) # Wait for 1 second
# Import single function
from math import sqrt
# Call
sqrt(2)
```

Maths

```
# Addition
1+3
# Exponent
2**3
# Subtraction
1-3
# Modulus
2%3
# Multiplication
1*3
# Integer Division
2//3
# Division
1/3
```

Logic

```
# Return True
True
not False
(True | False) # or
(True & True) #and
# Return False
False
not True
(False | False) # or
(True & False) #and
```

Compare

```
# Return True if comparison holds
x > y    # Greater than
x < y    # Less
x <= y   # Less or equal
x >= y   # Greater or equal
x == y   # Equal
x != y   # Not equal
```

If..else

```
if(x > y):
    print("x is larger ")
elif(x < y):
    print("y is larger ")
else:
    print("Equality! ")
```

Functions

```
# define a function
def threeTimes(xIn):
    return xIn*3
# Call with input=5
threeTimes(5)
# Define Lambda function
t = 6
tMult= lambda a : a*t
# Call with a = 5 t=6
tMult(5)
t = 1
# Call with a = 5 t=1
tMult(5)
```

Loops

```
# While less than
while(x < y):
    x  += 1
    print(x)
# While true with break
while(True):
    x  += 1
print(x)
    if(x>y):
        break

# For loop with default count
for i in range(1,10):
    print(2**i)
# For loop with custom count
for i in range(10,-10,-1):
    print(2**i)
# For loop over list elements
for i in myList:
    print(i)
```

Classes

```
# define Class
class myClass:
  # Initialisation
  def __init__(self,in1,in2):
    self.input1 = in1
    self.input2 = in2
  def add(self): # Class method
    return self.input1+self.input2
# Run the program
myObject = myClass(1,2)
myObject.add ()
```

DE CREATIEVE STEM

forum.core-electronics.com.au

Variabelen en Functies

DE CREATIEVE STEM

Een variabele

Een variabele is een doos om iets in te bewaren. De doos heeft een naam, en een inhoud

```
pennen = 5
kostprijs = 7
totaal = pennen * kostprijs
print (totaal)
```

Er zijn verschillende **operatoren** beschikbaar voor berekeningen

Arithmetic Operators

Operator	Meaning	Example
+	Addition	4 + 7 ⟶ 11
-	Subtraction	12 - 5 ⟶ 7
*	Multiplication	6 * 6 ⟶ 36
/	Division	30 / 5 ⟶ 6
%	Modulus	10 % 4 ⟶ 2
//	Quotient	18 // 5 ⟶ 3
**	Exponent	3 ** 5 ⟶ 243

DE CREATIEVE STEM

Functies

`print(...)` was een functie

Een functie is een stuk code die verschillende commando's bevat die we herhaaldelijk kunnen uitvoeren

```
def berekenTotaal(aantal, prijs):
    totaal = aantal * prijs
    print("Het totaal is", totaal)
```

Oefening: Roep deze verschillende keren op met andere **argumenten**

DE CREATIEVE STEM

Voorbeeld functie

```
def ask_ok(prompt, retries=4, reminder='Dat versta ik niet, probeer opnieuw!'):
    while True:
        ok = input(prompt + ' ')
        if ok.lower() in ('j', 'ja', 'yes'):
            return True
        if ok.lower() in ('n', 'nee', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('Ongeldig gebruikersantwoord')
        print(reminder)

ask_ok("Speel je met mij?")
```

DE CREATIEVE STEM

Nummers, Strings en Lijsten

DE CREATIEVE STEM

Tik volgende code in een cel, en voer uit

```
frieten = 5
cakes = 7
#string variabelen
Achternaam = 'Peeters'
Voornaam = 'Jan'
Adres = 'Zomergem 2'
#Lijst variabelen
Lijst01 = [1, 2, frieten]
Lijst02 = ['Afgevaardigde:',  Achternaam, Voornaam, Adres]
#Afdrukken
print(frieten, cakes)
print(Achternaam, Voornaam, Adres)
print(Lijst01)
print(Lijst02)
```

```
5 7
Peeters Jan Zomergem 2
[1, 2, 5]
['Afgevaardigde:', 'Peeters', 'Jan', 'Zomergem 2']
```

DE CREATIEVE STEM

Python geeft je vrijheid

- Strings gebruiken of ‘ of “
- Lijsten definieer je met [en]

- Werkt volgende lijst?

```
Lijst03 = [ Lijst01, 2, 'test']
```

- Lijsten hebben maar 1 operator: + = concatenatie

```
Lijst04 = Lijst03 + Lijst01
print(Lijst04)
```

DE CREATIEVE STEM

Lijst elementen opvragen

Je kan elementen opvragen. Tellen begint bij **0**

Je kan ook achteraan beginnen met **-1**

```
print(Lijst01[1])
print(Lijst02[2])
print(Lijst04[-2])
```

DE CREATIEVE STEM

Karakter in strings

Je kan een string beschouwen als een LIJST van karakters. Elementen opvragen kan via [..] en concatenatie werkt ook!

```
print(Achternaam[-1])
Voornaam + ' ' + Achternaam
```

Ingebouwde functie operatoren

Python heeft veel ingebouwde functie operatoren die je kan gebruiken op lijsten en string. Test volgende uit

```
Lijst01.append(8)
print(Lijst01)
Lijst01.remove(8)
print(Lijst01)
Lijst01.reverse()
print(Lijst01)
```

Terwijl een string geen reverse functie heeft, kan het wel op lijsten!

DE CREATIEVE STEM

Ingebouwde functie operatoren

Ook voor strings zijn er vele

```
stringVar = "De Koning onder de Berg"
nieuwestring = stringVar.upper()
print(nieuwestring)
print(stringVar.lower())
print(stringVar.split())
```

DE CREATIEVE STEM

Slices … de magie van [ ]

Heel belangrijk zijn **slices**. Dit kan op alle sequentie types, dus bv lijsten en strings. Beschouw een sequentie s:

s[i] : item met volgnummer i van s, de eerste is i=0, de laatste is i=-1

s[i:j] : een *slice* van s van volgnummer i tot j

s[i:j:k] : een *slice* van s van volgnummer i tot j met stap k

Bij een slice mag je index weglaten, dan betekent dat ALLE, bv s[:] = heel de sequentie, en s[::2], heel de sequentie doorlopen met stappen van 2

Oefening

Test slices uit met

```
s1 = 'Peeters'
s2 = list(range(10))
print(s2)
```

Zorg dat je volgende resultaat bekomt:

- ‘ters’
- [0, 2, 4, 6, 8]
- sreteeP

DE CREATIEVE STEM

Ingebouwde functies

DE CREATIEVE STEM

Ingebouwde functies: built-in functions

Meer dan 60. Bijvoorbeeld

```
x = max(60, 65.6, 80, 6, -9)
print(x)
x = round(34.568)
print(x)
x = round(34.499)
print(x)
```

https://docs.python.org/3/library/functions.html

DE CREATIEVE STEM

Modules

DE CREATIEVE STEM

Maak Python nuttiger via Modules

- Modules zijn folders waar gerelateerde functies verzameld zijn

- Er zijn interne modules, deel van de standaard Python versie
  ⇒ Kun je op rekenen per Python versie, bv 3.5.2

- Er zijn modules die anderen schrijven en via internet beschikbaar stellen ⇒ Moet je installeren en je moet weten voor welke versie van de module je software werkt

https://docs.python.org/3.9/library/math.html
https://docs.python.org/3.9/library/random.html

DE CREATIEVE STEM

Modules

Je importeert de **namespace** van de module met:

```
import modulename
```

Je kan alles importeren, maar dat wordt gezien als een **slechte** programmeertechniek

```
from modulename import *
```

Importeer enkel wat je nodig hebt

```
from modulename import functionname, classname, variablename
```

DE CREATIEVE STEM

De Standaard Modules

https://docs.python.org/3.9/py-modindex.html

Let op: **micropython** reduceert wat beschikbaar is om ruimte te sparen op de computerchip! De BBC Microbit en de Pico gebruiken micropython.

Zie: http://docs.micropython.org/en/v1.14/library/index.html

Famous modules

- Scipy https://www.scipy.org/

- https://www.pygame.org/news

- Tensor machine learning https://www.tensorflow.org/api_docs/python/tf

- GUI programs: https://pygobject.readthedocs.io/en/latest/ or https://kivy.org/#home or like Cura 3D print software, Uranium an on top of PyQt: https://github.com/Ultimaker/Uranium https://github.com/Ultimaker/Cura

Enkele voorbeelden

Voer uit:

```
import math
x = math.sqrt(63)
print (x)
print(x*x)
```

Voer uit:

```
import matplotlib.pyplot as plt
plt.plot([1, 3, 5, 7, 9], [1, 9, 25, 49, 81])
plt.show()
```

DE CREATIEVE STEM

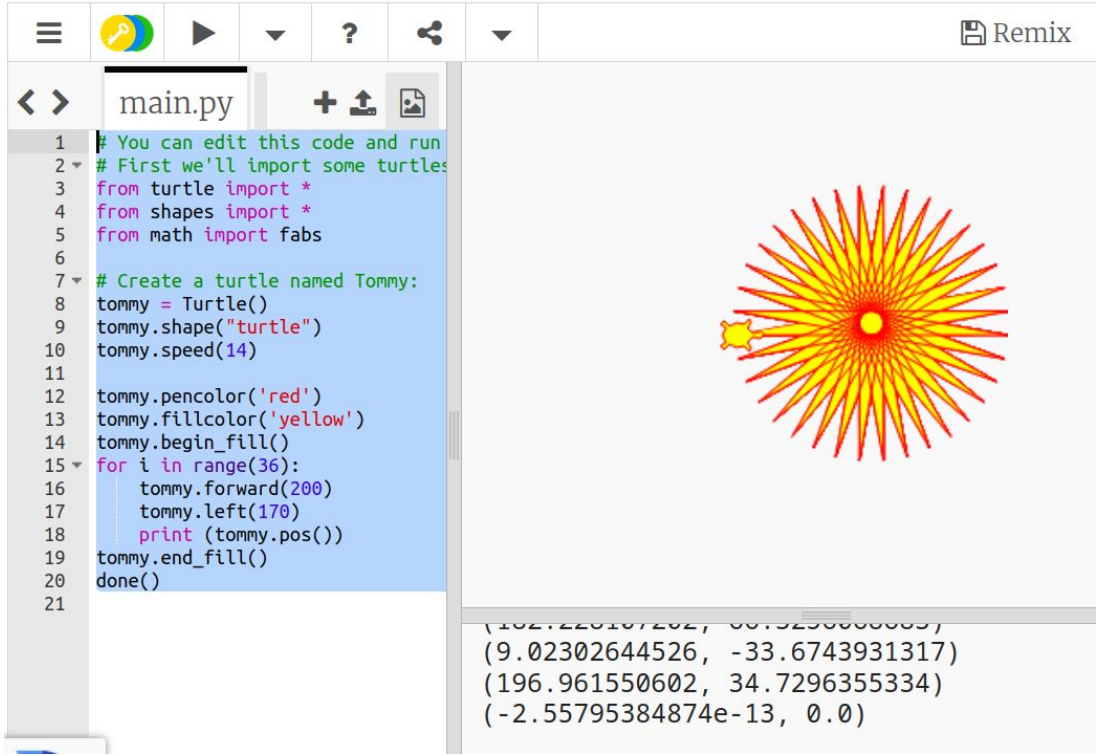
Optioneel - Turtle: een robot controleren

- Via https://trinket.io/

```
from turtle import *
from shapes import *
from math import fabs

# Create a turtle named Tommy:
tommy = Turtle()
tommy.shape("turtle")
tommy.speed(14)

tommy.pencolor('red')
tommy.fillcolor('yellow')
tommy.begin_fill()
for i in range(36):
    tommy.forward(200)
    tommy.left(170)
    print (tommy.pos())
tommy.end_fill()
done()
```



Optioneel - Turtle

Doe

```
left(90)
forward(100)
right(50)
forward(50)
right(90)
backward(50)
```

Oefening

- maak een driehoek
- color(‘red‘) om kleur te wijzigen

  ```
  VALID_COLORS = ('white', 'yellow', 'orange', 'red', 'green', 'blue', 'purple', 'grey', 'black')
  ```
- test getx(), gety(), goto(a,b)
- penup() en pendown() zetten pen op of neer

DE CREATIEVE STEM

Optioneel - Turtle - oefeningen

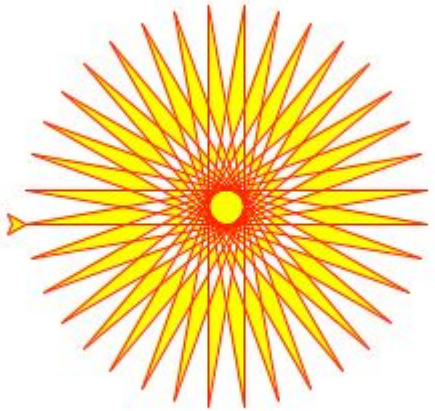
Maak met Turtle een tekening.

Is Anaconda of python geïnstalleerd op je PC, voer het dan lokaal uit!

Installeer je op windows Spyder full package, kun je volgende code rechtstreeks uitvoeren:

Voorbeeld :

```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```



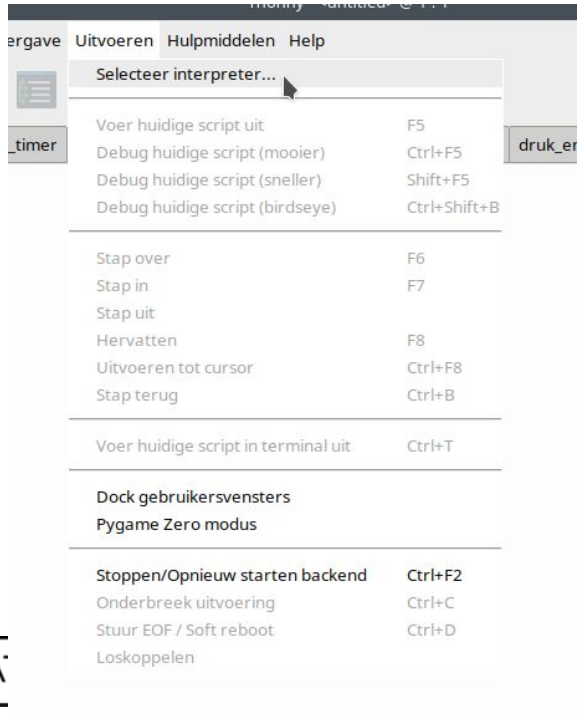
DE CREATIEVE STEM

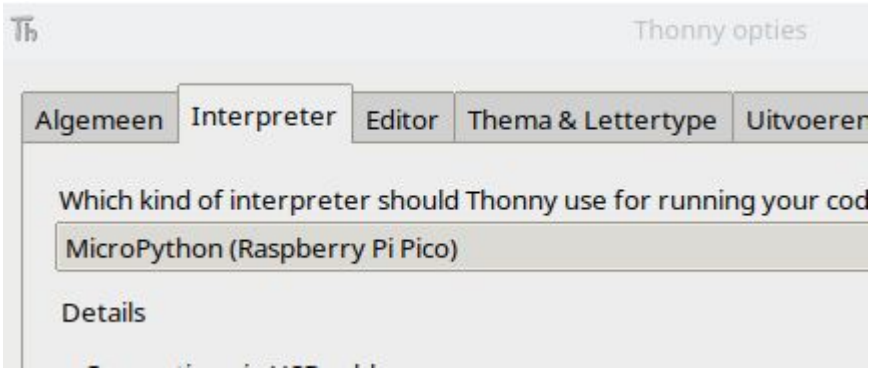
Micropython en de Pico

DE CREATIEVE STEM

Python interpreter selecteren

In menu “Uitvoeren”, selecteer Interpreter… ⇒ kies optie “MicroPython (Raspberry Pi Pico)”





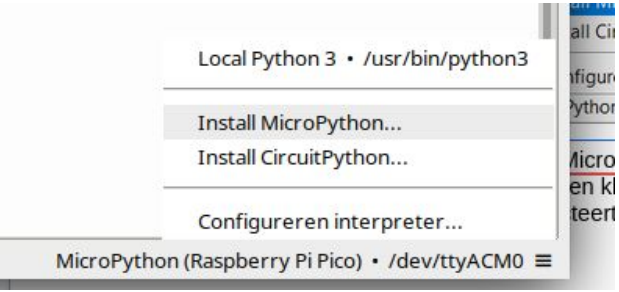
Micropython installeren op de Pico

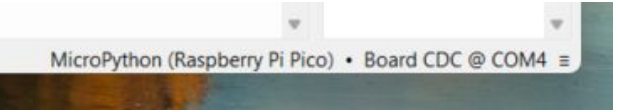
- de Pico voert een programma uit. Wil je je eigen python scripten uitvoeren, dan moet micropython als hoofdprogramma op de Pico geinstalleerd worden.

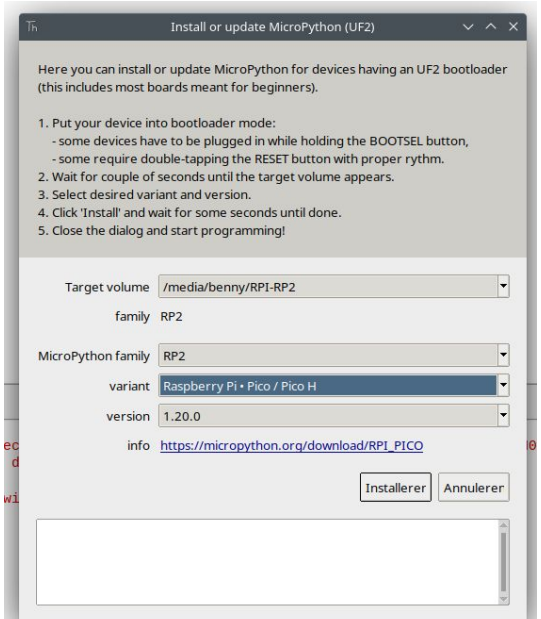
- Koppel de pico los van je PC
- Druk BOOTSEL-knop in, en connecteer met PC. Als de Pico als een usb drive herkend wordt, laat je BOOTSEL los
- Installeer nu MicoPython op de Pico als volgt:

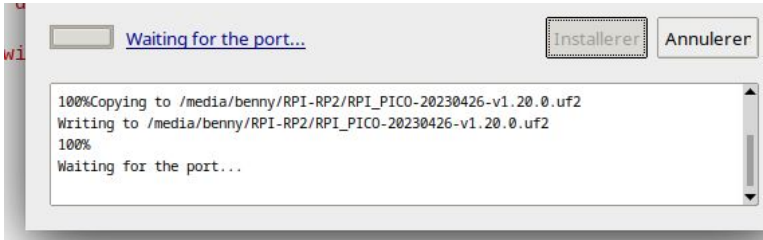
Micropython installeren op de Pico

- In linkerbenedenhoek zie je Python versie van Thonny, en de poort waarop een Pico herkend wordt.
- In linkerbenedenhoek, klik op de python versie, selecteer Installeer MicroPython. Selecteer je versie: RP2, Pico H. Installeer









DE CREATIEVE STEM

Micropython installeren op de Pico

- Installatie van MicroPython moet maar 1x, bij begin, of als je upgrade van MicroPython wil doen. Daarna hoeft BOOTSEL niet meer, de Pico is direct klaar op python scripts te ontvangen en deze uit te voeren op de Pico.

- Bij connectie met je PC en uitvoeren van programma's in Thonny zul je automatisch de output zien op je PC. Data van MicroPython wordt via REPL over de USB kabel naar je PC gestuurd.

- REPL: Read Evaluate Print Loop. Dit is een shell die op de Pico loopt, de output zie je in Thonny.

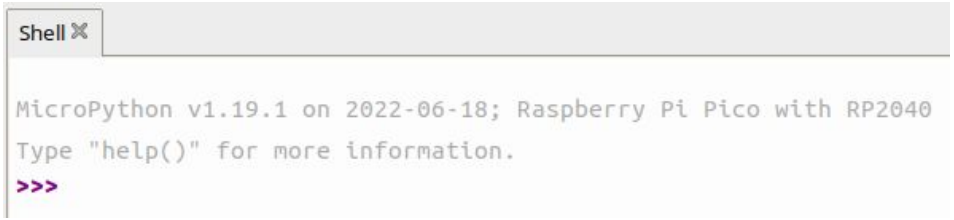
Micropython installeren op de Pico

- Je kan de usb kabel verwijderen om de verbinding te verbreken. Je krijgt dan in de Shell een foutmelding te zien

```
>>>
  Connection lost -- read failed: device reports readiness to read but returned no data (device disconnected or multiple
  access on port?)

  Use Stop/Restart to reconnect.
```

- Als je de verbinding wil herstarten, connecteer de Pico weer met je PC, en druk op de stop of start knop in de toolbar. In de Shell verschijnt of de verbinding gelukt is. Je ziet je versie van MicroPython, en het type microprocessor.



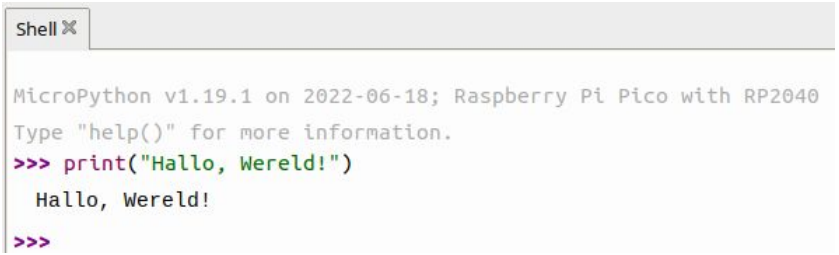
DE CREATIEVE STEM

Opdracht: Hello World

Voer nu het hello world voorbeeld opnieuw uit, maar laat deze code lopen op de Pico.

Doe dan de hello world in de shell van Thonny, en voer opnieuw uit op de Pico

Oplossing



Verschillende van de programma's die we op de lokale python konden uitvoeren, kunnen we nu niet meer uitvoeren. Test bijvoorbeeld ons plotten voorbeeld. Dit zal een fout geven, de module om te plotten is niet aanwezig op de Pico.

DE CREATIEVE STEM

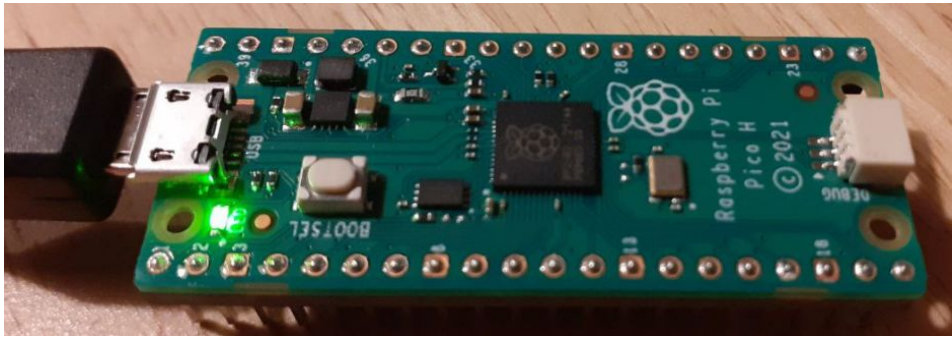
Oefeningen - Interne LED

DE CREATIEVE STEM

Ingebouwde LED

- Ingebouwde LED: verbonden met GPIO-pin 25
- Test code:

```
from machine import Pin
led = Pin(25, Pin.OUT)
led.value(1)
```



- Druk op Run in Toolbar
- We voeren code uit op de MicroPython op de Pico. Code wordt **NIET** opgeslagen op de Pico
- Via de shell kun je verderwerken. Tik in de shell:

```
led.value(0)
```

DE CREATIEVE STEM

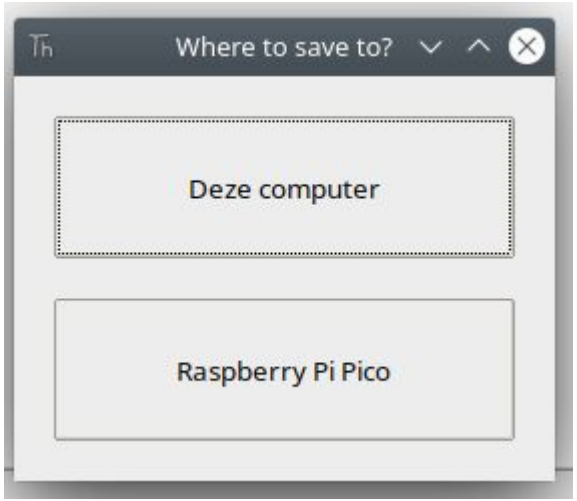
Oefening

- Test toggle: `led.toggle()`
- Trek usb uit, plug terug in na 5 seconden. Doe opnieuw in shell: `led.value(1)`

- Wat stel je vast?

Code opslaan en uitvoeren op Pico

- 3 manieren van opslaan als je op save drukt !
- Eerst: opslaan op PC of op de Pico kiezen

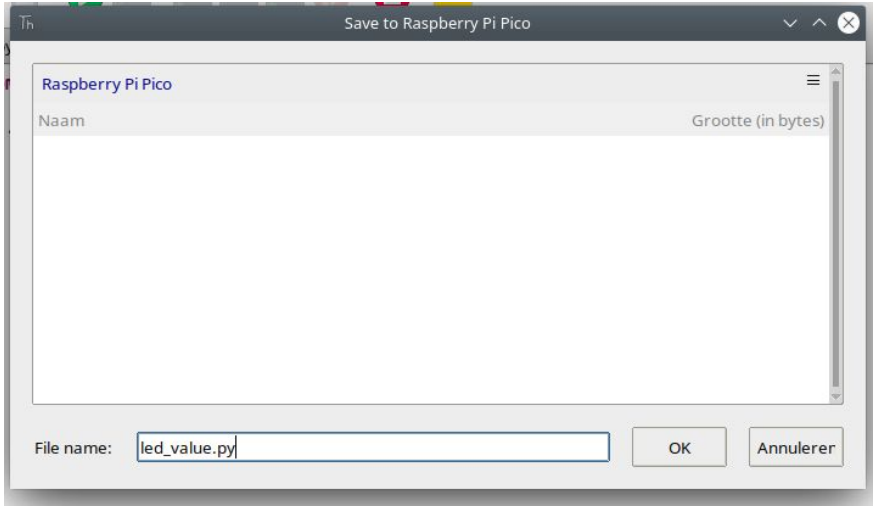


Code opslaan en uitvoeren op Pico

- Je slaat best ook eens op op PC als backup!

  ⇒ doe dit eerst

- Je slaat dan nog eens op op Pico, bv via CTRL+SHIFT+S



Code opslaan en uitvoeren op Pico

- Wil je dat je programma op de Pico wordt uitgevoerd bij aanschakelen?
  ⇒ sla op met als naam: **main.py**

- Het script met naam “main.py” is het script die MicroPython uitvoert bij aanschakelen van de Pico (na opstart van MicroPython zelf)
  Test dit uit met het flikkeren van de interne LED !

Oefeningen - Interne LED

DE CREATIEVE STEM

LED laten knipperen

Opdracht: Laat interne LED aan en uit knipperen met verschillende knippersnelheden

Tip: doe in shell: Wijzig python version naar lokale python, doe dan:

```
>>> from time import sleep
>>> help(sleep)
```

DE CREATIEVE STEM

LED knipperen Oplossing

Een mogelijke oplossing is:

```
# bibliotheken importeren: we gebruiken een Pin en de sleep functie
from machine import Pin
from time import sleep
# led variabele aanmaken die output is op pin 25
led = Pin(25, Pin.OUT)

# Opgelet, 4 spaties niet vergeten voor de while lus
# Oneindige lus: ons programma stopt NOOIT !
while True:
    # Wijzig led status van led.value(0) naar led.value(1) en omgekeerd
    led.toggle()
    # Laat het programma een halve seconden stoppen
    sleep(0.5)
```

DE CREATIEVE STEM

Opdracht

Verander het knipperpatroon van de LED

Een Timer toevoegen

Sleep doet de processor, en ons programma slapen. We kunnen ondertussen niets anders doen.
Beter is een Timer gebruiken, die voor ons na specifieke tijd een actie uitvoert.

"Timer" bibliotheekdocumentatie is te vinden in de officiële MicroPython-documentatie.

Probeer de volgende code:

Timer voorbeeld programma

```
# bibliotheken importeren: we gebruiken een Pin en de Timer
from machine import Pin, Timer

# led variabele aanmaken die output is op pin 25
led = Pin(25, Pin.OUT)
# timer variabele maken
timer = Timer()

# Een functie definieren met de naam “blink”. Deze toggled de LED
def blink(timer):
    # Wijzig led status van led.value(0) naar led.value(1) en omgekeerd
    led.toggle()

# Configureer de timer om onze blink functie op te roepen om de
# 500 milliseconden. Dit moet steeds opnieuw herhaald worden
timer.init(period=500, mode=Timer.PERIODIC, callback=blink)
```

DE CREATIEVE STEM

Vragen

Hoeveel Hertz is 500 ms ?

Wijzig de oefening om ipv period=500 de optie freq te gebruiken.

Oefeningen - Interne Temp

DE CREATIEVE STEM

Meet de temperatuur van de Pico

- De Pico heeft een ingebouwde temperatuursensor
- Deze is aangesloten op ADC4
- Spanning op de ADC4 pin is evenredig met temp, 27 graden Celcius = 0,706 V
- Pico ADC pinnen zijn 12 bit, dus 0 tem 4095, maar MicroPython kan dit schalen naar 16 bit, dus 0 tem 65535. De Pico werkt op 3,3V, dus 65535 op de ADC pin = 3,3V
- Bestudeer volgende code die Temp uitleest

Temp interne sensor

```
# bibliotheken importeren: we gebruiken een ADC Pin en de Timer
from machine import ADC
from time import sleep
# Maak variabele voor de Analoog naar digital conversie
# De temperatuur is via ADC 4.
TempSensor = ADC(4)
# Conversie factor van ADC meting naar 0-3.3V
conversion_factor = 3.3 / 65535
while True:
    #Doe meting en converteer naar gemeten volt
    data = TempSensor.read_u16() * conversion_factor
    #formule om de temperatuur te bekomen
    temperature = 27-(data-0.706)/0.001721
    #toon waarde temperatuur
    print(round(temperature,1), 'C')
    sleep(1)
```

DE CREATIEVE STEM

Vragen

- Hoe nauwkeurig ?
- Meet met multimeter waarde op 3V3 pin, echt 3,3 V?

- Zie dit als snelle indicatie van de temperatuur. Als de processor warm wordt, zul je het ook snel weten zo!

- Oefening: Wijzig de code om een Timer te gebruiken ipv sleep()

DE CREATIEVE STEM

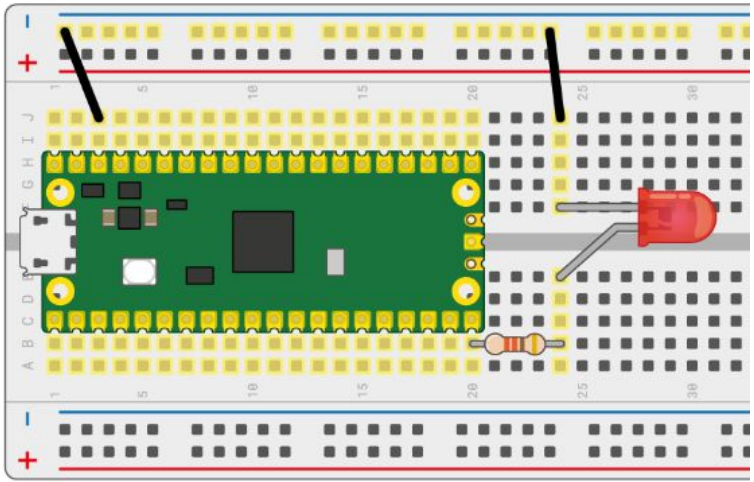
Oefeningen - Verkeerslicht

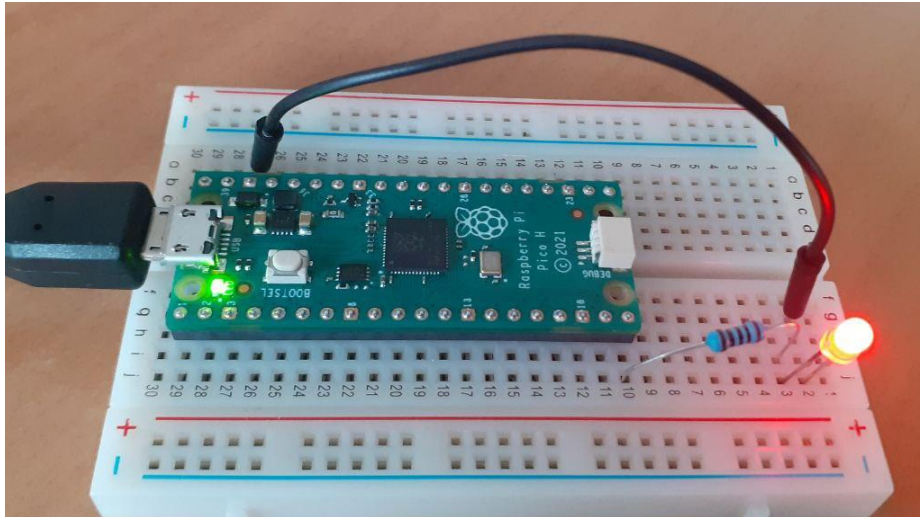
DE CREATIEVE STEM

Opdracht

We maken een eerste fysische schakeling met een breadboard: maak een enkele rode externe LED die je aanstuurt GPIO 15.

Circuit





DE CREATIEVE STEM

Oplossing

Lukt het?

Opdracht

Programmeer een verkeerslicht met Rood, Geel, Groen.
Zoals een echt verkeerslicht moet zich dit oneindig herhalen

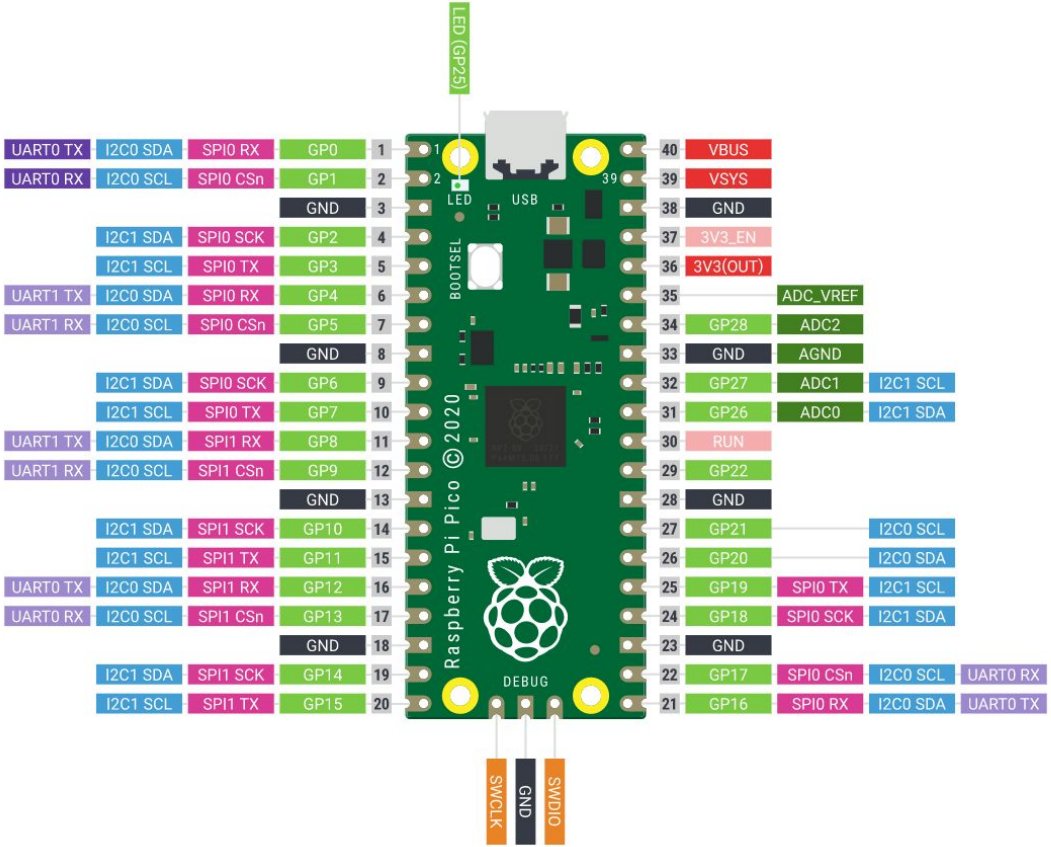
Oefeningen - Temp Meten

DE CREATIEVE STEM

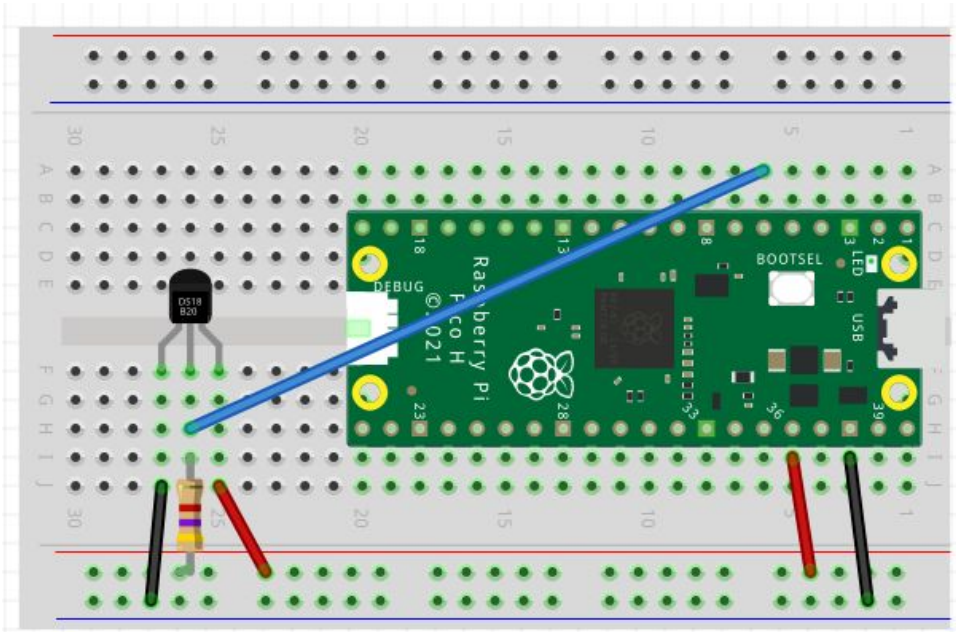
Opdracht

Meet de temperatuur elke halve seconde met de meegeleverde DS18B20 sensor. Laat de gemeten waarden verschijnen in °C in de shell. Warm de sensor op met je hand (en/of koel hem af met ijs) om te testen.

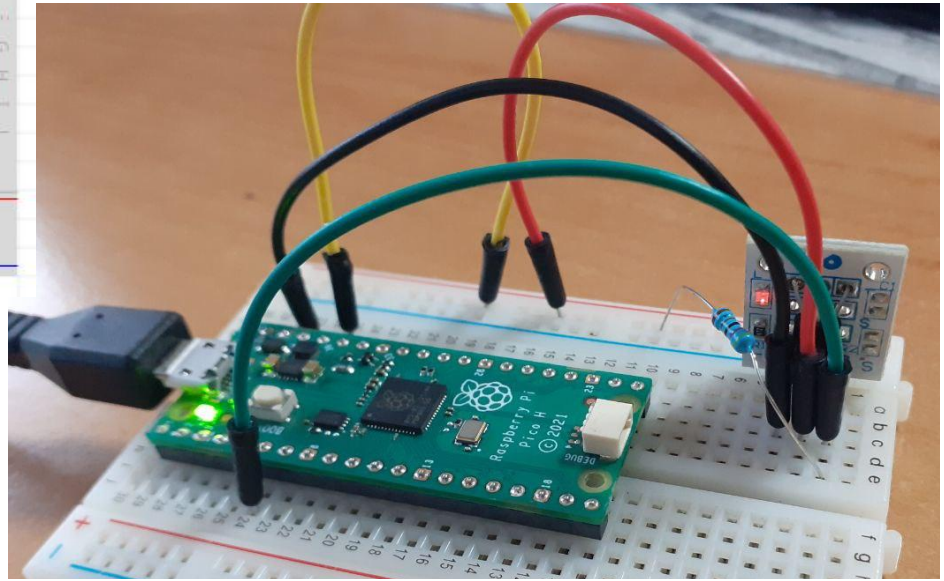
De Pico Pinnen - zie: de interactieve website



Circuit



1 weerstand 4.7 kΩ (geel, paars, rood, goud)



DE CREATIEVE STEM

Bibliotheken

- onewire: bibliotheek om communicatie te doen over een enkele draad, zoals gebruikt door onze senso

- ds18x20: bibliotheek om de DS18B20 uit te lezen, met functies die de temperatuur kunnen bekomen

- Deze bibliotheken zijn beschikbaar voor de Pico, ziede RP2 documentatie

for loop

- De for loop is hoe je een lus maakt in Python

```
for getal in range(1, 11, 1):
    print(getal)


for getal in range(1, 11, 2):
    print(getal)

for getal in range(11):
    print(getal)
```

DE CREATIEVE STEM

lijst inclusie

Lopen (itereren) over een lijst en deze omvormen naar een nieuwe lijst is lijst inclusie. Probeer:

```
[x.upper() for x in 'help']
```

```
[x*2 for x in range(10)]
```

DE CREATIEVE STEM

string join

Lijsten van tekst kun je samenvoegen met de join functie, probeer:

```
''.join([x.upper() for x in 'help'])

'+'.join([x.upper() for x in 'help'])
```

DE CREATIEVE STEM

Oplossing

Zorg dat 4.7 kOhm weerstand **JUIST** is aangesloten

Temperatuur lezen we uit met bv volgende code:

```
# importeer de nodige bibliotheken
import machine, onewire, ds18x20, time

# data pin verbonden met GPIO4
ds_pin = machine.Pin(4)
# sensor object aanmaken via doorgeven van de juiste communicatielijn (onewire over GPIO4)
ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin))

# bekom alle temp sensoren op de datalijn. Normaal maar 1 bij ons
roms = ds_sensor.scan()
# print hoeveel sensoren gevonden. len() bekomt lengte van een lijst
print('Found DS devices: ', len(roms))
```

DE CREATIEVE STEM

Oplossing

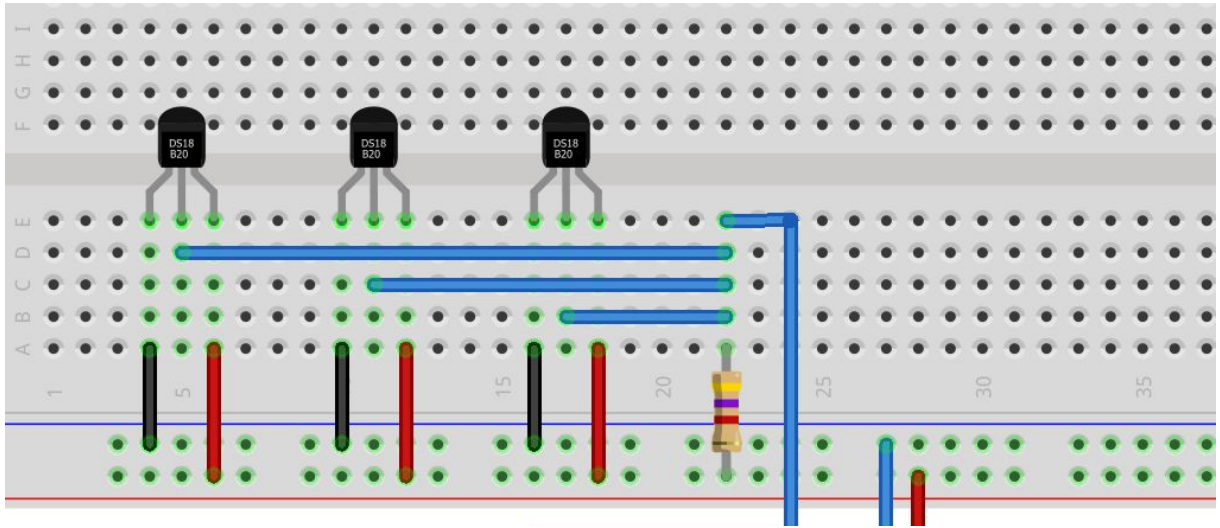
```
# oneindige lus voor programma dat nooit stopt
while True:
 # lees de temperatuur uit
 ds_sensor.convert_temp()
 # wacht 750 ms om tijd te geven om sensors uit te lezen
 time.sleep_ms(750)
 # print in de shell de temperatuur van elke sensor
 for rom in roms:
   # print de 64bit code van de sensor als hex string
   print('Device', ' '.join([hex(x) for x in rom]))
   # print de gemeten temperatuur
   print(ds_sensor.read_temp(rom))

 # wacht 5 seconden voor volgende meting te doen.
 time.sleep(5)
```

DE CREATIEVE STEM

Meerdere temperatuursensoren

Voor gevorderden, temp op verschillende locaties



Oefeningen - Luchtdruk Meten

DE CREATIEVE STEM

Opdracht

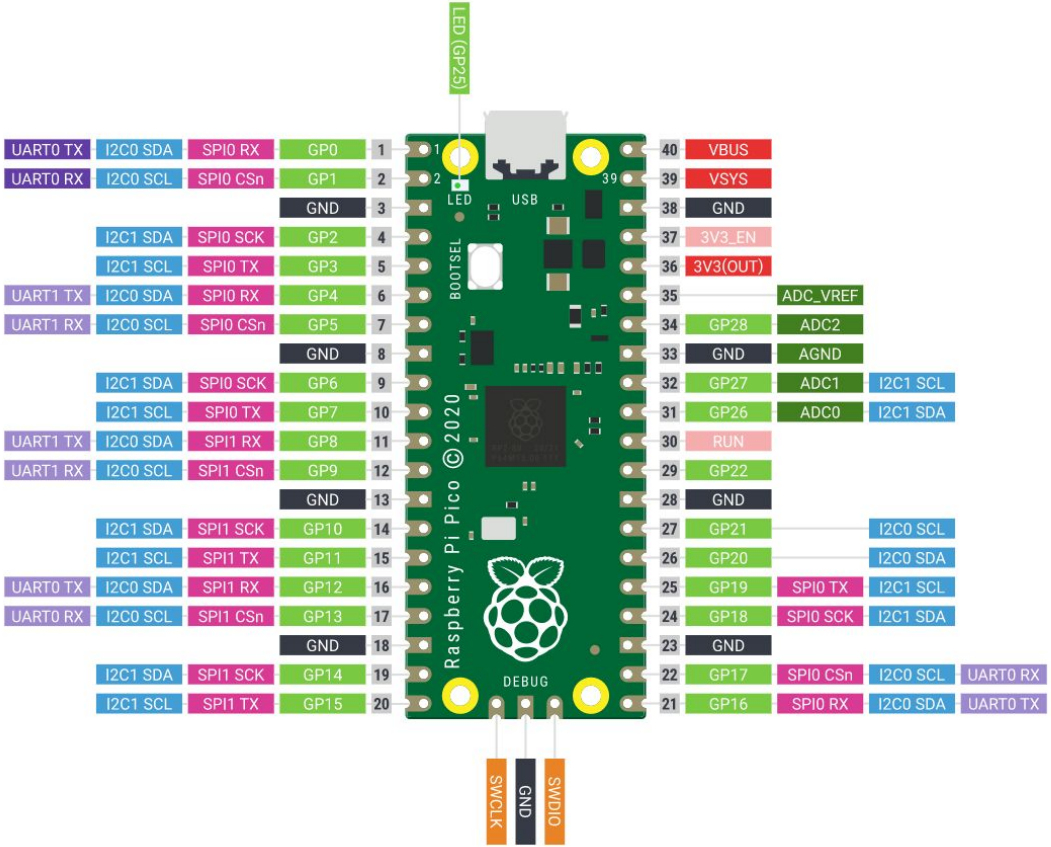
Meet de luchtdruk met de meegeleverde BMP/E280 sensor. Laat de gemeten waarden verschijnen in Pa in de shell. Creëer een overdruk of onderdruk rond de sensor om te testen.

Circuit

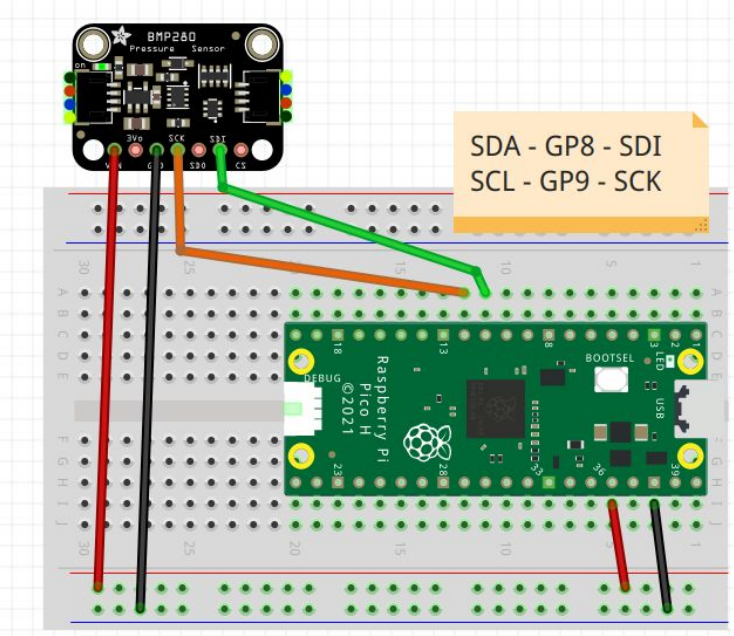
Voorlopig gebruiken we I2C op pinnen GP8 en GP9;

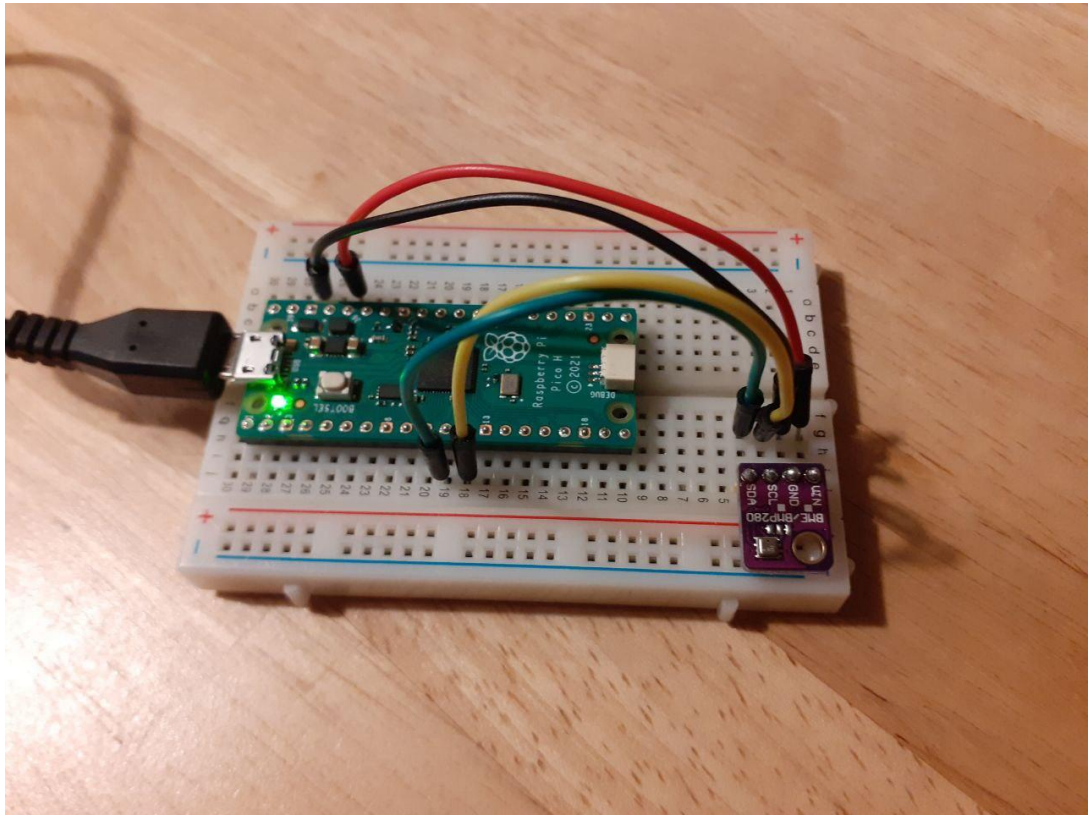
- Sensor Vin op de 3V3 pin
- Sensor GND pin op een van de GND pinnen
- Sensor SCK pin op de GP9 (SCL)
- Sensor SDI pin op de GP8 (SDA)

De Pico Pinnen - zie: de interactieve website



Circuit



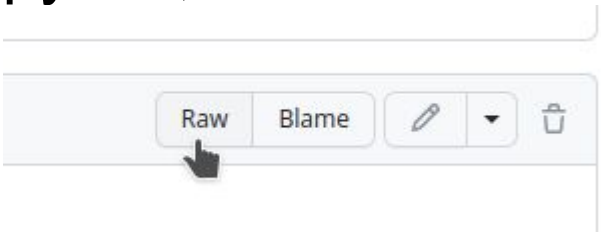


DE CREATIEVE STEM

Bibliotheek bmp280

We hebben een bibliotheek nodig die over I2C kan communiceren met de BMP280.
We gebruiken **micropython-bmp280**
https://github.com/dafvid/micropython-bmp280

Klik op het bmp280.py file, klik dan rechts op de Raw knop



Sla op met CTRL+S als bmp280.py.

DE CREATIEVE STEM

Module importeren

In python kun je modules importeren via

```
import bmp280
```

Eerst wordt gekeken of er een bestand is in dezelfde map met de naam <module>.py.

Gezien code op de Pico uitgevoerd wordt, moet dit dus **op de Pico opgeslagen** worden met deze naam!

DE CREATIEVE STEM

Script

Volgend script kun je gebruiken om temp en druk te meten met de bmp280 sensor :

```
from machine import Pin, I2C
import bmp280
import time

# zet I2C connectie op via pin 8 en 9.
bus = I2C(0, scl=Pin(9), sda=Pin(8), freq=200000)
# maak een bmp object over de I2C connectie
bmp = bmp280.BMP280(bus)
```

Script

```
# de bmp kan in low power en high power werken. We willen snel
# wijzigingen detecteren, dus kiezen we high power mode INDOOR.
# Kies je geen use case, dan wordt BMP280_CASE_HANDHELD_DYN
# geselecteerd, welke een lagere resolutie heeft, maar nog goed
# zou werken
bmp.use_case(bmp280.BMP280_CASE_INDOOR)

while True:
    # lees druk uit
    pressure = bmp.pressure
    p_bar = pressure/100000
    p_mmHg = pressure/133.3224
    # lees temperatuur uit
    temperature = bmp.temperature
    print("Temperature: {}°C".format(temperature))
    print("Pressure: {} Pa, {} bar, {} mmHg".format(pressure, p_bar,
          p_mmHg))
    # wacht 1 seconde
    time.sleep(1)
```

DE CREATIEVE STEM

Oefeningen - Hoogte Meten

DE CREATIEVE STEM

Hoogte uit luchtdruk

- Indien we weten wat de druk op zeeniveau is, en de temperatuur van de luchtlaag, kunnen we de hoogte bepalen uit een drukmeting.
- Beperkte nauwkeurigheid!
- De relatie tussen atmosferische druk en temperatuur wordt gegeven door de formule:
  - $P = 101325\ (1 - 2.25577\ 10^{-5}\ h)^{5.25588}$

P is in Pascal, en h in meter boven zeeniveau. Dus:

$$h = \frac{10^{\left(\frac{\log\log\left(\frac{P}{101325}\right)}{5.25588}\right)} - 1}{-2.25577 \times 10^{-5}}$$

DE CREATIEVE STEM

Code

```
from machine import Pin, I2C
import bmp280
import time

# zet I2C connectie op via pin 8 en 9.
bus = I2C(0, scl=Pin(9), sda=Pin(8), freq=200000)
# maak een bmp object over de I2C connectie
bmp = bmp280.BMP280(bus)

# de bmp kan in low power en high power werken. We willen snel
# wijzigingen detecteren, dus kiezen we high power mode INDOOR.
# Kies je geen use case, dan wordt BMP280_CASE_HANDHELD_DYN
# geselecteerd, welke een lagere resolutie heeft, maar nog goed
# zou werken
bmp.use_case(bmp280.BMP280_CASE_INDOOR)
```

DE CREATIEVE STEM

Code

```
def get_altitude(pressure, temperature, sea_level_pressure_hPa=1013.25):
        # de druk op het punt waar je de hoogte van wil bepalen
        # temperature = de buitenlucht temperatuur in graden Celcius.
        # sea_level_pressure_hPa = druk op zeeniveau, een getal in hPa

        # volgende formule berekent de hoogte.
        altitude = ((pow((sea_level_pressure_hPa / pressure),
                    (1.0 / 5.257)) - 1)
                   * (temperature + 273.15)) / 0.0065
        # onze functie geeft de hoogte terug aan de oproeper.
        return altitude

# Als de Pico aanschakelt meten we de druk, we slaan die op als druk
# op zeeniveau
sea_level_pressure_hPa = bmp.pressure/100
time.sleep(1)
```

DE CREATIEVE STEM

Code

```
while True:
    # lees druk uit
    pressure = bmp.pressure
    p_bar = pressure/100000
    p_mmHg = pressure/133.3224
    # lees temperatuur uit
    temperature=bmp.temperature
    print("Temperature: {}°C".format(temperature))
    print("Pressure: {} Pa, {} bar, {} mmHg".format(pressure, p_bar,
                                                    p_mmHg))
    # we berekenen nu de hoogte in meter
    hoogte = get_altitude(pressure/100, temperature,
                          sea_level_pressure_hPa)
    print("Hoogte: {} m".format(hoogte))
    # wacht 1 seconde    time.sleep(1)
```

DE CREATIEVE STEM

Opdracht

- Wandel rond met je laptop een trap op en af. Wordt de hoogtewijziging gedetecteerd?
- Wijzig de code om met een Timer te werken
- Is temperatuur meting even goed als de DS18B20 ?
- Kun je code aanpassen om BEIDE sensoren te gebruiken?
- Leuk ware een scherm toe te voegen aan het project en de hoogte/druk/temp te tonen. Hier een voorbeeld hoe dat kan: zie hier (electrocredible)

DE CREATIEVE STEM

Oefeningen - Data opslaan op SD kaart

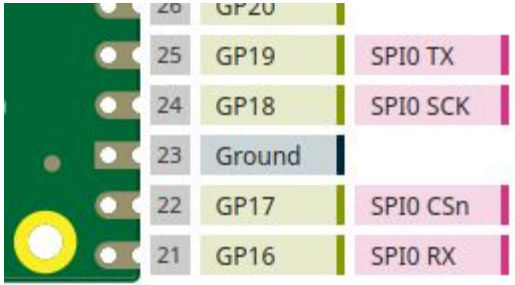
DE CREATIEVE STEM

SD-kaart module

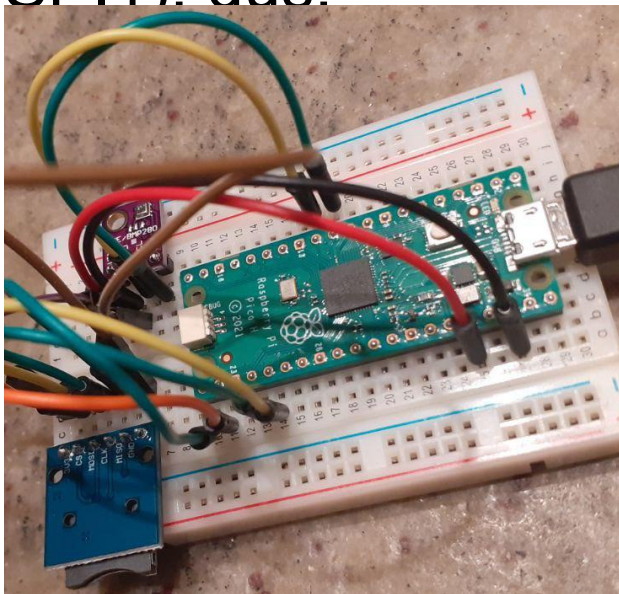
- Voor een satelliet slaan we data best op.
- Communicatie is voor gevorderden (radio of wifi), de Pico H kan dat niet, de Pico WH wel al WiFi
- Je hebt een bord voor de Pico nodig die werkt op 3.3V, veel Arduino borden kunnen dat niet!
- Communicatie met SD-kaart module is meestal met het SPI protocol

Circuit

- Combineer GND en VCC van de BMP280 met SD-kaart Shield
- Voeg SPI connecties toe: CS, SCK, MOSI en MISO
- We gebruiken SPI0 van de Pico (niet SPI1), dus:



1. CS naar pin 22 (GP17),
2. SCK of CLK naar pin 24 (GP18),
3. MOSI naar pin 25 (TX, GP19),
4. MISO naar pin 21 (RX, GP16).



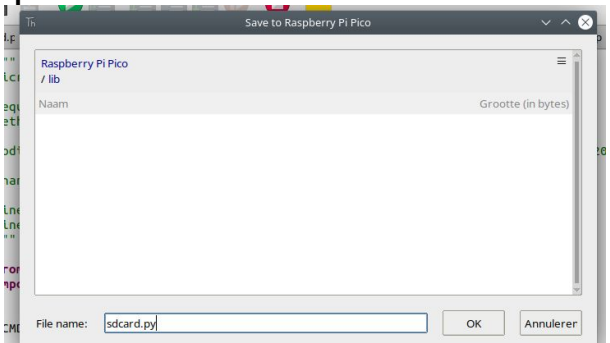
DE CREATIEVE STEM

sdcard bibliotheek

Om de code overzichtelijk te houden, is het een goed idee om om de bibliotheek op te slaan op de pico in een map **lib**, in die map kun je alle hulpcode verzamelen. In versie 1.19 van Micropython is de SDCard bibliotheek voor de Pico nog niet standaard inbegrepen, maar verschillende personen hebben online wel al zo'n bibliotheek beschikbaar. We gebruiken deze van CoreElectronics:

https://github.com/CoreElectronics/CE-Makerverse-R2R-DAC-MicroPython-Module/blob/main/sdcard.py

Ga, net zoals bij de bmp280, naar de **raw**, versie, download deze, open ze via Thonny, en sla op op de Pico in map **lib** met naam **sdcard.py**. Alle bestanden opgeslagen in de map lib zullen als eerste gevonden worden door python via het import commando. Je kan dus ook de bmp280.py code in deze map plaatsen.



Formateer de SD kaart in FAT32

- Kaarten zijn reeds geformateerd. Doe je toch, dan …
- Linux: gparted, en selecteer FAT32
- Windows: Connecteer de kaart met je Windows PC
  - Open Explorer (Bestanden)
  - Rechts klik op de kaart drive letter in Explorer
  - Selecteer "Formatteer..."
  - Zorg dat je FAT32 hebt geselecteerd onder het “Bestandsysteem” drop-down drop-down menu
  - Klik "Start" om het formatteren te starten.

DE CREATIEVE STEM

Script: opslaan gegevens

- We slaan op als een tekstbestand. Een csv bestand: comma seperated values.
- In windows: wijzig bestandsextensie van .txt naar .csv, daarna kun je het openen in Excel, Google spreadsheet, …
- Geen tijdsbepaling op Pico, dus opslag in nieuw bestand via een teller die op de SD-kaart staat. Dan tijd sinds opstart als tijd in bestand!

Script

```
from machine import Pin, I2C, SPI
import bmp280
import sdcard
import time, uos

# zet I2C connectie op via pin 8 en 9.
bus = I2C(0, scl=Pin(9), sda=Pin(8), freq=200000)
# maak een bmp object over de I2C connectie voor drukmeting
bmp = bmp280.BMP280(bus)

# Maak een chip select (CS) pin (en start het HIGH)
cs = machine.Pin(17, machine.Pin.OUT)
```

DE CREATIEVE STEM

```
# Intialiseer SPI peripheral 0 (start met 1 MHz)
# We doen dit met pinnen voor sck, mosi en miso:
spi = SPI(0,                                          # SPI0 of SPI1
          baudrate=1000000,
          polarity=0,
          phase=0,
          bits=8,
          firstbit=machine.SPI.MSB,
          sck=machine.Pin(18, machine.Pin.OUT),
          mosi=machine.Pin(19, machine.Pin.OUT),  # De SPI0 TX
          miso=machine.Pin(16, machine.Pin.OUT))  # De SPI0 RX
# Creeer een SDCard object sd met de spi en cs:
sd = sdcard.SDCard(spi, cs)
```

```
# de bmp kan in low power en high power werken. We willen snel
# wijzigingen detecteren, dus kiezen we high power mode INDOOR.
# Kies je geen use case, dan wordt BMP280_CASE_HANDHELD_DYN
# geselecteerd, welke een lagere resolutie heeft, maar nog goed
# zou werken
bmp.use_case(bmp280.BMP280_CASE_INDOOR)

def get_altitude(pressure, temperature, sea_level_pressure_hPa=1013.25):
    # de druk op het punt waar je de hoogte van wil bepalen
    # temperature = de buitenlucht temperatuur in graden Celcius.
    # sea_level_pressure_hPa = druk op zeeniveau, een getal in hPa

    # volgende formule berekent de hoogte.
    altitude = ((pow((sea_level_pressure_hPa / pressure), (1.0 / 5.257)) - 1)
                * (temperature + 273.15)) / 0.0065
    # onze functie geeft de hoogte terug aan de oproeper.
    return altitude
```

```
# Als de Pico aanschakelt meten we de druk, we slaan die op als druk op zeeniveau
sea_level_pressure_hPa = bmp.pressure/100
time.sleep(1)

# We openen de SD kaart en koppelen aan directory /sd
vfs = uos.VfsFat(sd)
uos.mount(vfs, "/sd")
# we schrijven de directory van de SD kaart uit in REPL
files = uos.listdir('/sd')
print("SD-kaart bestanden:", files)
# we wijzigen uos naar directory /sd
uos.chdir('/sd')
```

```
# Controleer of counter.txt aanwezig is
counter = 1
if "counter.txt" in files:
   # open het en lees de teller uit (read = 'r')
   print ("counter found")
   with open('/sd/counter.txt', "r") as countfile:
       counter = countfile.readline()
       print(counter)
       counter = int(counter)
   # we tellen 1 bij de teller en slaan op voor volgende keer (write = 'w')
   with open('/sd/counter.txt', "w") as countfile:
       countfile.write(str(counter + 1)) #schrijf teller
       countfile.write('\n')             # Een nieuwe lijn op einde
else:
   # counter.txt bestaat niet, dus maak het met inhoud 1
   with open('/sd/counter.txt', "w") as countfile:
       print("Maken counter.txt aan!")
       countfile.write(str(counter)) # de teller als string
       countfile.write('\n')         # een nieuwe lijn
```

```
# sla sensordata op in bTdat_XXX.txt met XXX de teller:
filenamedata = 'bTdat_' + str(counter) + '.txt'
# schrijf de header van de data uit (kolomheaders)
with open('/sd/' + filenamedata, "a") as datafile:
    datafile.write('time [s]') # Write time sample was taken in seconds
    datafile.write(' ; ') # Een separator
    datafile.write('temp [C]')
    datafile.write(' ; ') # Een separator
    datafile.write('druk [Pa]')
    datafile.write(' ; ') # Een separator
    datafile.write('druk [bar]')
    datafile.write(' ; ') # Een separator
    datafile.write('druk [mmHg]')
    datafile.write(' ; ') # Een separator
    datafile.write('hoogte [m]')
    datafile.write(' ; ') # Een separator
    datafile.write('zeedruk [hPa]')
    datafile.write('\n') # A nieuwe lijn
```

```
# voer nu metingen uit zolang er stroom is
while True:
    # lees druk uit
    pressure = bmp.pressure
    p_bar = pressure/100000
    p_mmHg = pressure/133.3224
    # lees temperatuur uit
    temperature=bmp.temperature
    print("Temperature: {}°C".format(temperature))
    print("Pressure: {} Pa, {} bar, {} mmHg".format(pressure, p_bar, p_mmHg))
    # we berekenen nu de hoogte in meter
    hoogte = get_altitude(pressure/100, temperature, sea_level_pressure_hPa)
    print("Hoogte: {} m".format(hoogte))
```

```
    # we slaan de sensordata op door achteraan in bestand te schrijven
    # toevoegen = append = "a"
    with open('/sd/' + filenamedata, "a") as datafile:
        # bereken aantal seconden dat de Pico aan staat:
        t = time.ticks_ms()/1000
        datafile.write(str(t)) # Schrijf tijd van meting in seconden
        datafile.write(' ; ') # Een separator
        datafile.write(str(temperature))
        datafile.write(' ; ') # Een separator
        datafile.write(str(pressure))
        datafile.write(' ; ') # Een separator
        datafile.write(str(p_bar))
        datafile.write(' ; ') # Een separator
        datafile.write(str(p_mmHg))
        datafile.write(' ; ') # Een separator
        datafile.write(str(hoogte))
        datafile.write(' ; ') # Een separator
        datafile.write(str(sea_level_pressure_hPa))
        datafile.write('\n') # Een nieuwe lijn
    # wacht 1 seconde
    time.sleep(1)
```

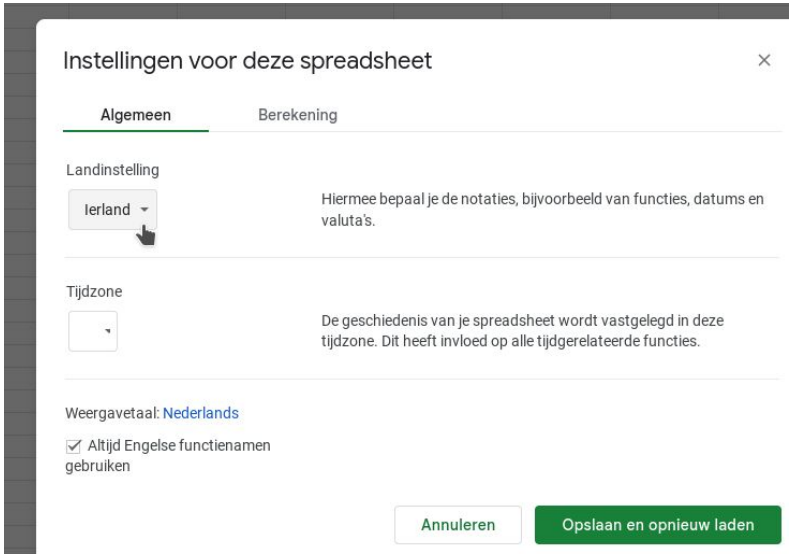
Voorbeeld output:

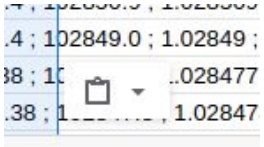
```
time [s] ; temp [C] ; druk [Pa] ; druk [bar] ; druk [mmHg] ; hoogte [m] ; zeedruk [hPa]
7.894 ; 21.33 ; 102857.4 ; 1.028574 ; 771.4939 ; 0.2160292 ; 1028.6
8.922 ; 21.41 ; 102856.4 ; 1.028564 ; 771.4864 ; 0.3025231 ; 1028.6
9.95 ; 21.45 ; 102856.4 ; 1.028564 ; 771.4865 ; 0.3025642 ; 1028.6
10.977 ; 21.46 ; 102856.7 ; 1.028567 ; 771.4888 ; 0.2755589 ; 1028.6
12.005 ; 21.45 ; 102856.8 ; 1.028568 ; 771.489 ; 0.2701466 ; 1028.6
13.034 ; 21.44 ; 102856.9 ; 1.028569 ; 771.4897 ; 0.2647346 ; 1028.6
14.063 ; 21.45 ; 102856.1 ; 1.028561 ; 771.484 ; 0.3295788 ; 1028.6
15.096 ; 21.47 ; 102855.3 ; 1.028553 ; 771.4777 ; 0.3998441 ; 1028.6
16.124 ; 21.47 ; 102852.4 ; 1.028524 ; 771.4561 ; 0.6375892 ; 1028.6
17.151 ; 21.48 ; 102851.0 ; 1.02851 ; 771.446 ; 0.7510839 ; 1028.6
18.179 ; 21.48 ; 102851.6 ; 1.028516 ; 771.4502 ; 0.7024527 ; 1028.6
19.207 ; 21.48 ; 102853.2 ; 1.028532 ; 771.4622 ; 0.5727691 ; 1028.6
20.236 ; 21.48 ; 102852.8 ; 1.028528 ; 771.4596 ; 0.5997865 ; 1028.6
21.265 ; 21.46 ; 102853.9 ; 1.028539 ; 771.4676 ; 0.5132959 ; 1028.6
22.298 ; 21.46 ; 102854.5 ; 1.028545 ; 771.4718 ; 0.4646679 ; 1028.6
23.326 ; 21.47 ; 102854.0 ; 1.028539 ; 771.4678 ; 0.5079101 ; 1028.6
24.353 ; 21.47 ; 102854.8 ; 1.028548 ; 771.474 ; 0.4376672 ; 1028.6
25.381 ; 21.47 ; 102856.5 ; 1.028565 ; 771.4868 ; 0.2971814 ; 1028.6
26.409 ; 21.47 ; 102855.4 ; 1.028554 ; 771.4784 ; 0.3890375 ; 1028.6
27.438 ; 21.48 ; 102856.1 ; 1.028562 ; 771.4844 ; 0.3242089 ; 1028.6
28.466 ; 21.47 ; 102857.0 ; 1.02857 ; 771.491 ; 0.2485517 ; 1028.6
29.495 ; 21.48 ; 102857.4 ; 1.028574 ; 771.4939 ; 0.2161393 ; 1028.6
30.527 ; 21.48 ; 102857.6 ; 1.028576 ; 771.4955 ; 0.1999288 ; 1028.6
```

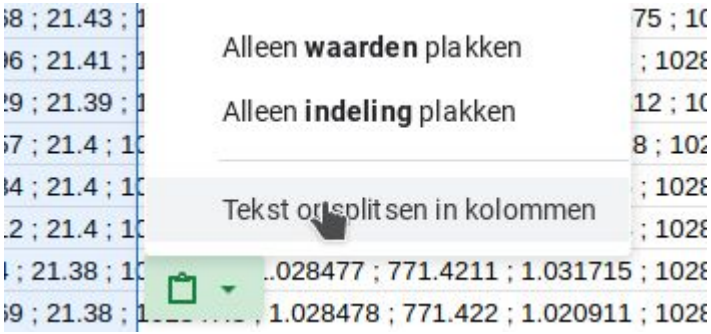
DE CREATIEVE STEM

Analyseren sensor data

menu Bestand → Instellingen ⇒ Ierland

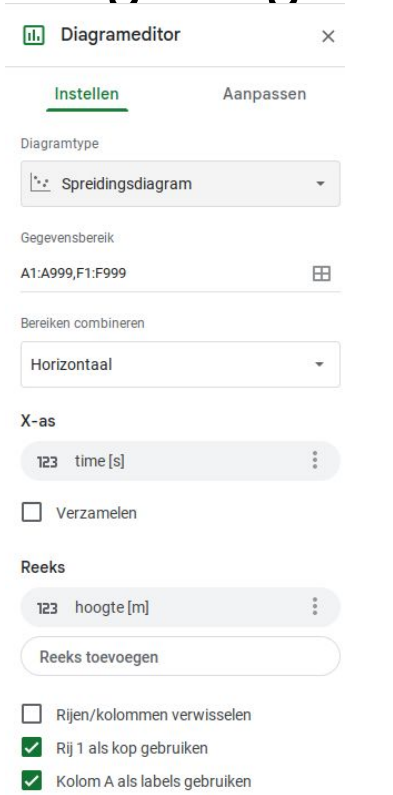


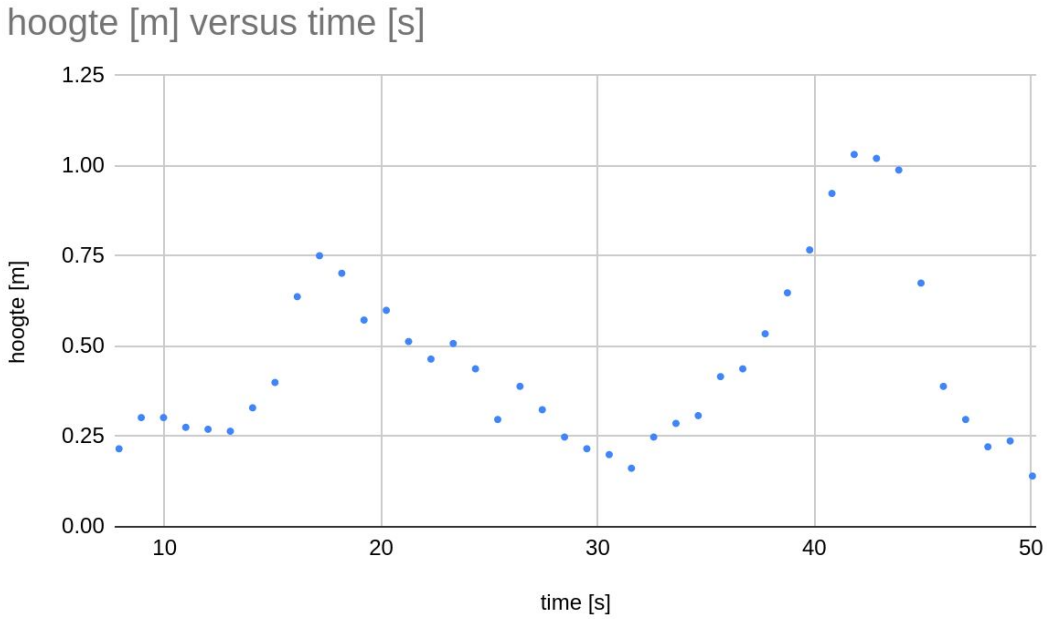




Analyseren sensor data

Selecteer kolom A en F, toolbar: diagram toevoegen, kies type Spreidingsdiagram:





Opdracht

1. Maak een diagram van de temperatuur versus de tijd
2. Kun je de snelheid van stijgen berekenen? Hoe?
3. De “zeedruk” is de druk bij opstart van de Pico. Maar mogelijks stond de Pico niet op de grond toen je startte? Je kan achteraf, of op bepaald moment na opstart Pico, een nulmeting op de grond doen (de vloer). Je dient dan de correcte hoogte te herberekenen in je rekenblad achteraf. Een idee hoe je dat doet?

DE CREATIEVE STEM

Radio Communicatie

DE CREATIEVE STEM

Voor gevorderden

Discussie over hoe radio communicatie opzetten

Optioneel Materiaal
Meer over Python

DE CREATIEVE STEM

Herhaling

- Variabelen. Vb: `t = turtle.Turtle()`
- Range: `range(10)`
- import modules, functies: `import turtle`
- functies maken: `def berekenTotaal(aantal, prijs):`
- strings (str)
- Lijst, operatoren & slicen: `Lijst03[0][-2]` en `s1[::-1]`
- print functie

DE CREATIEVE STEM

Oefening

Laat gebruiken een tekst ingeven. Geef als antwoord het woord dat elke seconde een letter langer wordt.

Voorbeeld:

Geef woord: pen

Antwoord:

p

pe

pen

DE CREATIEVE STEM

Booleans en Condities

DE CREATIEVE STEM

Boolean

Een type variabele die twee waarden kan hebben:

True en False

```
x = 1000
naam = 'Jan'
checks = [x == 1000, x > 100, x <= 100, naam == 'Jan', naam != 'Jan', naam == 'Mieke']
print(checks)
```

Symbol	Description	Example
<	Less than	print(10< 20)
>	Greater than	print(10> 20)
<=	Less than or equal to	print(10<= 20)
>=	Greater than or equal to	print(10>= 20)
!=	Not equal to	print(10 != 20)
==	Equal to	print(10 == 20)

DE CREATIEVE STEM

Data Types: Boolean

- Boolean Data Type: True, False

  Note:
  - number **0 is False**, others True
  - Empty string **' ' is False**, others True

- Result of conditional test.
- Logical operators to compute with them

```
a = 10
checks = [a < 15 and a > 15, a < 15 or a > 15, not 10 < 15]
print(checks)
```

Symbol/ Keyword	Description	Example
and	Return True if both conditions are True. Return False if any of the condition is False.	print(10<20 and 20<30) Output: True
or	Return True if any of the conditions are True. Return False when both conditions are False.	print(10<20 or 20<30) Output: True
not	Reverses the Boolean value. Return True if given condition returns False, return False if the given condition is True.	print(not 10<20) Output: False

DE CREATIEVE STEM

Control Flow: if elif else

```
def testInput():
 x = int(input("Geef een geheel getal in:"))
 if x < 0:
   x = 0
   print('Negatief, gewijzigd in nul')
 elif x == 0:
   print('Nul')
 elif x == 1:
   print('Single')
 else:
   print('Meer')
 return x

gegeven_waarde = testInput()
```

Een functie in input vraagt en er een bewerking mee doet.

De berekende waarde wordt teruggegeven via

```
return x
```

Oef: maak functie die een getal vraagt en de derde macht a^3 berekend

Test: Geef String als input

Loops

DE CREATIEVE STEM

Control Flow: for loop

For loop loops over a sequence type

```
# Measure some strings:
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
```

Range is often used in combination with for loop:

```
range(5, 10)
   5, 6, 7, 8, 9

range(0, 10, 3)
   0, 3, 6, 9

range(-10, -100, -30)
  -10, -40, -70
```

```
a = ['Mary', 'had', 'a', 'little', 'lamb']
for i in range(len(a)):
    print(i, a[i])
```

DE CREATIEVE STEM

Control Flow: for loop - else, break and continue

Test:

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
    else:
        # loop fell through without finding a factor
        print(n, 'is a prime number')
```

En:

```
for num in range(2, 10):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found a number", num)
```

DE CREATIEVE STEM

Control Flow: While loop

Doe volgende loop:

```
# Fibonacci series:
# the sum of two elements defines the next
a, b = 0, 1
while a < 10:
   print(a)
   a, b = b, a+b
```

Turtle opnieuw:

```
from turtle import *
def spiral(X, Y, staptoename, startstaplengte,
           aantalstappen, hoek):
 bgcolor('grey')
 color('black')
 penup()
 goto(X, Y)
 pendown()
 steps = 0
 stap = startstaplengte
 while steps < aantalstappen:
   forward(stap)
   right(hoek)
   stap = stap + staptoename
   steps += 1

spiral(200, 200, 2, 10, 20, 30)
```

DE CREATIEVE STEM

Oefening

Maak een lijst van honderd willekeurig gekozen ASCII letters.

Gebruik hiervoor:

```
from random import seed, randint
from string import ascii_uppercase
```

Toon je lijst.

Print dan je oplossing als een string van 100 karakters.

DE CREATIEVE STEM

Witruimte & Commentaar

DE CREATIEVE STEM

Whitespace Rules

De belangrijkste inspringingsregels van PEP 8 zijn de volgende:

- Gebruik 4 opeenvolgende spaties om de inspringing aan te geven.
- Geef de voorkeur aan spaties boven tabs.

Gebruik:

- Witruimte is zinvol in Python!
- Gebruik een nieuwe regel om een regel code te beëindigen
- Gebruik \ wanneer je voortijdig naar de volgende regel moet gaan
- Geen accolades {} om codeblokken te markeren, gebruik in plaats daarvan consistente inspringing
- Eerste regel met minder inspringing is buiten het blok
- Eerste regel met meer inspringing start een genest blok
- Dubbele punten (:) beginnen een nieuw blok in veel constructies, b.v. functie def, als-dan-clausules

Comments

- Gebruik symbool # voor commentaar
- Commentaar met meerdere regels begint met drievoudige aanhalingstekens ''' en eindigt met driedubbele aanhalingstekens ''' … het is meerregelige tekst die niet aan een variabele is toegewezen!

```
# My first Python Comment
print("5+5=",10) # Printing Command
'''
this is a
Multiline
Comment
'''
```

Commentaar over meerdere regels wordt gebruikt voor de ingebouwde hulp. In Notebook, doe: `help(print)`

DE CREATIEVE STEM

Fouten opvangen

DE CREATIEVE STEM

Catching errors: try … except

Je kan fouten opvangen (catch) die opgeworpen worden via een **try** block. Je specificeert wat moet gebeuren in de **except**. Voorbeeld:

```
while True:
  try:
      x = int(input("Geef een nummer aub: "))
      break
  except ValueError as msg:
      print("Oops!  Dat was geen geldig nummer. \n Error: ", msg, "\nProbeer opnieuw...")
```

Hier, de ingebouwde functie int() **raises** een **Error** als geen nummer

Meer: https://docs.python.org/3/tutorial/errors.html

DE CREATIEVE STEM

Object Georiënteerd Programmeren

DE CREATIEVE STEM

Object Oriented Programming: Classes

```
class MyClass:
    """A simple example class"""
    students = None

    def __init__(self):
        self.students = ['John', 'Mary']

    def add_student(self, name):
        self.students.append(name)

    def print_students(self):
        for student in self.students:
          print(student)

    def number_of_students(self):
        return len(self.students)


testclass = MyClass()
print(testclass.number_of_students())
testclass.add_student('Rose')
print(testclass.number_of_students())
testclass.print_students()
print(testclass.students)
```

Let op volgende elementen:

1. klasse maak je met **class** keyword
2. klasse variabelen worden gedeeld over een instantie (hier testclass)
3. Er is een **__init__** functie die bij instantiëren uitgevoerd wordt, dus bij aanmaak van een object via **MyClass()**
4. ALLE functies hebben als eerste parameter **self**, het gemaakt object van de klasse is dit.
5. In de code van de klasse gebruik je self.xxxx om een variabele of functie aan te roepen. In gebruik doe je objectnaam.xxxx
6. Een klasse is een goede manier om propere code te schrijven, met samenhorende elementen samen

DE CREATIEVE STEM

Datatypes - In Diepte

DE CREATIEVE STEM

Variables : non Typed!

- een container die wordt gebruikt om waarden in een programma op te slaan
- het is niet nodig om variabelen te declareren (is het een geheel getal, een decimaal getal, tekst?) voordat je ze gebruikt
- De Python Interpreter identificeert het type variabele en wijst ruimte toe tijdens runtime

```
# Python variable Declaration
x=25
y=25
print("x+y=",x+y)
sum=x+y
print("sum=",sum)
```

- In Notebook, doe deze code en voer uit: **`type(sum)`**.

  Wijzig x in 25.8. Resultaat? Wijzig x in 'a'. Resultaat ?

DE CREATIEVE STEM

Regels voor het benoemen van variabelen

De variabelenaam kan tekens (zowel a-z als A-Z), onderstrepingstekens en cijfers (0-9) bevatten. Bij het benoemen van een variabele moeten we de volgende punten volgen:

- variabelenaam kan alleen beginnen met een willekeurig teken of onderstrepingsteken. GEEN nummer!
- Een variabelenaam kan cijfers bevatten.
- Een variabele mag geen trefwoord zijn:

import	while	elif	else	break
continue	false	assert	for	class
except	def	nonlocal	in	from
global	none	return	not	is
lambda	raise	yield	true	or
pass	with	finally	and	try

- het kan gelijk welke lengte hebben

DE CREATIEVE STEM

Waarde toekennen

= wordt gebruikt om een waarde aan een variabele toe te kennen.

Multi-assign is toegelaten. Probeer:

```
variable1 = variable2 = variable3 = 13

variable1, variable2, variable3 = 13, 'Python', 40.9
```

DE CREATIEVE STEM

De kracht van Python: Data Types

Het gevarieerde aantal datatypes creëert veel mogelijkheden

Nummer data types:

1. Integer number
2. Float number
3. Complex number

```
# a and b  are integer variables
a = 10
b = 20
# c and d are float variables
c = 10.4
d = 20.0

#complex number example
e = 2 + 3j
print( e.real )
print( e.imag )
```

DE CREATIEVE STEM

Data Types: None

None is een speciaal gegevenstype in Python dat een lege variabele of afwezigheid van waarde vertegenwoordigt

```
[8]  a = None

[9]  type(a)

     NoneType
```

DE CREATIEVE STEM

Data Types: Sequence

Sequence data types (reeksen in het Nederlands) :

1. String
2. List
3. Tuple

Een reeks heeft een index die verwijst naar de elementen. Dit begint met 0!
Een reeks heeft een lengte. Een sequentie-element kan worden verkregen met operator [].
Slicing is mogelijk: sequence[3:5]

A sequence has membership operators: **in** and **not in**

Data Type: String

```
[24] text1 = 'a string'
     text2 = "also a string"
     text3 = ''' multi line
     string like
     this'''
     text4 = 'a string' \
     'continued here'
     text5 = '\t\ta string \n and more \"yes\" '
```

```
[25] test = [print(x) for x in (text1, text2, text3, text4,text5)]
```

```
#Sequence Data Type Example
text = "learn Python on  tutorials Impact"
print ( text )                          # prints whole String
print ( text[ 0 ] )                     # prints character at 0 index
print ( text[ 3 ] )                     # prints character at 3 index
print ( text[ 1 : 10 ] )                # prints range of character from index 1 to 9
print ( text[ -1 ] )                    # prints last element
print ( text [ -13 : -1 ] )             # prints range from -13 to second last element
```

More: https://docs.python.org/3.8/library/stdtypes.html#string-methods

https://docs.python.org/3.8/library/string.html

DE CREATIEVE STEM

Data Type: List

1. List is een reeks waarden.
2. De waarden staan tussen vierkante haken [ ].
3. De waarden worden gescheiden door komma's [1, 2, 3].
4. toegang tot elk item van de lijst met behulp van het indexnummer. Lijsten kunnen zelfs een andere lijst als element bevatten. Bewerkingen kunnen worden uitgevoerd op lijsten: maken, bijwerken, lezen, verwijderen

Data Type: List

```
#  Example of Lists in Python
pythonList = [ 1, 'tutorials', 3, 4.6, 3.7]
pyList = [ 2, 'py']
print ( pythonList )                    # whole list
print ( pythonList[ 0 ] )               # element at index 0
print ( pythonList[ 1 ])                # element at index 1
print ( pythonList[ 1 ][ 0 ])           # element at index 0, in element 1 of list
print ( pythonList[ 1 ][ 0 : 3 ])       # element from index 0 to 2,  in element 1 of list
print ( pythonList[ 0 : 3 ])            # element from index 0 to 2 of list
print ( pythonList[ -1 ] )              # last element of list
print ( pyList * 2)                     # prints twice
print ( pyList + pythonList)            # prints combination of both lists
```

More: https://docs.python.org/3.8/library/stdtypes.html#lists
https://docs.python.org/3.8/tutorial/datastructures.html

DE CREATIEVE STEM

Data Type: Tuple

Tuple is precies hetzelfde als de list.

MAAR: we kunnen de elementen van de tuple niet veranderen.

Kortom, wanneer een tuple eenmaal wordt gedeclareerd, wordt deze alleen-lezen. We kunnen de tuple niet updaten.

Declaration: gebruik ronde haakjes: ( )

Data Type: Tuple

```
# Example of Tuple in Python
pythonTuple = ( 1, 'Lets', 'start', 'python', 3.7)
pyTuple = ( 2, 'py')
print ( pythonTuple)                    # whole tuple
print ( pythonTuple [ 0 ] )             # element at index 0
print ( pythonTuple [ 1 ])              # element at index 1
print ( pythonTuple [ 1 ][ 0 ])         # element at index 0, in element 1 of tuple
print ( pythonTuple [ 1 ][ 0 : 3 ])     # element from index 0 to 2, in element 1 of tuple
print ( pythonTuple [ 0 : 3 ])          # element from index 0 to 2 of tuple
print ( pythonTuple [ -1 ] )            # last element of tuple
print ( pyTuple * 2)                    # prints twice
print ( pyTuple + pythonTuple)          # prints combination of both tuples
```

DE CREATIEVE STEM

Data Types: Set (Verzameling)

- Sets is een **verzameling** waarin we waarden van verschillende gegevenstypen kunnen opslaan.
- Het slaat gegevens op in **niet-sequentiële volgorde** en we hebben geen toegang tot de gegevenselementen door te indexeren.
- Sets worden aangegeven tussen accolades { }.
- Een ander bijzonder ding in sets is dat **dubbele vermeldingen niet** zijn toegestaan. Het bevat alleen unieke gegevensitems.

```
# Example of Sets in Python
pyset = { 1, 2, "impact", 3, "python"}
print( pyset )                          # prints whole set
```

DE CREATIEVE STEM

Data Types: Dictionary (Woordenboek)

- Key, Value paren
- Toevoegen key: `mijndict[KEY] = VALUE`
- De sleutels in het woordenboek moeten uniek zijn

```
# Example of Dictionary
pythonDict = { 'my' : 1, 13: 'dictionary' }
print ( pythonDict[ 'my' ] )            # prints value in 'my' key
print ( pythonDict[ 13 ])               # prints values in 13 key
print ( pythonDict.keys() )             # prints all keys
print ( pythonDict.values() )           # prints all the values
```

```
# Strategy:  Iterate over a copy
for user, status in users.copy().items():
    if status == 'inactive':
        del users[user]

# Strategy:  Create a new collection
active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status
```

DE CREATIEVE STEM

Oefening

DE CREATIEVE STEM

Samenvoegen Getal woordenboek

Schrijf een Python-programma om twee woordenboekwaarden te combineren voor gemeenschappelijke sleutels

test1 = {'leerling01': 100, 'leerling02': 200, 'leerling03': 300}
test2 = {'leerling01': 300, 'leerling02': 200, 'leerling04': 400}

Voorbeeld output:

Resultaat = {'leerling01': 400, 'leerling02': 400, 'leerling04': 400, 'leerling03': 300}

Gevorderden: Collections

DE CREATIEVE STEM

Collections

Zie: https://docs.python.org/3/library/collections.html

Deze module implementeert gespecialiseerde containerdatatypes die alternatieven bieden voor Python's ingebouwde containers voor algemeen gebruik.

- namedtuple: Point = namedtuple('Point', ['x', 'y'])
- Counter: Dict met nummers om te tellen
- OrderedDict: geordende versie van dict
- defaultdict: waarde voor niet aanwezige elementen

List Comprehension

DE CREATIEVE STEM

List Comprehension

Een manier om lijsten te maken in een eenvoudige syntax:

[expression for item in list]

```
h_letters = [ letter for letter in 'Mijn Naam' ]
print( h_letters)
```

Herdoe oefening, vervang for met list comprehension:

Maak een lijst van honderd willekeurig gekozen ASCII letters.

DE CREATIEVE STEM

List Comprehension: condities

Je kan list comprehension uitbreiden met voorwaarden op de lijst die je maakt:

```
[ x for x in range(20) if x % 2 == 0]
```

Meerdere condities:

```
[y for y in range(100) if y % 2 == 0 if y % 5 == 0]
```

Met if en else constructie:

```
["Even" if i%2==0 else "Odd" for i in range(10)]
```

OPGELET: Zorg dat de code leesbaar blijft, for loop is ook goed!

DE CREATIEVE STEM

Lambda functie en map/filter

DE CREATIEVE STEM

Lambda functie en map

lambda functie is een functie maken zonder er een naam aan te geven, omdat je iets snel wil doen:

lambda arguments: expression

Ken zo een functie toe aan een variabele. Vb:

```
verdubbel = lambda x: x * 2
print(verdubbel(5), verdubbel(30))
```

Vermijd lambda als niet nodig, gebruik def !

DE CREATIEVE STEM

map

Lambda functies zul je vaak zien samen met de map functie. De map() functie retourneert een map object (wat een iterator is) van de resultaten na het toepassen van de gegeven functie op elk item van een gegeven iterabel (lijst, tuple etc.).

```
mijn_lijst = [1, 5, 4, 6, 8, 11, 3, 12]
resultaat = list(map(lambda x: x * 2 , mijn_lijst))
print(resultaat)
```

DE CREATIEVE STEM

filter

De functie filter() in Python neemt een functie en een lijst als argumenten op.
De functie wordt aangeroepen met alle items in de lijst en er wordt een nieuwe lijst geretourneerd die items bevat waarvoor de functie **True** evalueert.

```
mijn_lijst = [1, 5, 4, 6, 8, 11, 3, 12]
resultaat = list(filter(lambda x: (x%2 == 0), mijn_lijst))
print(resultaat)
```

DE CREATIEVE STEM