



BBC micro:bit mit Python

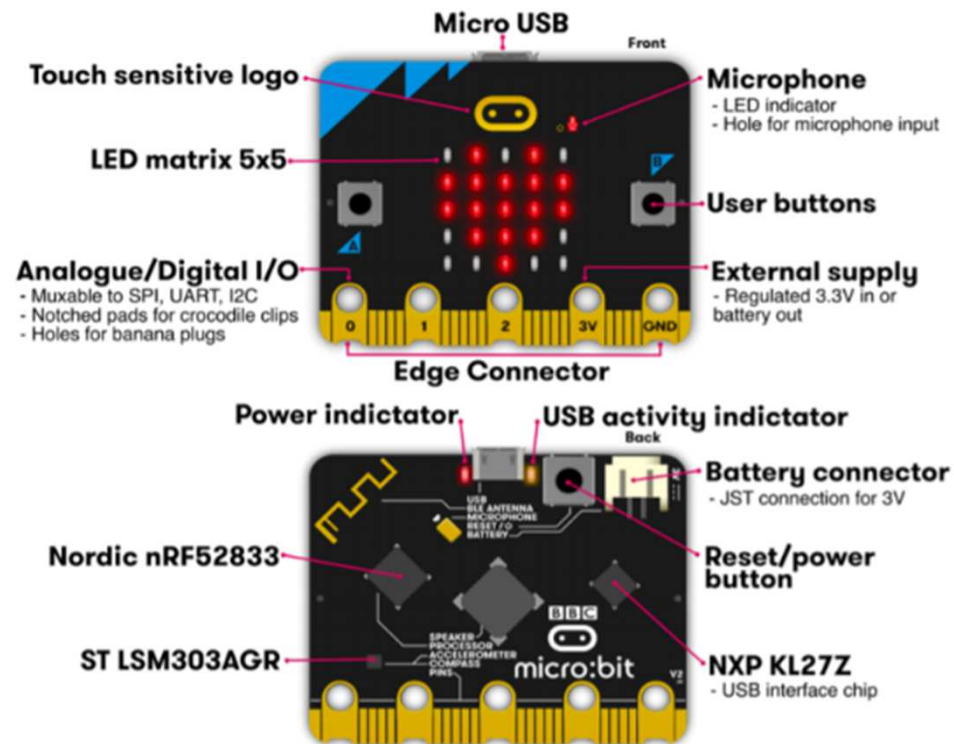
PADERBORN IST
INFORMATIK



Vorstellung des micro:bit

- Einplatinencomputer
- Entwickelt von der BBC zur Verbesserung der Schulbildung in England
- Bisher > 1M Geräte wurden kostenlos an 11 und 12 jährige Schüler verteilt
- Inzwischen gibt es die 2. Version mit
 - Mikrophon, Lautsprecher
 - 5x5 LED-Matrix zur Anzeige von blinkenden Texten, alphanumerische Zeichen und Muster
 - Zwei programmierbare Tasten zur Verwendung als Spiele-Controller, Steuerung der Musik auf eigenem Smartphone und vieles mehr
 - Eingebauter Kompass zum Erfassen Ihrer Bewegungen
 - Temperaturmessung
 - Integriertes 3D-Magnetometer zum Erkennen bestimmter Metalle und Magneten
 - Bluetooth-Technologie
 - Fünf Ein- und Ausgänge zur Steuerung von Motoren, Roboter usw.
 - Berührungsempfindliches Logo
 - Eingebauter Sleep-/Aus-Modus

Vorstellung des micro:bit



Programmierung

micro:bit Get started Projects Lessons Let's code

Let's code

Share

Quick links

New to coding or new to micro:bit

Text-based programming, widely used in education

Manage whole class micro:bit coding sessions

MakeCode editor

Python editor

micro:bit classroom

- **Microsoft MakeCode**
- Python
- Mobile and tablet apps
- Scratch
- Swift Playgrounds
- Other editors

Microsoft MakeCode

Microsoft's MakeCode editor is the perfect way to start programming and get creating with the BBC micro:bit. The colour-coded blocks are familiar to anyone who's previously used Scratch, and yet powerful enough to access all the [features of this tiny computer](#). You can also switch to JavaScript to see the text-based code behind the blocks.

Our [getting started pages](#) will guide you through your first steps.

You can find out more about MakeCode in the [FAQ](#) and as well as using it in your web browser, MakeCode for micro:bit is also available as a free [Windows 10 app](#).

[MakeCode reference](#) **Go to MakeCode editor**

Was ist Python?

- Python gibt es seit Anfang der 1990er Jahre - <https://www.python.org/>
- Python ist eine – zumeist – interpretierte Programmiersprache (oft auch als „Skriptsprache“ bezeichnet)
- Erfunden wurde die Sprache von Guido van Rossum
- Der Name geht auf die englische Komikertruppe „Monty Python“ zurück – und nicht auf die gleichnamige Schlange
- Python ist im Web frei für (fast) alle Systeme verfügbar
- Python ist Open Source und wird von der Python Software Foundation weiterentwickelt

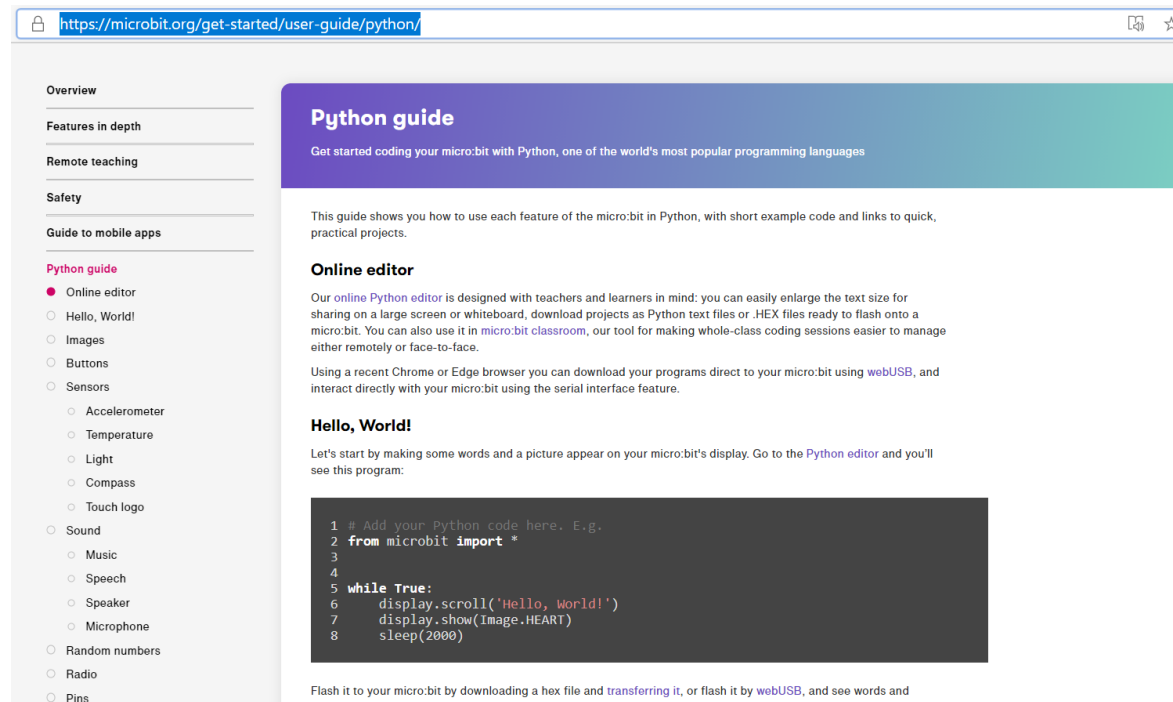


Was ist Python?

- Python unterstützt eine objektorientierte Programmierung (wie bei Java) und auch eine dynamische Typisierung (wie bei JavaScript)
- Blöcke durch Einrückung und nicht durch Klammern
- Die gemischte Verwendung von Leerzeichen und Tabulator-Einrückungen wird zu Problemen führen
- Stark erweiterbar durch eine Vielzahl an Bibliotheken
- Python wird in der Regel für „Backend-Programmierung“ verwendet und nicht für Apps oder Browser-Applikationen

micro:bit Programmierung mit Python

- <https://microbit.org/get-started/user-guide/python/>



micro:bit Programmierung mit Python

- <https://microbit-micropython.readthedocs.io/en/v1.0.1/>



BBC micro:bit MicroPython v1.0.1

Search docs

TUTORIALS

- Introduction
- Hello, World!
- Images
- Buttons
- Input/Output
- Music
- Random
- Movement
- Gestures
- Direction
- Storage
- Speech
- Network
- Radio
- Next Steps

API REFERENCE

- micro:bit Micropython API
- Microbit Module

Docs » BBC micro:bit MicroPython documentation

Edit on GitHub

BBC micro:bit MicroPython documentation

Welcome!

The BBC micro:bit is a small computing device for children. One of the languages it understands is the popular Python programming language. The version of Python that runs on the BBC micro:bit is called MicroPython.

This documentation includes lessons for teachers and API documentation for developers (check out the index on the left). We hope you enjoy developing for the BBC micro:bit using MicroPython.

If you're a new programmer, teacher or unsure where to start, begin with the tutorials.

First Steps with MicroPython by Mike Rowbitt

MicroPython was created by Damien...

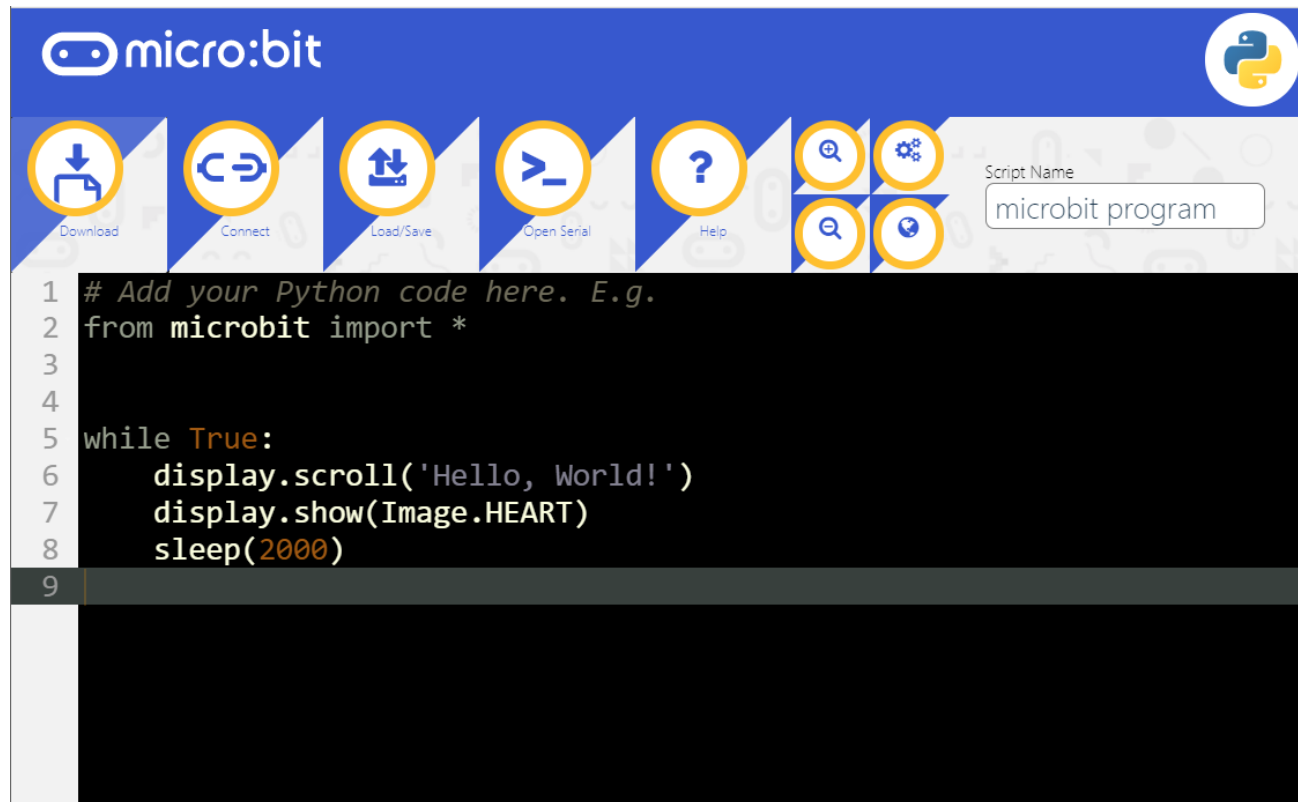
It works with the BBC micro:bit.

```
from microbit import *  
# Edit your code here!  
display.scroll("Hello, world!")
```

Everything you need to know about MicroPython on the BBC micro:bit is found in this documentation.

Generated by Python Comics. MAKE YOUR OWN

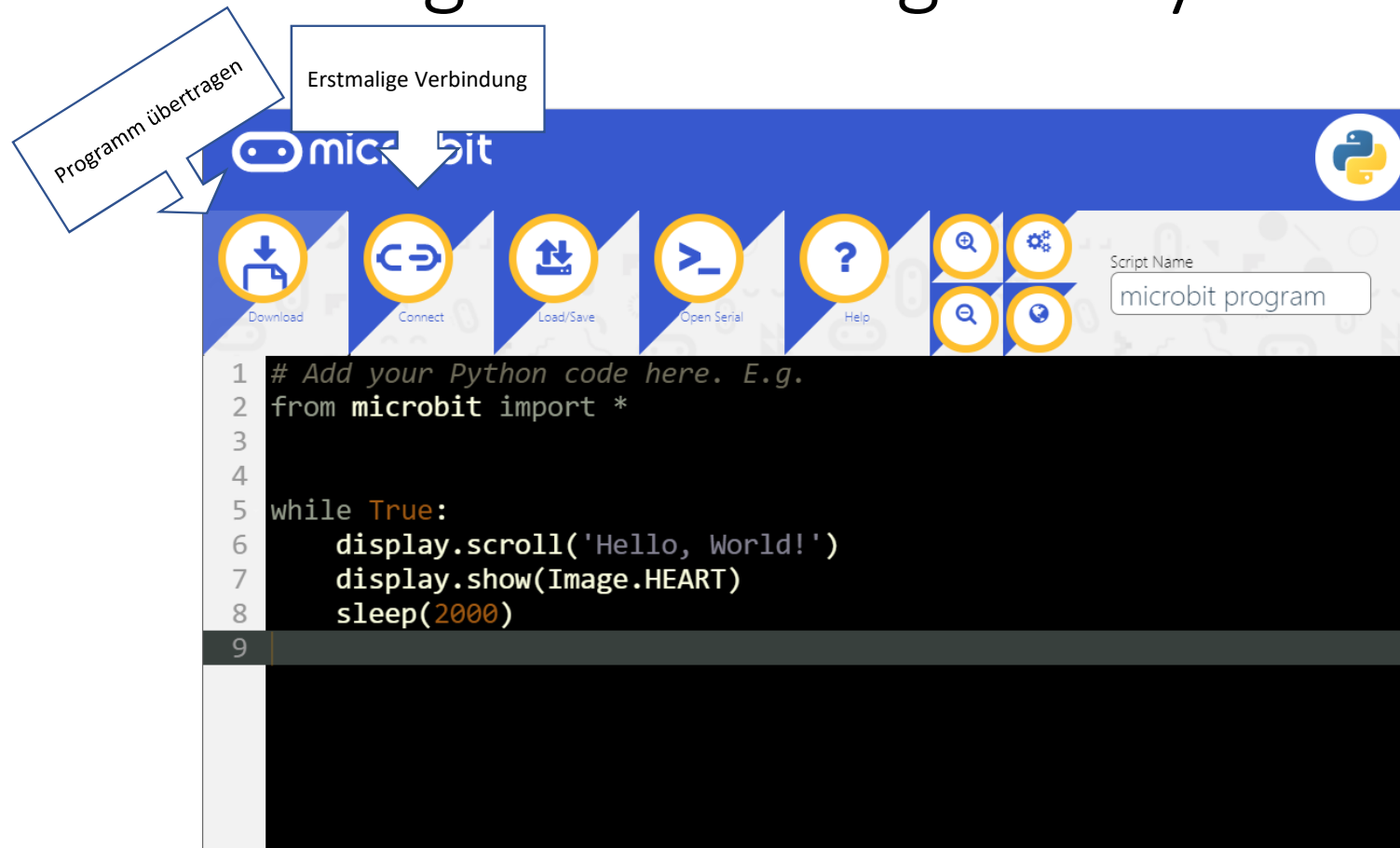
micro:bit Programmierung mit Python



micro:bit Programmierung mit Python

- Link
 - <https://python.microbit.org/v/2>
- micro:bit mit dem Rechner verbinden
 - USB-Kabel
 - Strom- und Datenversorgung
- Programme können direkt aus dem Editor auf das Gerät übertragen werden
 - Sie werden dann mit micropython ausgeführt
 - micropython ist bereits auf dem micro:bit installiert
- Auch ohne Stromversorgung bleibt das letzte Programm auf dem micro:bit

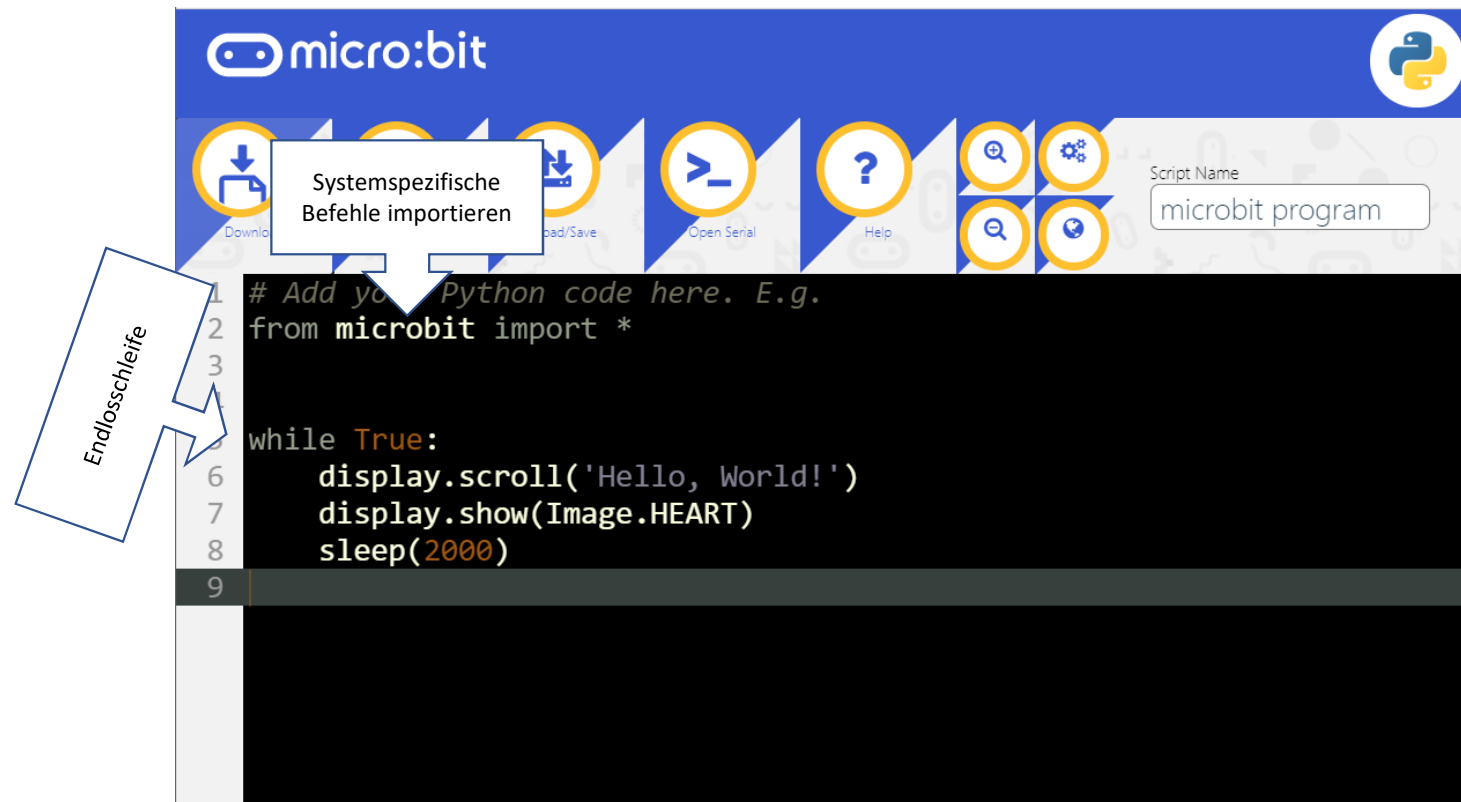
micro:bit Programmierung mit Python



The screenshot shows the micro:bit Python programming environment. At the top, there's a blue header with the 'micro:bit' logo and a Python logo. Below the header is a toolbar with icons for Download, Connect, Load/Save, Open Serial, Help, and a 2x2 grid of icons for Search, Settings, Find, and Run. To the right of the toolbar is a 'Script Name' input field containing 'microbit program'. Two callout boxes are present: one pointing to the 'Download' icon labeled 'Programm übertragen' and another pointing to the 'Connect' icon labeled 'Erstmalige Verbindung'. The main area is a code editor with a dark background and light text. It contains the following Python code:

```
1 # Add your Python code here. E.g.
2 from microbit import *
3
4
5 while True:
6     display.scroll('Hello, World!')
7     display.show(Image.HEART)
8     sleep(2000)
9
```

micro:bit Programmierung mit Python

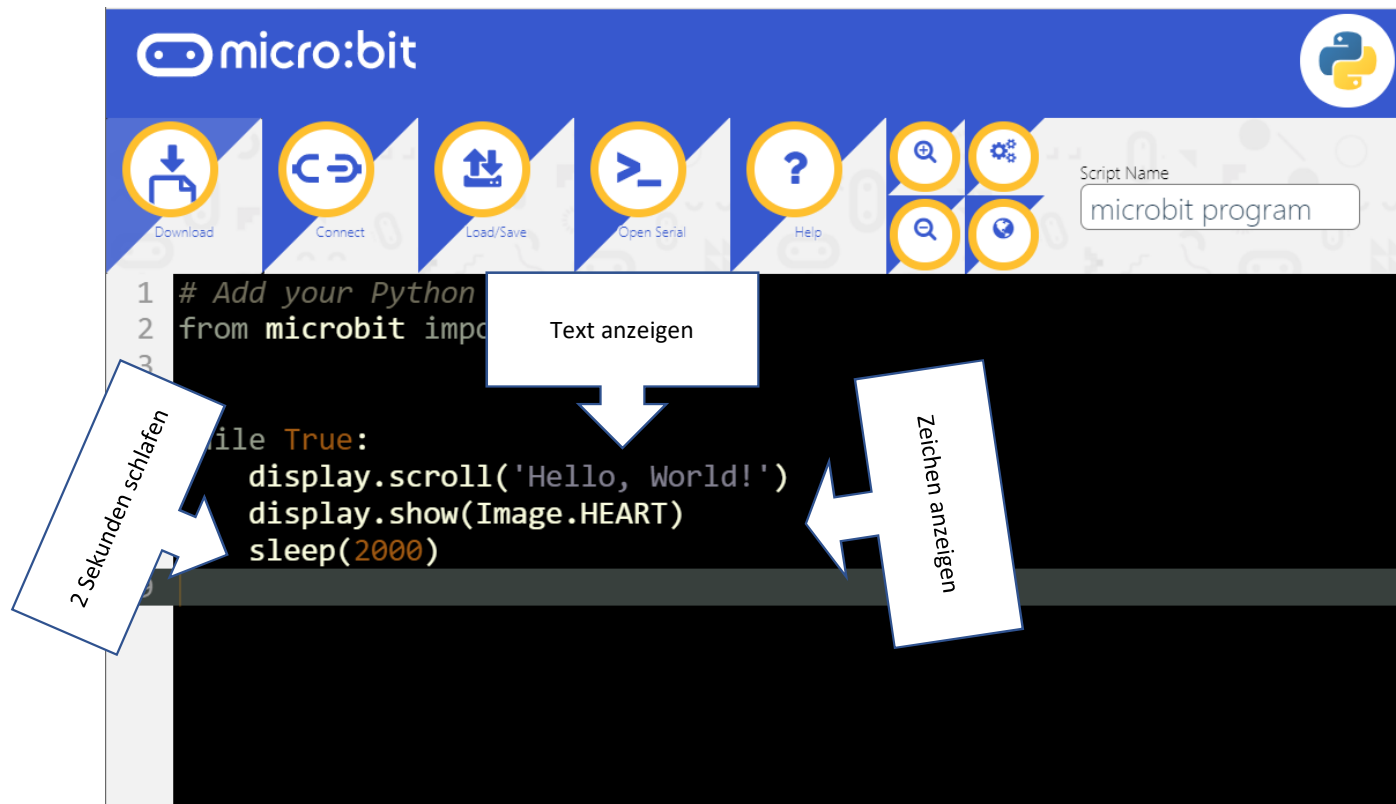


The image shows the micro:bit Python IDE interface. At the top, there's a blue header with the 'micro:bit' logo and a Python logo. Below the header is a toolbar with icons for downloading, uploading, opening serial, and help. A callout box points to the download/upload icons with the text 'Systemspezifische Befehle importieren'. To the right of the toolbar is a 'Script Name' input field containing 'microbit program'. The main area is a code editor with a black background and white text. It contains the following Python code:

```
1 # Add your Python code here. E.g.  
2 from microbit import *  
3  
4  
5 while True:  
6     display.scroll('Hello, World!')  
7     display.show(Image.HEART)  
8     sleep(2000)  
9
```

A callout box points to the 'while True:' loop with the text 'Endlosschleife'.

micro:bit Programmierung mit Python



micro:bit Programmierung mit Python

- Zusätzliche Module importieren (um deren Funktionen zu nutzen)

```
# Man kann alles aus dem Modul importieren oder nur das was man benötigt  
from microbit import *
```

- Endlosschleife

```
While True:  
    # Tue irgendwas
```

- Einrückungen

- Python kennt keine {} um Blöcke zu kennzeichnen (wie bei Java)
- Ein „:“ schliesst die vorherige Zeile ab
- Einrückungen (**Tab** oder 4 Leerzeichen – aber niemals (!) gemischt)

micro:bit Programmierung mit Python

- Auf der Anzeige etwas anzeigen

```
# So lassen sich Texte auf der kleinen Anzeige darstellen  
display.scroll('Hello')
```

```
# So können einzelne - vordefinierte - Grafiken angezeigt werden  
display.show(Image.HEART)
```

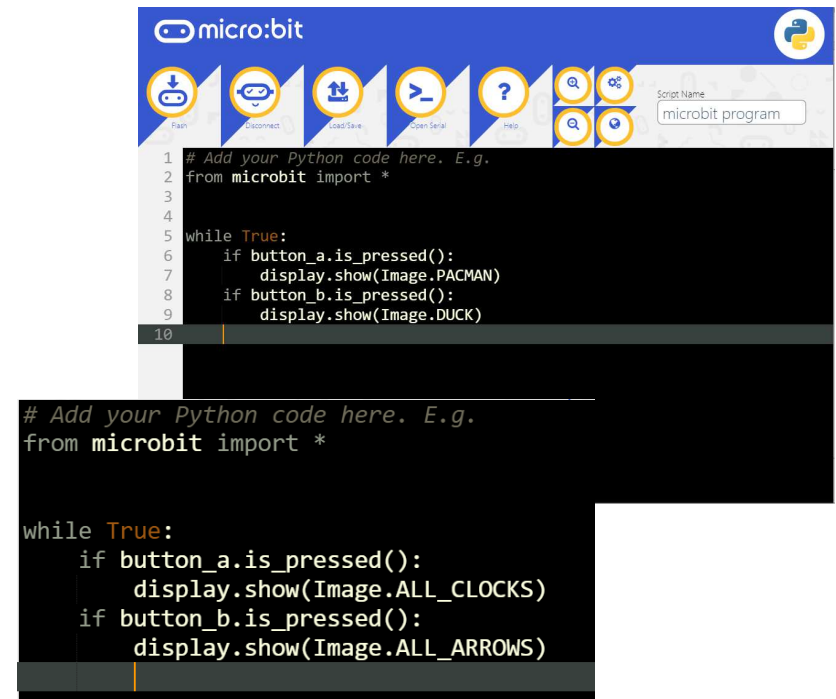
- Weitere Möglichkeiten => Siehe Doku!

micro:bit Programmierung mit Python

- Was kann man alles programmieren?
 - Bilder
 - Knöpfe
 - Sensoren
 - Beschleunigung
 - Temperatur
 - Licht
 - Kompass
 - Logo
 - Tonausgabe
 - Radio – Datenaustausch zwischen micro:bits

micro:bit Programmierung mit Python

- Knöpfe
 - a / b – links / rechts
 - `is_pressed()` – gedrückt
 - True / False
 - `get_presses()` – Anzahl
- Bilder anzeigen
 - Vorgefertigte Bilder
 - Listen (mehrere Bilder)
 - Selbstdefinierte Bilder



The image shows a screenshot of the micro:bit Python IDE. The top bar is blue with the 'micro:bit' logo and a Python logo. Below the bar are several icons: a download icon, a disconnect icon, a load/save icon, a run icon, a help icon, and a search icon. To the right of these icons is a text input field labeled 'Script Name:' with the text 'microbit program' entered. Below the icons is a code editor with a black background and white text. The code in the editor is as follows:

```
1 # Add your Python code here. E.g.
2 from microbit import *
3
4
5 while True:
6     if button_a.is_pressed():
7         display.show(Image.PACMAN)
8     if button_b.is_pressed():
9         display.show(Image.DUCK)
10
```

Below the first code editor is a second code editor, also with a black background and white text. The code in this editor is as follows:

```
# Add your Python code here. E.g.
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.ALL_CLOCKS)
    if button_b.is_pressed():
        display.show(Image.ALL_ARROWS)
```

Übung

- Erstelle eine Liste aus 5 (oder mehr) Grafiken
- Der User kann durch Knopfdruck (rechts/links) durch die Grafiken blättern

```
• Image.HEART , Image.HEART_SMALL
• Image.HAPPY , Image.SMILE , Image.SAD , Image.CONFUSED , Image.ANGRY , Image.ASLEEP ,
  Image.SURPRISED , Image.SILLY , Image.FABULOUS , Image.MEH , Image.YES , Image.NO
• Image.ARROW_N , Image.ARROW_NE , Image.ARROW_E , Image.ARROW_SE , Image.ARROW_S ,
  Image.ARROW_SW , Image.ARROW_W , Image.ARROW_NW
• Image.MUSIC_CROCHET , Image.MUSIC_QUAVER , Image.MUSIC_QUAVERS
• Image.XMAS , Image.PACMAN , Image.TARGET , Image.ROLLERSKATE , Image.STICKFIGURE , Image.GHOST ,
  Image.SWORD , Image.UMBRELLA
• Image.RABBIT , Image.COW , Image.DUCK , Image.HOUSE , Image.TORTOISE , Image.BUTTERFLY ,
  Image.GIRAFFE , Image.SNAKE
```

Gestures

- Vereinfachte Nutzung des Beschleunigungssensors

```
while True:
    gesture = accelerometer.current_gesture()
    if gesture == "face up":
        display.show(Image.HAPPY)
    else:
        display.show(Image.ANGRY)
```

MicroPython is able to recognise the following gestures: `up`, `down`, `left`, `right`, `face up`, `face down`, `freefall`, `3g`, `6g`, `8g`, `shake`. Gestures are always represented as strings. While most of the names should be obvious, the `3g`, `6g` and `8g` gestures apply when the device encounters these levels of g-force (like when an astronaut is launched into space).

Übung – Magic 8-Ball

- Erstellt eine Liste an „Vorhersagen“
 - Alles wird gut
 - Es könnte klappen
 - Usw.
- Durch das Schütteln des Micro:Bit wird eine Antwort aus der Liste zufällig ausgewählt und angezeigt

micro:bit Programmierung mit Python

- Selbstdefinierte Bilder

```
# Add your Python code here. E.g.  
from microbit import *  
  
xmas_1 = Image("00000:00500:05550:55555:00500")  
xmas_2 = Image("00900:00500:05550:55555:00500")  
  
while True:  
    display.show(xmas_1)  
    sleep(1000)  
    display.show(xmas_2)  
    sleep(1000)
```

Pixel setzen

- Pixel können einzeln mit einer vorgegebenen Helligkeit (0-9) auf der (5x5) Anzeige gesetzt werden

```
from microbit import *  
  
display.clear()  
for x in range(0, 5):  
    for y in range(0, 5):  
        display.set_pixel(x,y,9)  
        sleep(500)
```

Pixel setzen

- Um gezielt Pixel anzusteuern und auch zu merken (z.B. für Spiele)
- `d = microbit.display` # Zugriff auf die Anzeige
- `img_screen = microbit.Image(5,5)` # Virtuell ein Abbild anlegen
- `img_screen.set_pixel(4, x, 0)` # Pixel gezielt z.B. ausschalten
- `d.show(img_screen)` # Anzeige mit einem Rutsch darstellen
- Ausprobieren.

Etwas komplexer

```
import microbit
import os
import random

d = microbit.display
img_screen = microbit.Image(5,5)
last_col_height = 1
tick = 1

def draw():
    global last_col_height
    for j in range(4):
        for i in range(5):
            img_screen.set_pixel(j, i, img_screen.get_pixel(j+1,i))
        # Fill last column
        for x in range(5):
            img_screen.set_pixel(4, x, 0)
        for x in range(last_col_height):
            img_screen.set_pixel(4, 4-x, 9)

    d.show(img_screen)

    entropy = random.randint(0,9)
    if entropy > 4 and last_col_height < 4:
        last_col_height = last_col_height + 1
    if entropy <=4 and last_col_height > 1:
        last_col_height = last_col_height -1

while True:
    draw()
    microbit.sleep(300)
```


micro:bit Programmierung mit Python

- Temperaturmessung
- Die Temperaturmessung im micro:bit gibt nur einen ungefähren Wert
- Zudem „wärmt“ sich das Gerät auf
- Daher zeigt die Temperatur eher die Temperatur „auf dem Gerät“

```
temperatur_wert = temperature()
```

- **ÜBUNG** – Programm erstellen:
 - Das Programm zeigt stets die aktuelle Temperatur an
 - Wird der linke Knopf gedrückt, wird die tiefste Temperatur angezeigt

```
if button_a.is_pressed():
```
 - Wird der rechte Knopf gedrückt, wird die höchste Temperatur angezeigt

micro:bit Programmierung mit Python

- ÜBUNG - Temperaturmessung

```
# Add your Python code here. E.g.
from microbit import *

low_tmp = high_tmp = temperature()

while True:

    # Aktuelle Temperatur
    current_temperature = temperature()

    # Anzeigen
    display.scroll(current_temperature)

    # Höchst- oder Tiefsttemperatur merken
    if current_temperature > high_tmp:
        high_tmp = current_temperature
    if current_temperature < low_tmp:
        low_tmp = current_temperature

    # Knöpfe gedrückt?
    if button_b.is_pressed():
        display.scroll(high_tmp)
    if button_a.is_pressed():
        display.scroll(low_tmp)
```

micro:bit Programmierung mit Python

- Python auf dem micro:bit ist keine vollwertige Version von Python
 - Nicht alles wird unterstützt
- Es gibt ein eingeschränktes Datei-System
 - Es werden z.B. nicht alle Datei-Modi unterstützt – so geht z.B. ein „Append“ nicht
 - Das Datei-System wird immer dann neu angelegt wenn man eine neue Programmversion draufspielt („Flash“)
 - Dennoch lässt sich das Dateisystem für eigene Programme nutzen

micro:bit Programmierung mit Python

- Datei öffnen und lesen

```
open_file = open('save_temps.dat')  
temps = open_file.read()  
open_file.close()
```

- Datei öffnen und schreiben

```
open_file = open('save_temps.dat', 'w')  
open_file.write('Hello' + ',' + 'World')  
open_file.close()
```

micro:bit Programmierung mit Python

- Sleep / Pause einlegen

```
sleep(Anzahl_Milisekunden) - z.B. sleep(2000)
```

- Methode anlegen / aufrufen

```
def datei_schreiben():  
    #Einrückung nicht vergessen  
    open_file = open(Dateiname, Schreibmodus)  
    # Etwas hinenschreiben
```

```
# Hauptprogramm  
while True:  
    #Einrückung nicht vergessen  
    if bedingung == True:  
        datei_schreiben()
```

micro:bit Programmierung mit Python

- Prüfen ob eine Datei bereits vorhanden ist

```
import os

# Eine Liste der vorhandenen Dateien abfragen
dateien = os.listdir()
# dateien = ['erste_datei', 'zweite_datei' ]

# Durch alle Dateien iterieren
for file in dateien:
    if file == 'save_temps.dat':
        # Datei ist vorhanden
```

micro:bit Programmierung mit Python

- **ÜBUNG** – Programm erstellen:

- Das Programm zeigt stets die aktuelle Temperatur an
- Wird der linke Knopf gedrückt, wird die tiefste Temperatur angezeigt

```
if button_a.is_pressed():
```

- Wird der rechte Knopf gedrückt, wird die höchste Temperatur angezeigt

- Die aktuellen Hoch- und Tiefwerte werden in einer Datei geschrieben und beim Programmstart eingelesen

micro:bit Programmierung mit Python

```
from microbit import *
import os

# Low und High setzen
low_temp = high_temp = temperature()

def write_temps():
    open_file = open('save_temps.dat', 'w')
    open_file.write(str(low_temp)+' '+str(high_temp))
    open_file.close()

# Zuerst checken ob es die Datei schon gibt
dateien = os.listdir()

for file in dateien:
    if file == 'save_temps.dat':
        open_file = open('save_temps.dat')
        temps = open_file.read()
        open_file.close()
        low_temp, high_temp = temps.split(' ')
```

```
while True:
    current_temperature = int(temperature())
    display.scroll(current_temperature)
    if current_temperature < low_temp:
        low_temp = current_temperature
        write_temps()
    if current_temperature > high_temp:
        high_temp = current_temperature
        write_temps()
    if button_b.is_pressed():
        display.scroll(high_temp)
    if button_a.is_pressed():
        display.scroll(low_temp)
    sleep(2000)
```


micro:bit Programmierung mit Python

- Miteinander kommunizieren
 - micro:bits können einander Nachrichten schicken
 - Nachrichten werden immer an alle verschickt (Broadcast)
 - Eine Filterung ist möglich, damit man nur bestimmte Nachrichten bekommt
 - Nachrichten werden als Strings oder ByteArrays verschickt
 - Eine Länge bis 251 Zeichen ist möglich
 - `import radio`
- Einige Befehle
 - `radio.on()`, `radio.off()`
 - `radio.send(String)`, `radio.receive()`, `radio.receive_bytes()`
 - `radio.config(channel=1..83)`, `radio.config(group=1..99)`,
`radio.config(power=0..7)`

Radio ÜBUNG

- PICTURE BROADCAST
- Per Knopfdruck werden „Bilder“ an alle empfangenden Micro:Bits verschickt und entsprechend angezeigt
 - Button Rechts:
 - Standardbilder – zufällig aus einer Liste mit 5 Grafiken ausgewählt (z.B. String `herz` = Image.Heart) – gleiche Liste verwenden!
 - Button Links:
 - Eigenes Bild - MY_IMAGE = '09090:99999:99999:09990:00900' (voher entwerfen und testen!)
 - MY_IMAGE = '09090:99999:99999:09990:00900'
 - display.show(Image(MY_IMAGE))
 - display.clear()
 - Einigt euch auf ein Sonderzeichen zu Beginn des Strings – damit können dann eigene Bilder einfacher erkannt werden. Das Zeichen muss dann wieder abgeschnitten werden.
 - newString = receivedString[1:]
 - Nach dem Empfang muss dann für die Anzeige unterschieden werden
 - Standardbild
 - Eigenes Bild

Ideen für das eigene Projekt

- Spiele
 - Snake (Schlange und frisst zufällig platzierte ,Äpfel' und wird dabei länger)
 - Space-Shooter (Asteroiden ausweichen)
 - Blackjack – mit Kartenanzeige
 - Tic Tac Toe mit einem Gegner
- Anwendungen
 - Lautstärke aufzeichnen
 - Schrittzähler
 - Erdanziehung messen (Pendel, Video auf YouTube)
- <https://www.microbit.org/projects/>