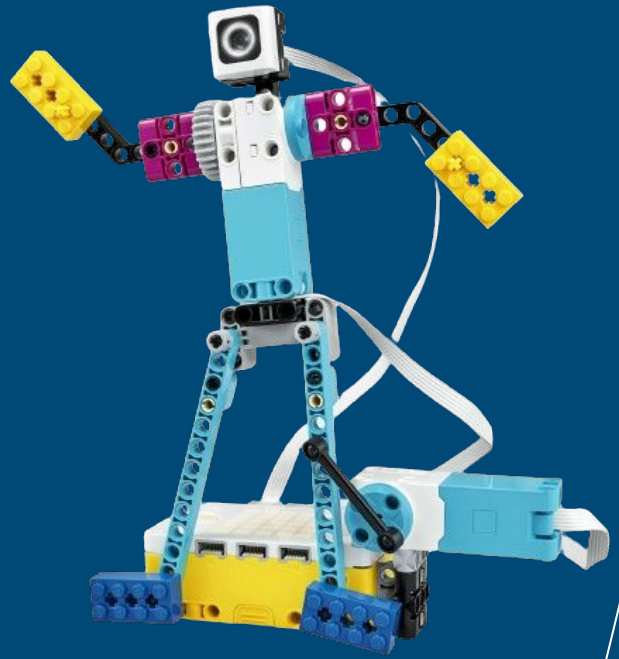


Lego Spike Prime mit Python programmieren

UNIVERSITÄT DES SAARLANDES

SIC Saarland Informatics Campus

Der Roboter / Das Set

Was ist im Lego Spike Prime Set?

Aktoren:

- Motoren
- Display
- Lautsprecher
- LED am Knopf

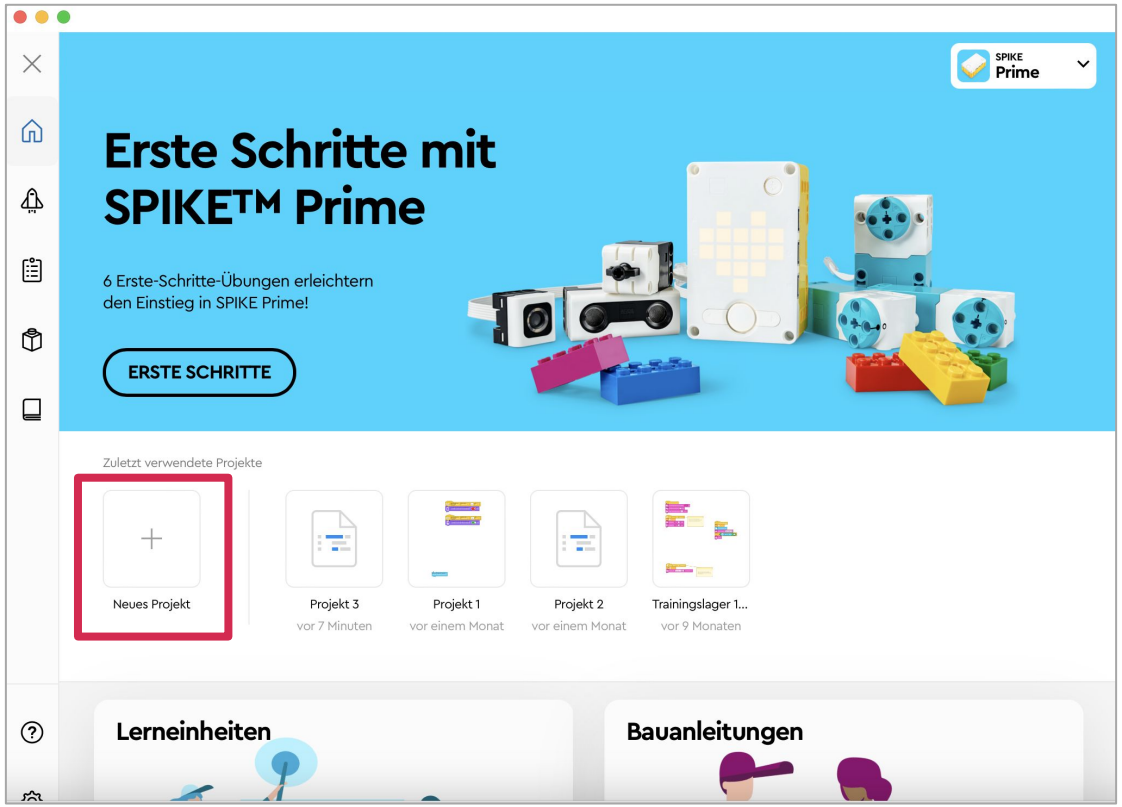
Sensoren:

- Ultraschallsensor
- Knopf/Drucksensor
- Tasten auf dem Hub
- Farbsensor
- Mikrofon
- Gyro-Sensor

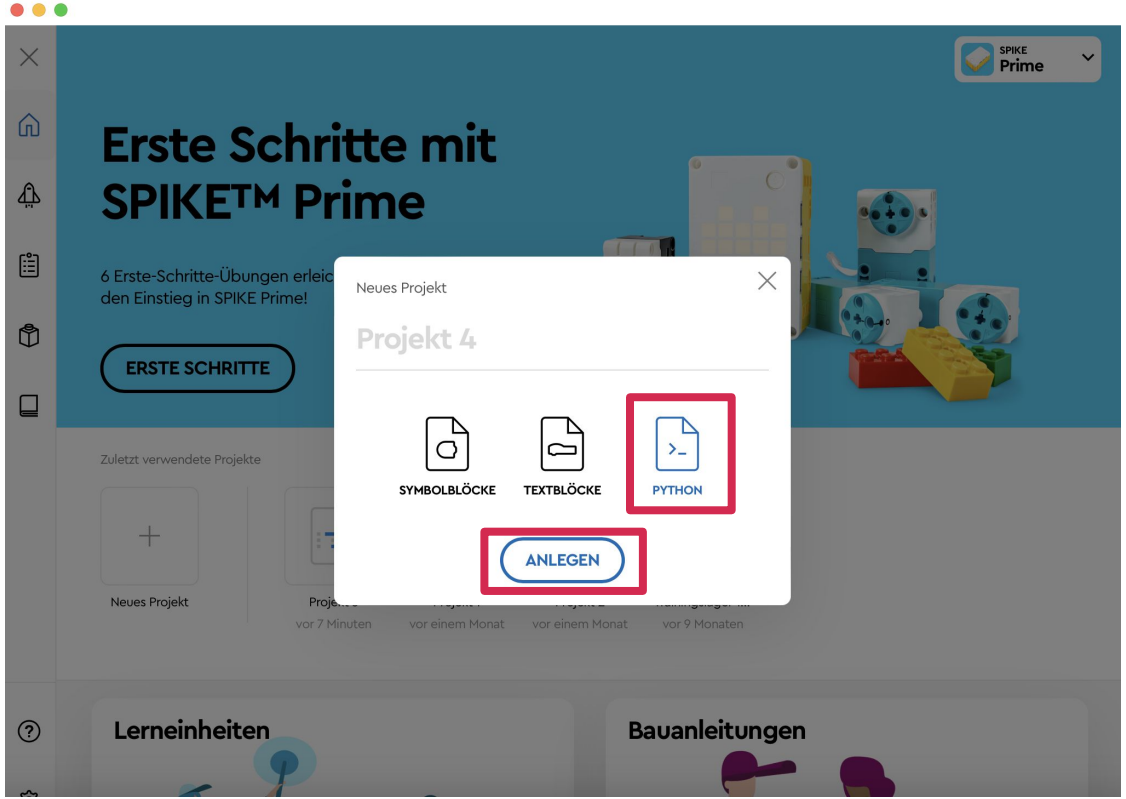
Unterschiede zu Lego Mindstorms Ev3

Die Entwicklungsumgebung

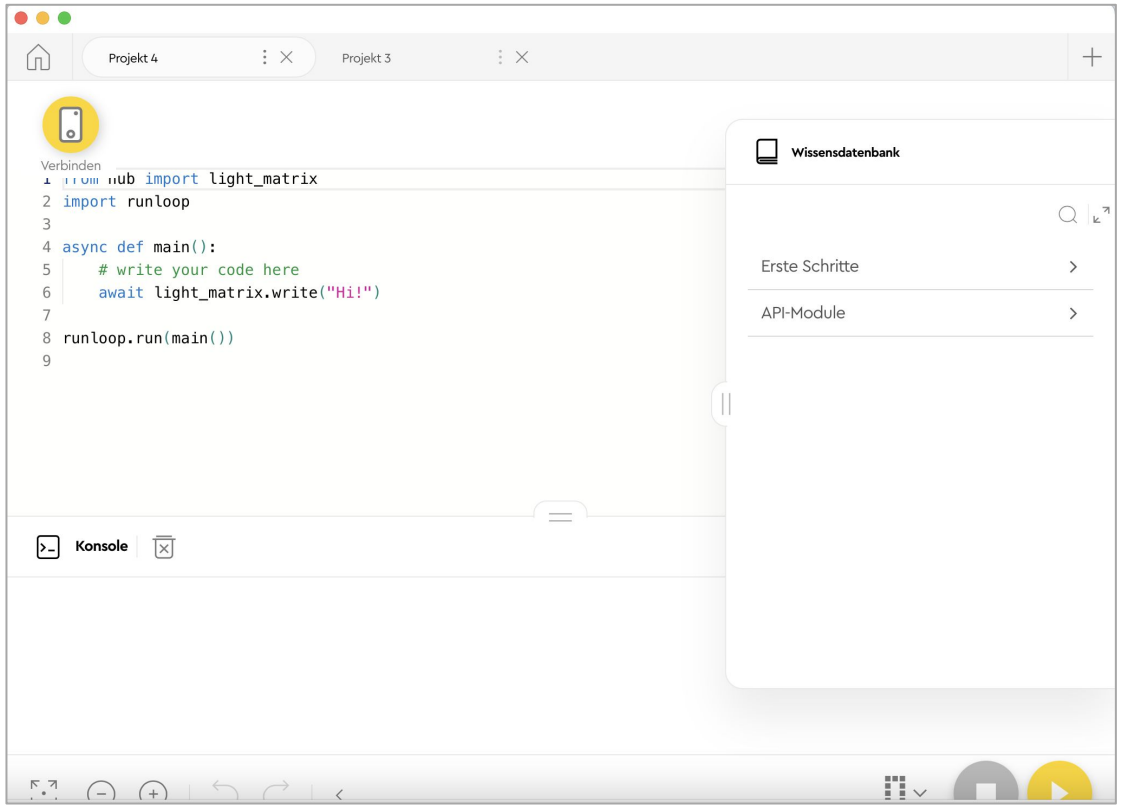
Entwicklungsumgebung mit Python



Entwicklungsumgebung mit Python



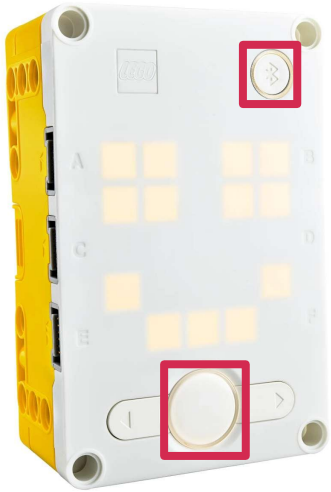
Entwicklungsumgebung mit Python



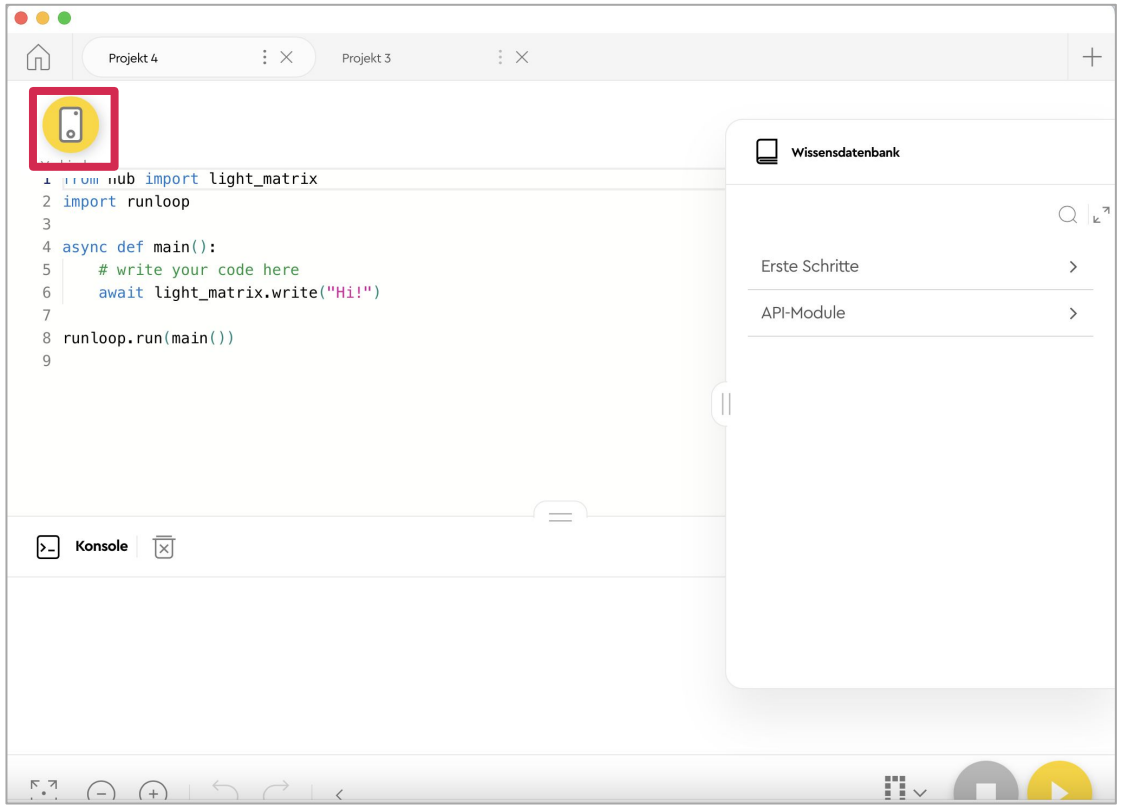
Entwicklungsumgebung - Hub verbinden

3. Hub verbinden

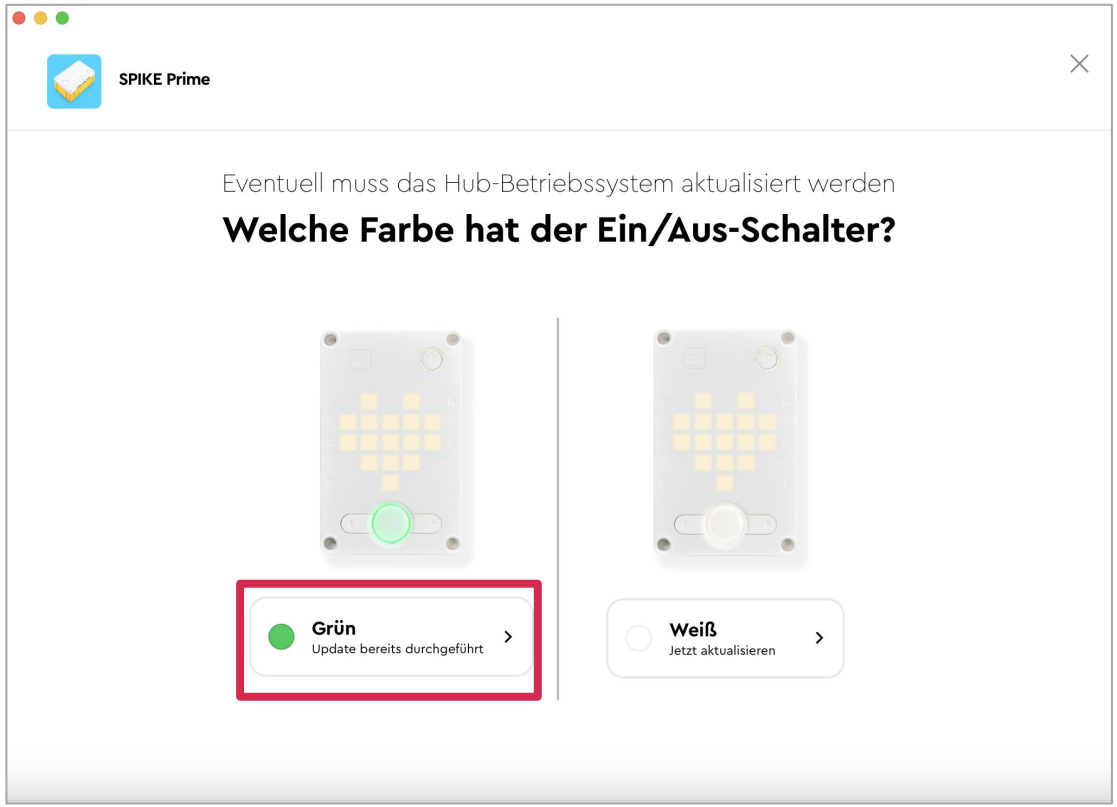
2. Bluetooth



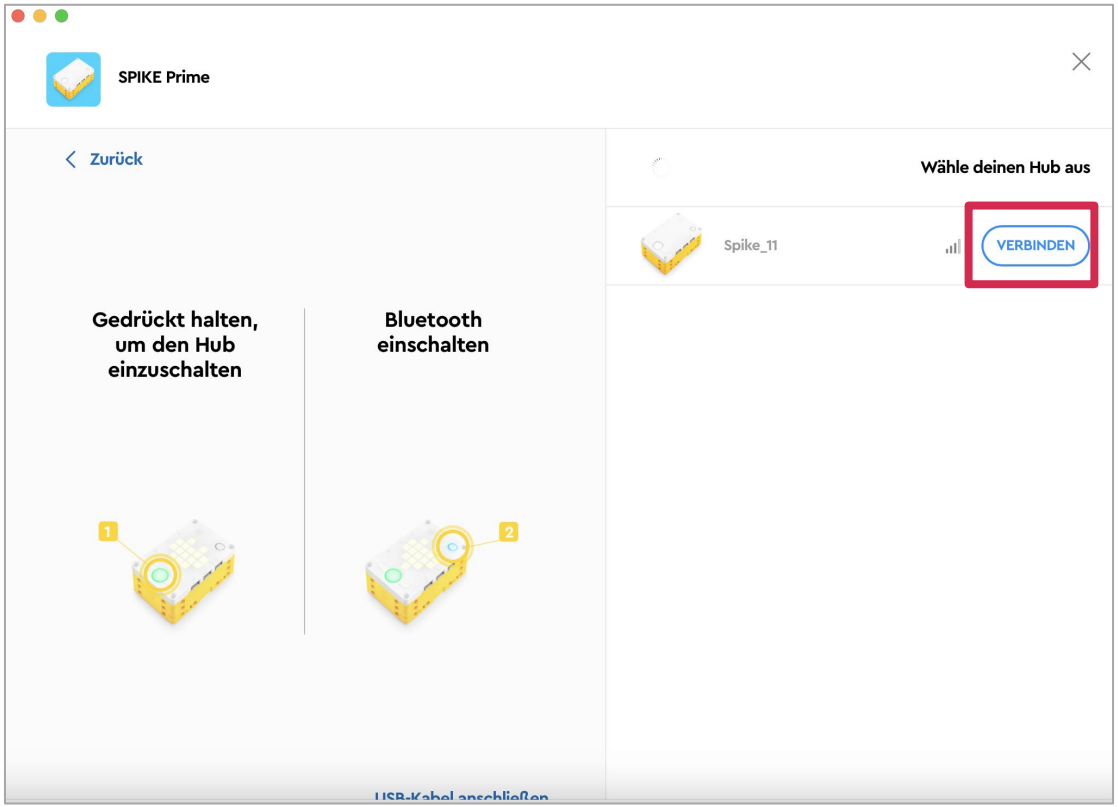
1. Einschalten



Hub verbinden



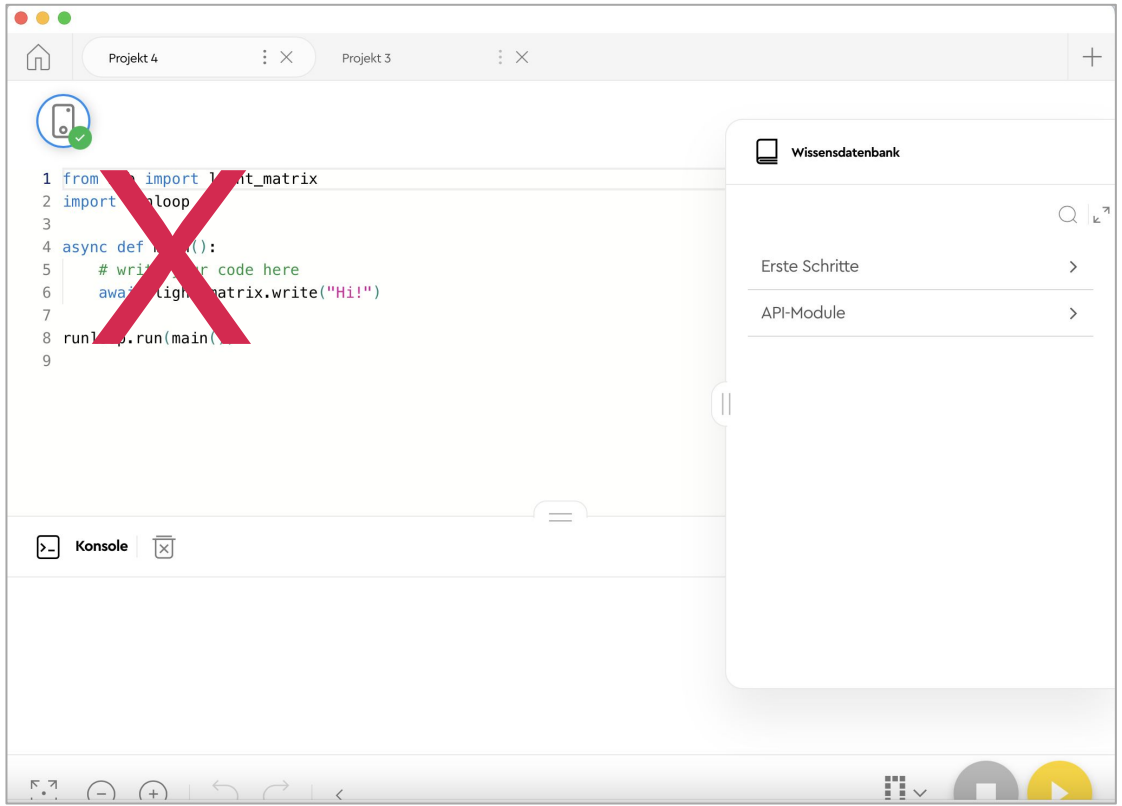
Hub verbinden



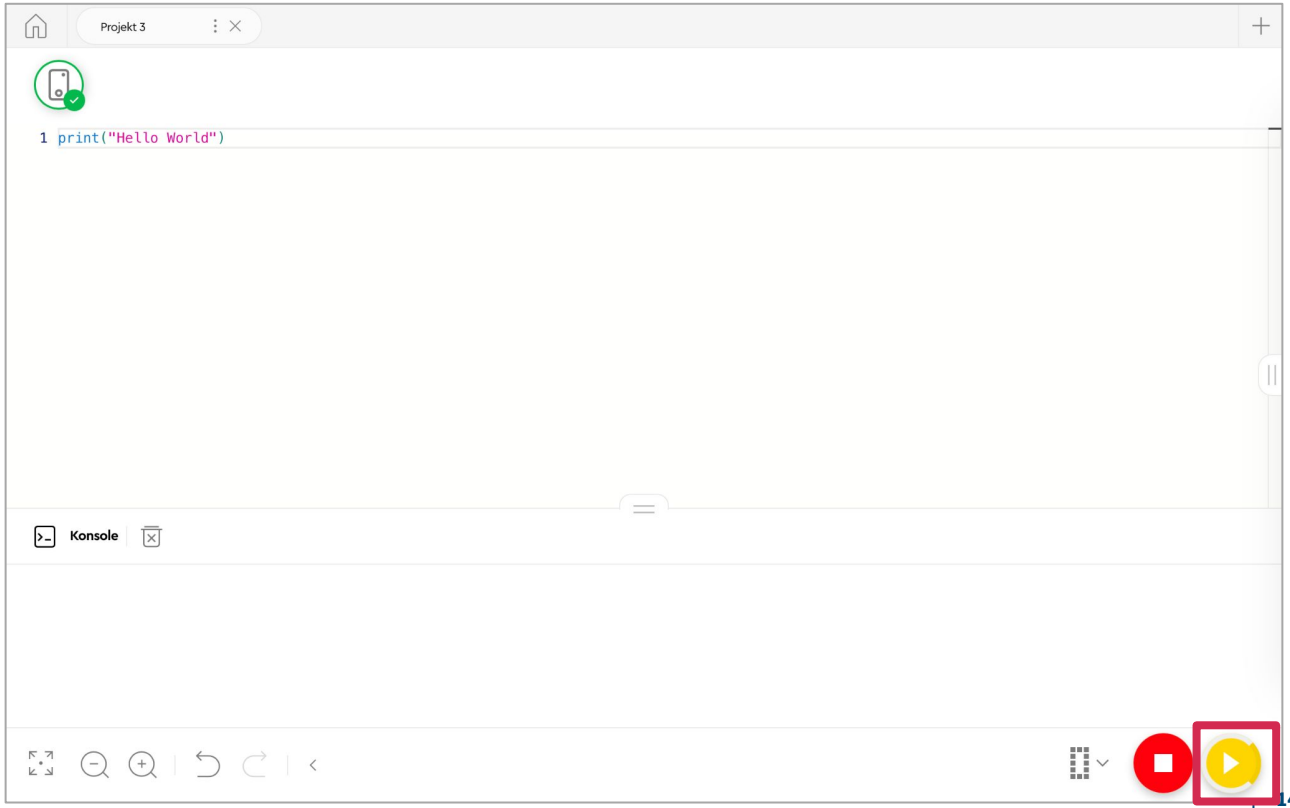
Entwicklungsumgebung mit Python

Vorgegebenen Code löschen.

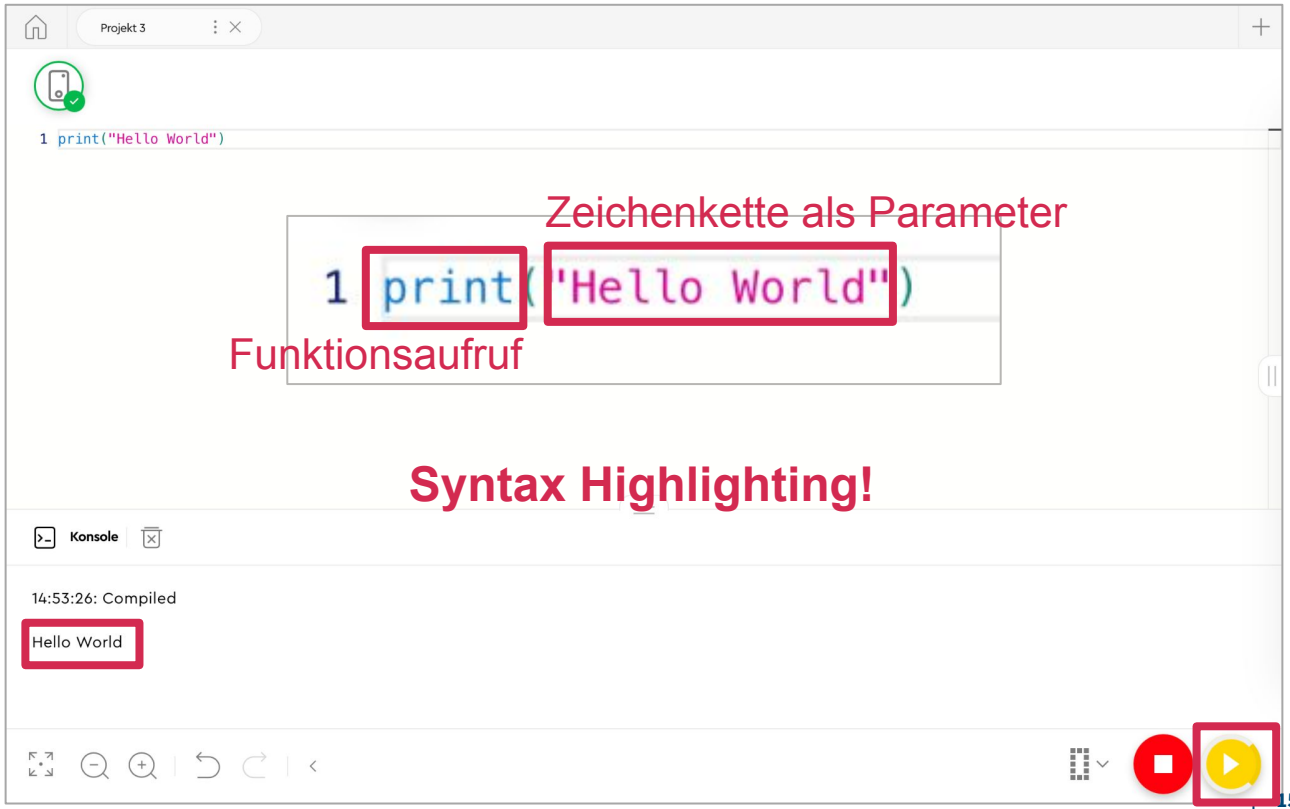
Wir fangen kleiner an.



“Hello World!” - auf der Konsole



“Hello World!” - erklärt



Der Hub

“Hello World!” - auf dem Hub

“Hello World!” - erklärt

Bibliothek (= Library)

Importiertes Modul

Import

ausgeführter Code

```
1 from hub import light_matrix
2
3 light_matrix.write('Hello, World!')
```

Importiertes Modul

Zeichenkette als Parameter

Funktionsaufruf

Smiley auf dem Hub

```
1 from hub import light_matrix
2
3 light_matrix.show_image(light_matrix.IMAGE_HAPPY)
```

Welche Bilder gibt es noch? -> Blick in die Dokumentation

Python

Erste Schritte

Einführung in Python

Python ist eine beliebte textbasierte Programmiersprache, die sich hervorragend für Anfänger eignet, weil sie so knapp und verständlich ist. Außerdem bietet sich Python auch für Programmiererinnen und Programmierer an, weil sich diese Programmiersprache nicht nur für die Web- und Softwareentwicklung, sondern auch für wissenschaftliche Anwendungen eignet, beispielsweise für die Datenanalyse und maschinelles Lernen.

Im Abschnitt **Erste Schritte** werden die Grundlagen erläutert, wie sich Python mit LEGO® Education SPIKE™ Prime nutzen lässt. Der Abschnitt enthält Kapitel, in denen du:

Einführung in Python

1. Lernst, wie du den *Code-Editor* in der LEGO® Education SPIKE™ App benutzt, um Python-Programme zu schreiben.

Hello, World!

2. Eine Mitteilung auf der Lichtmatrix des SPIKE Prime Hubs schreibst.

Kommentare in Python

3. Lernst, wie dir Kommentare helfen, Programm-Entwürfe und fertige Programme zu beschreiben.

Steuerung von Motoren

4. *Asynchrone Funktionen* definierst und startest, um Motoren zu steuern.

Variablen

5. Zwei Motoren mit *lokalen* und *globalen* Variablen steuerst.

Die Macht des Zufalls

6. Möglichkeiten entdeckst, witzige Zufallsprogramme zu schreiben, die das Licht am Hub steuern.

Sensorsteuerung

7. Einen Motor mithilfe des Kraftsensors steuerst. Dann erfährst du, wie du die Konsole zum *Debuggen* deines Programms benutzt.

Sensorbedingungen

8. *Logische Ausdrücke* verwendest, um auf verschiedene Bedingungen zu reagieren. Anschließend lernst du, verschiedene Teile deines Programms zusammen auszuführen, um auf mehrere Bedingungen zu reagieren.

Nächste Schritte

9. Vorschläge für zusätzliche Informationsquellen erhältst, um mehr über die Nutzung von Python mit SPIKE Prime zu erfahren.

Python-Syntax

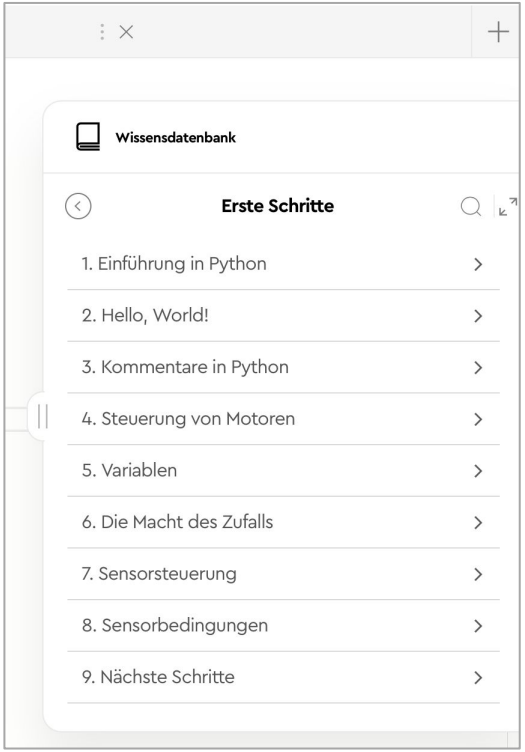
Beim Erlernen einer textbasierten Programmiersprache musst du als Erstes ihre *Syntax* verstehen. Diese Sprachsyntax gibt die Regeln fürs Schreiben von *Anweisungen* (Programmzeilen) vor und definiert, wie *Programmblöcke* kenntlich gemacht werden, die aus mehreren Anweisungen bestehen.

In Python beginnt jede Anweisung mit einer Ebene der *Einrückung* und endet mit einem *Zeilenumbruch*. Als Einrückung wird die Anzahl der Leerzeichen vor einer Anweisung bezeichnet. Programmzeilen mit derselben Anzahl von Leerzeichen besitzen dieselbe *Einrückungsebene* und gehören zum selben Programmblock. Die SPIKE App verwendet 4 Leerzeichen für jede Einrückungsebene.

Du schreibst dein Programm im *Code-Editor*. Dort sind Funktionen vorhanden, die dich dabei unterstützen, das Programm richtig zu schreiben. Wenn du zum Beispiel einen Programmblock anfängst, beispielsweise eine *Funktion* oder eine `if`-Anweisung, dann rückt der Editor die nächste Zeile um 4 Leerzeichen ein. Außerdem wird jede Zeile nummeriert, damit du dich in deinem Programm besser zurechtfindest.

Die *Syntaxhervorhebung* im Code-Editor zeigt *Kommentare*, *Schlüsselwörter*, Text und Zahlen in unterschiedlichen Farben an, damit das Programm leichter zu lesen ist. Im nachstehenden Programm ist der Kommentar in der ersten Zeile grün dargestellt, die Schlüsselwörter `print`, `if` und `True` dagegen blau, der Text `'LEGO'` violett und die Zahl `123` orange.

```
# Das ist ein Kommentar.
print('LEGO')
if True:
    print(123)
```



Als PDF

In der Entwicklungsumgebung

Welche Bilder gibt es? Ein Blick in die Dokumentation

IMAGE_HEART = 1
IMAGE_HEART_SMALL = 2
IMAGE_HAPPY = 3
IMAGE_SMILE = 4
IMAGE_SAD = 5
IMAGE_CONFUSED = 6
IMAGE_ANGRY = 7
IMAGE_ASLEEP = 8
IMAGE_SURPRISED = 9
IMAGE_SILLY = 10
IMAGE_FABULOUS = 11
IMAGE_MEH = 12
IMAGE_YES = 13
IMAGE_NO = 14
IMAGE_CLOCK12 = 15
IMAGE_CLOCK1 = 16
IMAGE_CLOCK2 = 17
IMAGE_CLOCK3 = 18
IMAGE_CLOCK4 = 19
IMAGE_CLOCK5 = 20
IMAGE_CLOCK6 = 21
IMAGE_CLOCK7 = 22
IMAGE_CLOCK8 = 23
IMAGE_CLOCK9 = 24
IMAGE_CLOCK10 = 25
IMAGE_CLOCK11 = 26
IMAGE_ARROW_N = 27
IMAGE_ARROW_NE = 28
IMAGE_ARROW_E = 29
IMAGE_ARROW_SE = 30
IMAGE_ARROW_S = 31
IMAGE_ARROW_SW = 32
IMAGE_ARROW_W = 33
IMAGE_ARROW_NW = 34
IMAGE_GO_RIGHT = 35
IMAGE_GO_LEFT = 36
IMAGE_GO_UP = 37
IMAGE_GO_DOWN = 38
IMAGE_TRIANGLE = 39
IMAGE_TRIANGLE_LEFT = 40
IMAGE_CHESSBOARD = 41
IMAGE_DIAMOND = 42
IMAGE_DIAMOND_SMALL = 43
IMAGE_SQUARE = 44
IMAGE_SQUARE_SMALL = 45
IMAGE_RABBIT = 46
IMAGE_COW = 47
IMAGE_MUSIC_CROTCHET = 48
IMAGE_MUSIC_QUAVER = 49
IMAGE_MUSIC_QUAVERS = 50
IMAGE_PITCHFORK = 51
IMAGE_XMAS = 52
IMAGE_PACMAN = 53
IMAGE_TARGET = 54
IMAGE_TSHIRT = 55
IMAGE_ROLLERSKATE = 56
IMAGE_DUCK = 57
IMAGE_HOUSE = 58
IMAGE_TORTOISE = 59
IMAGE_BUTTERFLY = 60
IMAGE_STICKFIGURE = 61
IMAGE_GHOST = 62
IMAGE_SWORD = 63
IMAGE_GIRAFFE = 64
IMAGE_SKULL = 65
IMAGE_UMBRELLA = 66
IMAGE_SNAKE = 67

Aufgabe: Zwinkern

Ergänzen Sie das Programm, das den Smiley anzeigt.

Tipps

```
import time
```

```
while True:
```

```
time.sleep_ms(1000)
```

```
light_matrix.IMAGE_HAPPY
```

```
light_matrix.IMAGE_SMILE
```

Lösung: Zwinkern

```
1 from hub import light_matrix
2 import time
3
4 while True:
5     light_matrix.show_image(light_matrix.IMAGE_HAPPY)
6     time.sleep_ms(1000)
7     light_matrix.show_image(light_matrix.IMAGE_SMILE)
8     time.sleep_ms(1000)
```

Aufgabe: Alle Matrix-Bilder

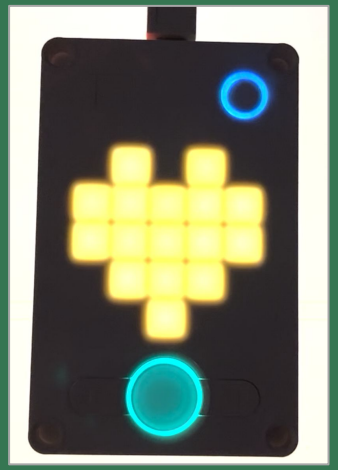


Bild 1

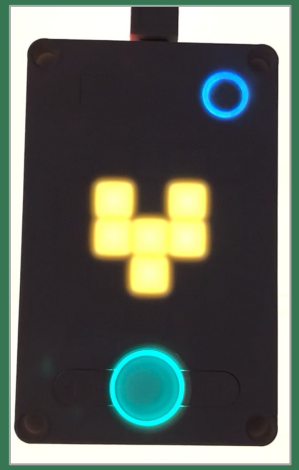


Bild 2

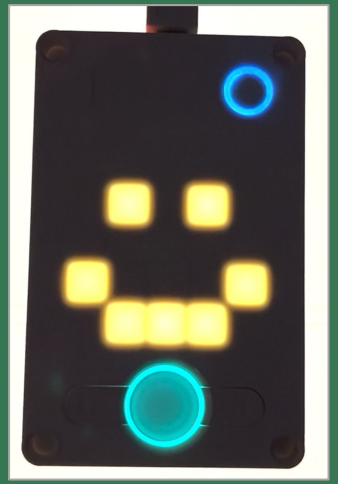


Bild 3

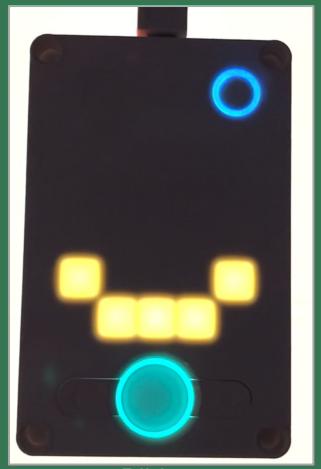


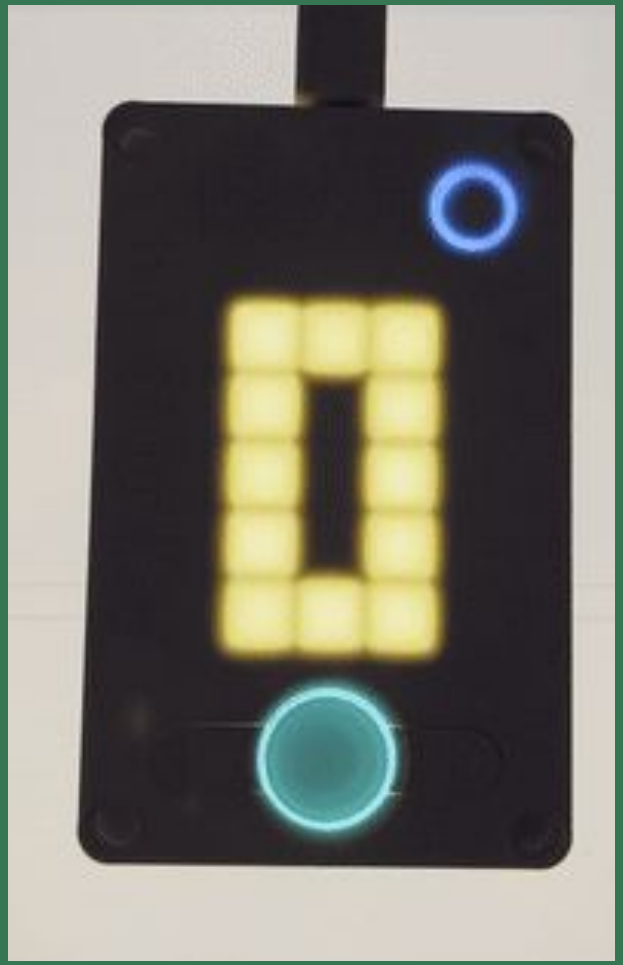
Bild 4

```
1
2  from hub import light_matrix
3  import time
4
5  light_matrix.show_image(1)
6  time.sleep_ms(500)
7
8  light_matrix.show_image(2)
9  time.sleep_ms(500)
10
11 light_matrix.show_image(3)
12 time.sleep_ms(500)
13
14 light_matrix.show_image(4)
15 time.sleep_ms(500)
```

Ändern Sie das Programm links, um mit einer For-Schleife alle Bilder zu zeigen.

Tipp

```
for image_number in range(...):
```



Lösung: Alle Matrix-Bilder

```
1 from hub import light_matrix
2 import time
3
4 for image_number in range(1, 68):
5     light_matrix.show_image(image_number)
6     time.sleep_ms(500)
```

Zwei Texte auf dem Hub - unerwartete Probleme

```
1  from hub import light_matrix
2  import time
3
4
5  light_matrix.show_image(light_matrix.IMAGE_HAPPY)
6  # Pause!
7  time.sleep_ms(1000)
8  light_matrix.write('Hello World!')
9  # Pause! - Aber wie lang?
10 light_matrix.write('Hello Dudweiler!')
```

Run Loop, Async und Await

Um await-Anweisungen zu nutzen und so flexibel auszulegen, dass Befehle entweder gleichzeitig oder nacheinander ausgeführt werden, musst du dein Programm in einer *asynchronen Funktion * mithilfe einer *run-Schleife* ausführen. Das Modul `runloop` steuert die run-Schleife auf dem Hub und lässt dich asynchrone Funktionen mit seiner Funktion `run()` ausführen. Eine asynchrone Funktion, die auch als *Coroutine* bezeichnet wird, ist ein „Awaitable“, bei dem das Schlüsselwort `async` vor die Funktionsdefinition gesetzt wird. Üblicherweise wird die Coroutine angegeben, die dein Hauptprogramm `main()` enthält.

Zwei Texte auf dem Hub - Die Lösung

```
1  from hub import light_matrix
2  import time
3  import runloop
4
5
6  async def main():
7      light_matrix.show_image(light_matrix.IMAGE_HAPPY)
8      # Pause!
9      time.sleep_ms(1000)
10     await light_matrix.write('Hello World!')
11     # Pause! - Aber wie lang?
12     light_matrix.write('Hello Dudweiler!')
13
14 runloop.run(main())
```

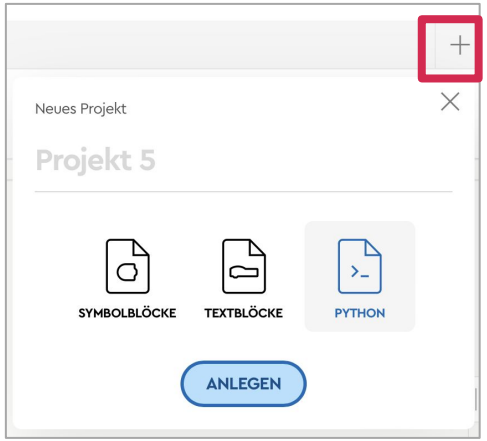
write

write(text: str, intensity: int = 100, time_per_character: int = 500) -> Awaitable

Zeigt einen Text auf der Lichtmatrix als einzelne Buchstaben an, die von rechts nach links durchlaufen. Wenn nur einzelnes Zeichen angezeigt werden soll, läuft dieses nicht von links nach rechts durch

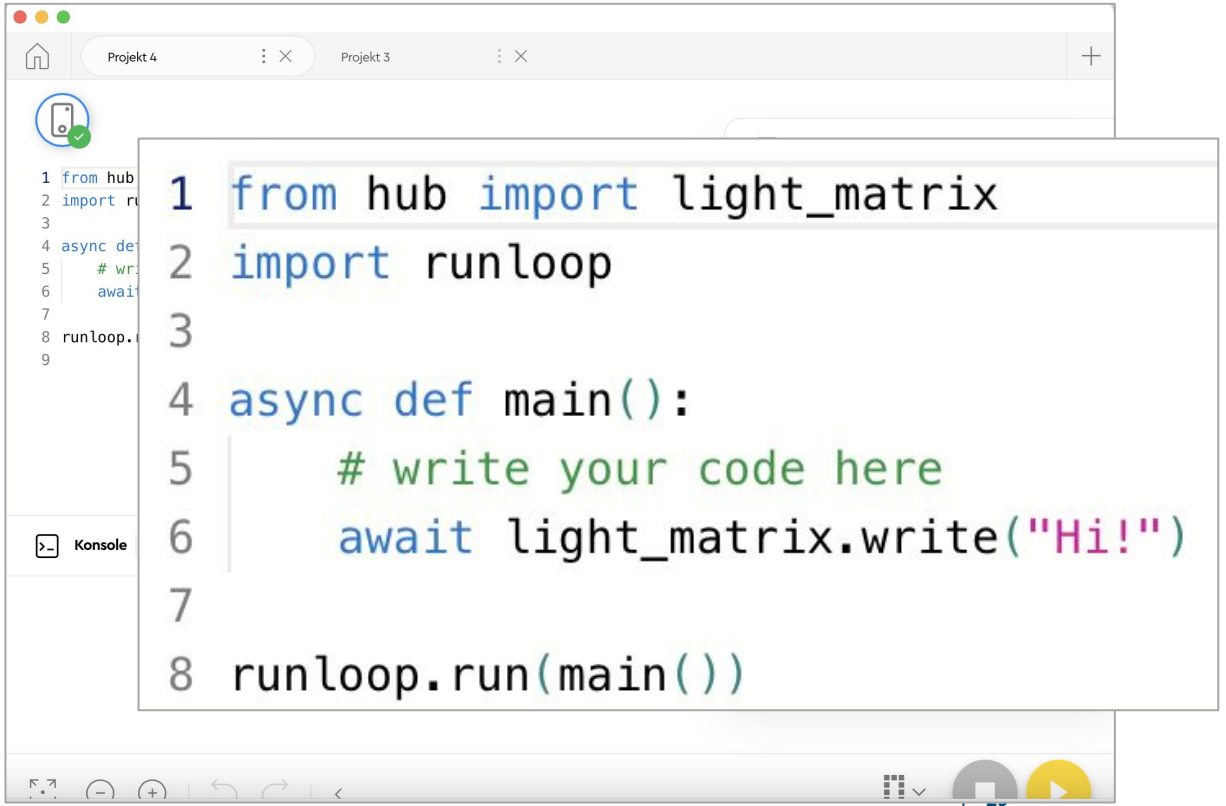
Funktionsdefinition

Startcode in der Entwicklungsumgebung



oder

Auf der Startseite



Töne

```
1  import runloop
2  from hub import sound
3
4
5
6  async def main():
7      await sound.beep(440, 1000000, 100)
8      await sound.beep(880, 200, 100)
9
10 runloop.run(main())
```

beep(frequency: int = 440, duration: int = 500, volume: int = 100, *, attack: int = 0, decay: int = 0, sustain: int = 100, release: int = 0, transition: int = 10, waveform: int = WAVEFORM_SINE, channel: int = DEFAULT) -> Awaitable

Lässt am Hub einen Piepton ertönen

Farben auf dem Hub

```
1 import runloop
2 from hub import light
3 import color
4
5 async def main():
6     light.color(light.POWER, color.)
7
8 runloop.run(main())
9
```

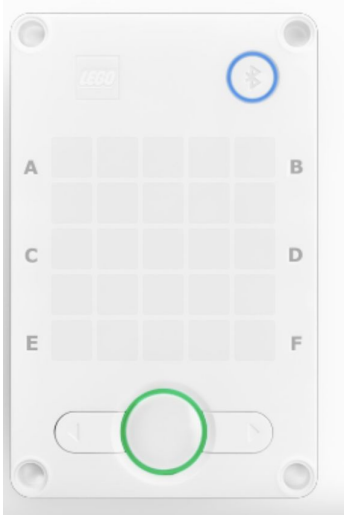
- AZURE
- BLACK
- BLUE
- GREEN
- MAGENTA
- ORANGE
- PURPLE
- RED
- TURQUOISE
- UNKNOWN
- WHITE
- YELLOW

```
1 import runloop
2 from hub import light
3 import color
4
5 async def main():
6     light.color(light., color.RED)
7
8 runloop.run(main())
9
```

- POWER
- color
- CONNECT

Ein-Ausschaltknopf

Bluetooth-Connect-Knopf



Farben auf dem Hub: Powerknopf blinkt

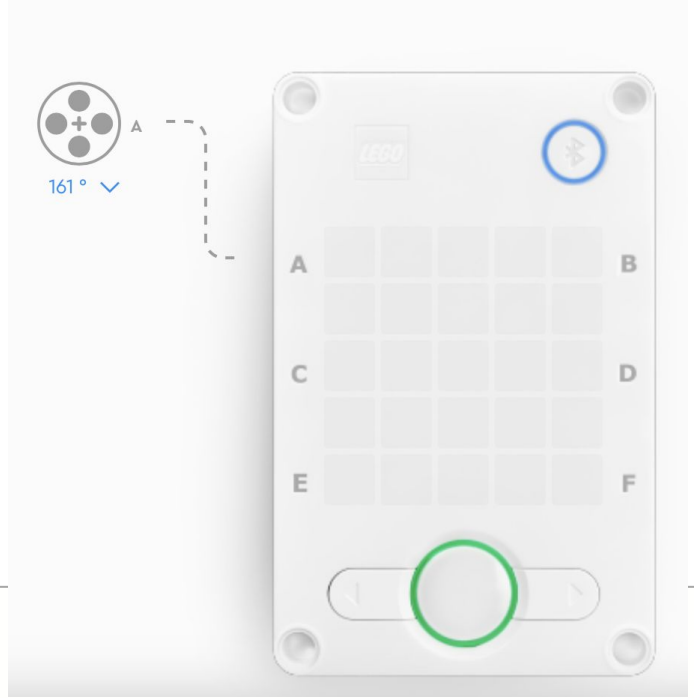
```python
import runloop
from hub import light
import color
import time

async def main():
    while True:
        light.color(light.POWER, color.RED)
        time.sleep_ms(500)
        light.color(light.POWER, color.BLUE)
        time.sleep_ms(500)

runloop.run(main())
```

Alternative

```python
from hub import light
import color
import time

while True:
    light.color(light.POWER, color.RED)
    time.sleep_ms(500)
    light.color(light.POWER, color.BLUE)
    time.sleep_ms(500)
```

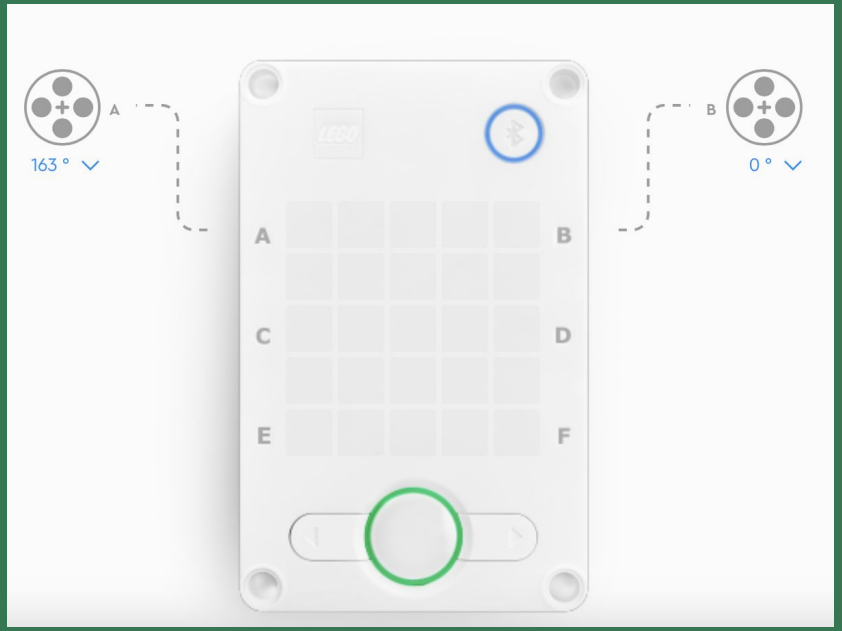
Motoren

Ein Motor




```python
import motor
from hub import port
# Lass einen an Port A (Ein-/Ausgang A) angeschlossenen Motor
# um 360 Grad mit 720 Grad pro Sekunde laufen.
motor.run_for_degrees(port.A, 360, 720)
```

Aufgabe: Zwei Motoren



Ergänzen Sie das Programm, das einen Motor startet so, dass …

…. beide gleichzeitig starten

… sie nacheinander starten

Tipps

```
import runloop
```

```
async def main():
```

```
await
```

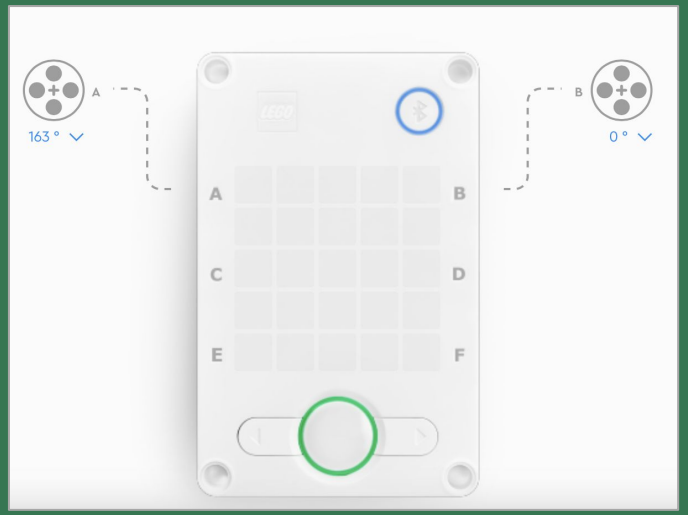
```
runloop.run(main())
```

Lösung:
Zwei Motoren gleichzeitig starten

```python
import motor
from hub import port
motor.run_for_degrees(port.A, 360, 720)
motor.run_for_degrees(port.B, 360, 720)
```

oder

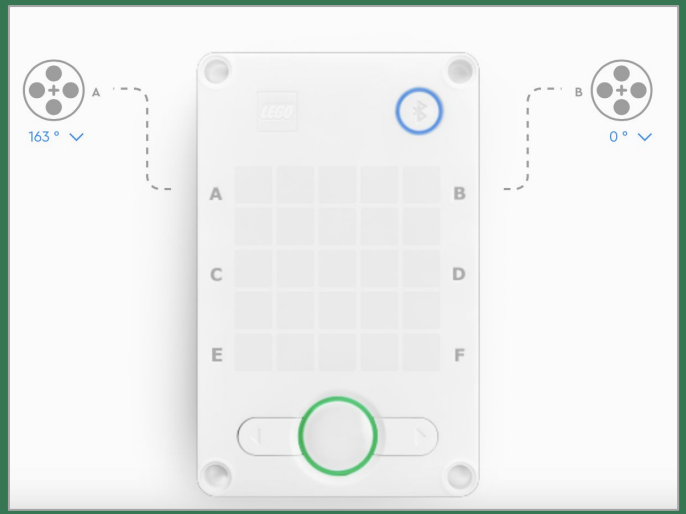
```python
import runloop
import motor
from hub import port

async def main():
    motor.run_for_degrees(port.A, 360, 720)
    motor.run_for_degrees(port.B, 360, 720)

runloop.run(main())
```



Lösung:
Zwei Motoren
nacheinander starten

```
1 import motor
2 from hub import port
3 motor.run_for_degrees(port.A, 360, 720)
4 motor.run_for_degrees(port.B, 360, 720)
```

```
1 import runloop
2 import motor
3 from hub import port
4
5 async def main():
6     await motor.run_for_degrees(port.A, 360, 720)
7     motor.run_for_degrees(port.B, 360, 720)
8
9 runloop.run(main())
```



Drucksensor

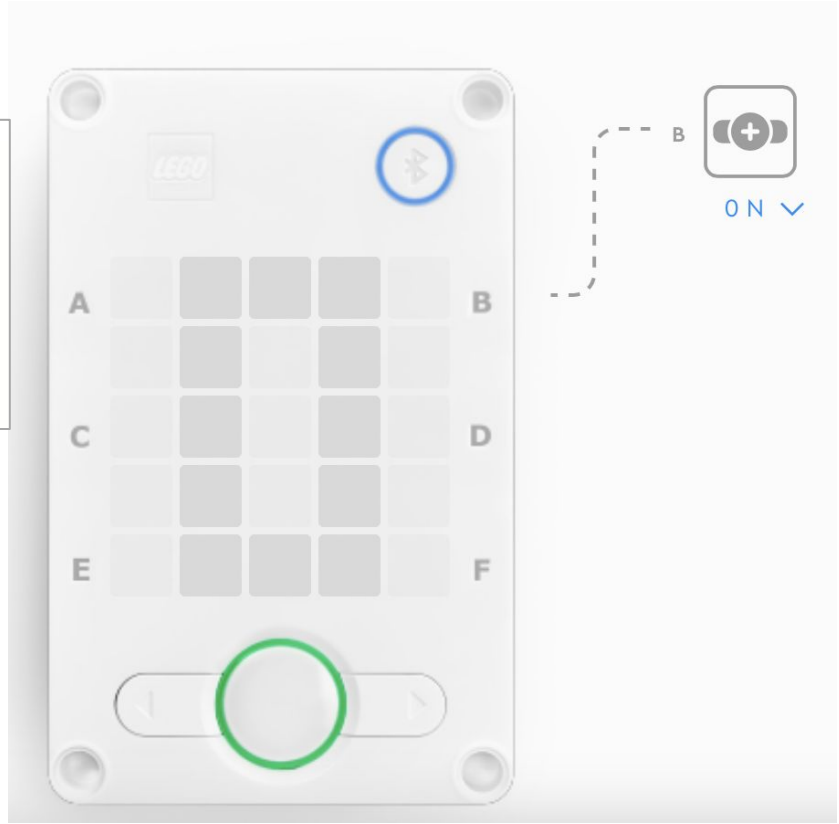
Drucksensor

```
1 import force_sensor
2 from hub import port
3
4 while True:
5     print(force_sensor.force(port.B))
```

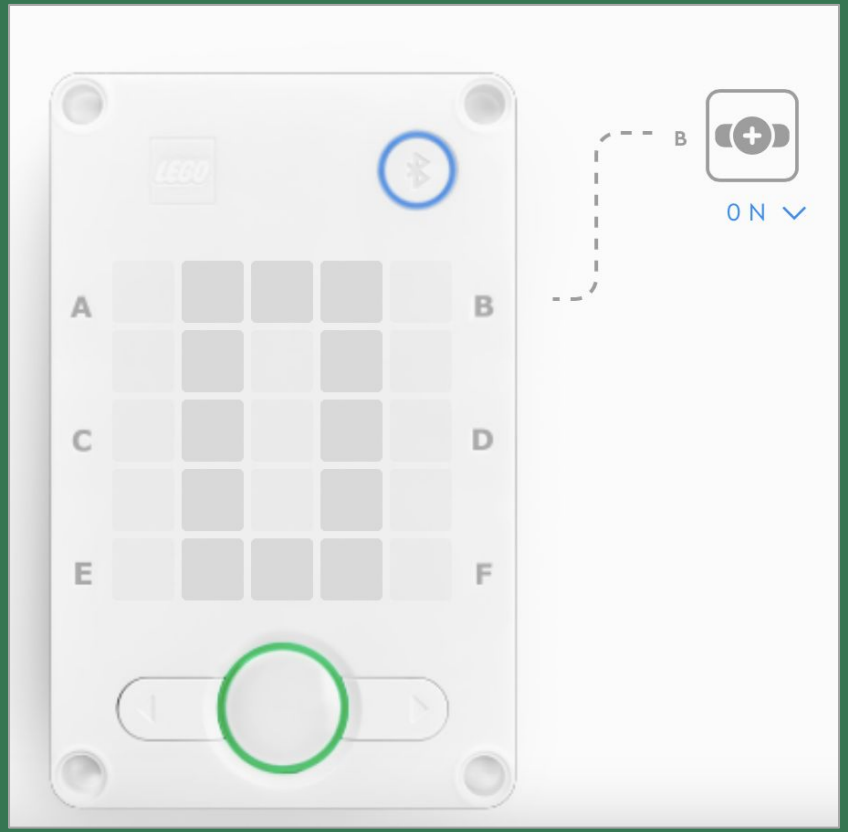
Konsole

```
0
0
0
0
0
0
```

Rückgabewerte der Funktion anzeigen



Aufgabe: Druckampel



Ergänzen Sie das Drucksensor-Programm so, dass der Powerknopf unter 50 grün leuchtet und ab 50 rot.

```
if (                                    ):
    light.color(light.POWER, color.GREEN)
else:
    light.color(light.POWER, color.RED)
```

Tipps

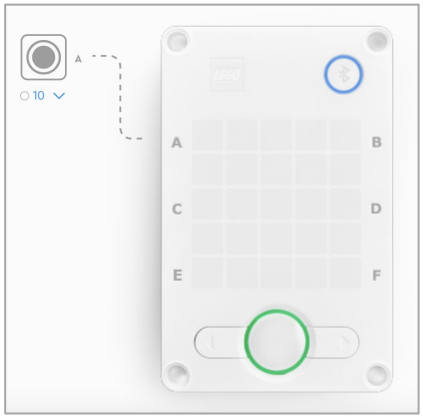
Lösung: Druckampel

```
import force_sensor
from hub import port
from hub import light
import color

while True:
    print(force_sensor.force(port.B))
    if (force_sensor.force(port.B) < 50):
        light.color(light.POWER, color.GREEN)
    else:
        light.color(light.POWER, color.RED)
```

Farbsensor

Farbsensor

```
1 from hub import port
2 import color_sensor
3 import time
4
5 while True:
6     print(color_sensor.color(port.A))
7     time.sleep_ms(500)
```



Aufgabe: Farben erkennen

Ergänzen Sie das Farbsensor-Programm so, dass der Powerknopf jeweils die erkannte Farbe anzeigt

Tipp

```
light.color(light.POWER, erkannte_farbe)
```

Lösung: Farben erkennen

```
from hub import port, light
import color_sensor
import time

while True:
    erkannte_farbe = color_sensor.color(port.A)
    if (erkannte_farbe >= 0 and erkannte_farbe <= 10):
        light.color(light.POWER, erkannte_farbe)
    time.sleep_ms(500)
```

Ultraschallsensor

Ultraschallsensor

```
1 import distance_sensor
2 from hub import port
3 import time
4
5 while True:
6     print(distance_sensor.distance(port.A))
7     time.sleep_ms(500)
```