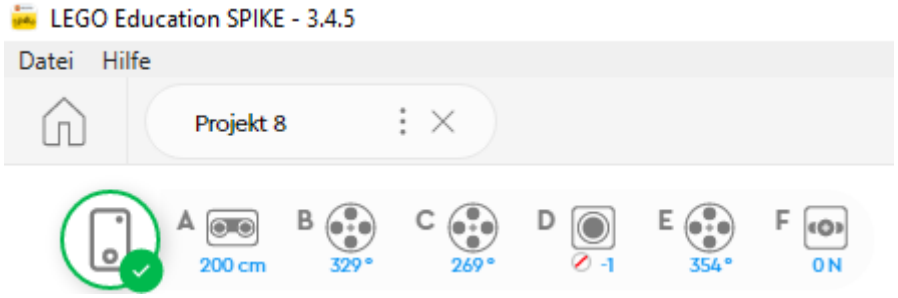
Grundlagen Phyton LEGO Spike Version 3

Entwicklungsumgebung

Auf dem HUB wird *Micro Python* ausgeführt!
Version Spike3 ab 2022 !!!!
Die Angaben der Ports müssen den eigenen Anschlüssen angepasst werden !



Syntax

```
print(123)        # Anweisung
if True:          # Programmblock (mehrere Anweisungen)
   print(123)     # Anweisungen im Block mit 4 Leerzeichen
```

Syntaxhervorhebung: Kommentar, Schlüsselwörter, Text, 123 (Zahlen)

Module / Klassen

Weitere Einträge siehe Anlage **1**

Notwendig, um den Spike HUB sowie die Sensoren und Motoren zu steuern.
Zu jeder Komponente gibt es ein Modul, welches importiert werden muss.
Grundgerüst für Import vom HUB

```
from hub import light, button, light_matrix, motion_sensor, sound, port
```

Kommentar

```
#Das ist ein Kommentar
```

Pause

```
import time                    # Modul einbinden
time.sleep_ms(1000)            # Pause 1 Sekunde
```

Zeitmessung

```
start = time.ticks_ms()                # Zeitmessung starten
time.sleep_ms(1000)                    # zu testende Zeit
stop = time.ticks_ms()                 # Zeitmessung beenden
print(time.ticks_diff(stop, start) / 1000, 'Sekunden') #Zeit ausgeben
```

Funktion definieren

```
from hub import light_matrix           # Modul einbinden
def hello():                           # Schlüsselwort und Funktionsname
   light_matrix.write('Hello, World!') # Hauptteil eingerückt
hello()                                # Aufruf der Funktion
```

mit Übergabeparameter

```
def hello(name):                       # Funktionsname mit Parameter
   light_matrix.write('Hello, ' + name ) # Hauptteil Ausgabe mit Parameter
hello('Spike')                         # Aufruf mit Parameter
```

mit Rückgabewert

```
import force_sensor                    # Modul einbinden
import motor                           # Modul einbinden
from hub import port                   # Modul einbinden
def motor_speed():                     # Funktionsname
   return force_sensor.force(port.C) * 5 # Rückgabe (Drucksensor mal 5)
while True:                            # Endlosschleife
   motor.run(port.F, motor_speed())    # Motor dreht sich
```

Variable

global

```
speed = 360                            # globale Variable
print(speed)                           # Ausgabe
```

lokal

```
def test():                            # Funktionsname
   speed = 180                         # locale Variable (nur für die Funktion)
   print(speed)                        # Ausgabe
test()                                 # Funktionsaufruf
```

Kontrollstrukturen

Zählschleife

```
import time                            # Modul für sleep einbinden
for i in range(4):                     # Schleife wird 4 mal durchlaufen
   print(i)                            # Ausgabe von 0 bis 3
   time.sleep_ms(1000)                 # Jeweils 1 Sek. Pause
# bei jedem Schleifendurchlauf wird i um eins erhöht
```

Endlosschleife

```
while True:                            # Beginn
   light.color(light.POWER,3)          # Powerbutton blau
```

	time.sleep_ms(1000) # Pause 1 Sek. light.color(light.POWER,9) # Powerbutton rot time.sleep_ms(1000) # Pause 1 Sek.

Kopfschleife	durchgang = 1 # Variable Wert zuweisen
	while durchgang < 3: # Kopf prüft ob durchgang <3
	print(durchgang) # Körper
	durchgang = durchgang + 1 # Körper durchgang wird um 1 erhöht
	print("Schleife Ende ") # Ausgabe nach Schleifenende

If-Anweisung (Bedingung) if – else	while True: # Endlosschleife if (force_sensor.force(port.C)) < 50: # Abfrage ob Kraftsensor unter 50 print("zu gering") # entsprechende Ausgabe else: # Altenative zu if (wenn Sensor über 50) print("stark") # entsprechende Ausgabe time.sleep_ms(1000) # Pause 1 Sekunde
If- elif- else	while True: # Endlosschleife if (color_sensor.color(port.A)) == 9: # Abfrage ob Fabe Rot erkannt wird print("Rot") # entsprechende Ausgabe elif (color_sensor.color(port.A)) == 6: # zweite Abfrage ob Farbe Grün erkannt print("Grün") # entsprechende Ausgabe else: # Sonst-Zweig für Rest (Rot/Grün nicht) print("nicht erkannt") # entsprechende Ausgabe time.sleep_ms(1000) # Pause 1 Sekunde
Zufallszahl erzeugen	import random # Modul für Zufall einbinden print(random.randint(0, 10)) # Zufallszahl zwischen 0 und 10

Listen	my_list = [1,3,7] # Liste mit 3 Farbwerten
	farb=random.choice(my_list) # Var farb bekommt Zufallswert v. my_list
	light.color(light.POWER,farb) # Farbausgabe Powerbutton

Anhang 1:

```
from hub import light, button, light_matrix, motion_sensor, sound, port
import device
import runloop
import color
import time
import motor
import force_sensor
import color
import color_sensor
import random
import motor_pair
import distance_sensor
import math
```

Werden diese Module am Anfang in das Programm übernommen, brauchen sie nicht mehr aus den Beispielen kopiert werden.

HUB - Anzeige, Eingabe und Sound	

Textausgabe auf HUB	from hub import light_matrix # Modul einbinden
	light_matrix.write("HalloWelt")

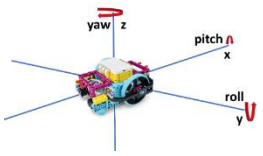
Smilies auf dem Hub
Tabelle Anhang **2**
	from hub import light_matrix # Modul einbinden
	light_matrix.show_image(light_matrix.IMAGE_HAPPY) # oder
	light_matrix.show_image(2)

Anzeige löschen	from hub import light_matrix # Modul einbinden light_matrix.show_image(4) # Ausgabe Smilie time.sleep_ms(1000) # Pause 1 Sekunde light_matrix.clear() # Ausgabe löschen

Pixel der Matrix ausgeben	from hub import light_matrix # Modul einbinden
	light_matrix.set_pixel(4,4,100) # x-Pos, y-Pos, Helligkeit (0 – 100)

Powerbutton Farbe ändern	import color # Modul einbinden from hub import light # Modul einbinden light.color(light.POWER, color.RED) # POWER oder CONNECT für BT-Knopf # AZURE, BLACK, BLUE, GREEN, MAGENTA, ORANGE, PUPLE, RED, WHITE, YELLOW

Button links / rechts	from hub import button # Modul einbinden
	while True: # Endlosschleife
	  if button.pressed(button.LEFT): # linker Knopf
	  light.color(light.POWER, color.RED) # Powerbutton rot
	  elif button.pressed(button.RIGHT): # rechter Knopf
	  light.color(light.POWER, color.GREEN) # Powerbutton grün

Kreiselkompass
 	
Beschleunigung
 
 	
Winkelgeschwindigkeitswerte
 	
Gestenerkennung1
 	
Gestenerkennung2
 	
 
 	
Seite des Hubs, die oben ist
 	
Richtungsmessung
	from hub import motion_sensor
	motion_sensor.set_yaw_face(motion_sensor.FRONT)
	motion_sensor.reset_yaw(0) # Nullpunkterkennung
	while True: # Dauerschleife
	  print(motion_sensor.acceleration()) # Ausgabe der Beschleunigungswerte
	  # x, y, z als 1/1000 G in Milli-G
	  time.sleep_ms(500) # Pause 0,5 Sekunden
	……
	while True: # Dauerschleife
	  print(motion_sensor.angular_velocity()) # x, y, z als Dezigrad pro Sekunde
	  time.sleep_ms(500) # Pause 0,5 Sekunden
	……
	while True: # Dauerschleife
	  print(motion_sensor.gesture()) # -1=nicht erkannt, 0= Klick, 1= Doppel-
	  # klick, 2= schütteln, 3= fallen
	  time.sleep_ms(500) # Pause 0,5 Sekunden
	……
	while True: # Dauerschleife
	  if motion_sensor.gesture() == 2: # Abfrage der Geste
	  print("OK") # Ausgabe OK wenn geschüttelt
	  time.sleep_ms(500) # Pause 0,5 Sekunden
	……
	print(motion_sensor.up_face()) # Wert in Klammer -> gegenüber
	……
	while True: # Dauerschleife
	  print(motion_sensor.tilt_angles()) # die Werte werden in Dezigrad angegeben
	  time.sleep_ms(500) # Pause 0,5 Sekunden
	# yaw(z)=links/rechts drehen(gier), roll(y)=links/rechts kippen(roll),
	# pitch(x)=vor/zurück kippen(nick)

Soundausgabe	from hub import sound # Modul einbinden sound.beep(440, 500, 100) # Frequenz, Dauer in ms, Volumen 0-100 …… sound.stop() # bei Bedarf

Aktoren und Sensoren	
Motor ein/ausschalten	`import motor, time` # Modul einbinden `from hub import port` # Modul einbinden `motor.run(port.F, 1000)` # Motor starten – Speed = 1000 `time.sleep_ms(2000)` # Pause 2 Sekunden `motor.stop(port.F)` # Motor stoppen
Motorsteuerung (ein Motor – Werte definiert)	`import motor` # Modul einbinden `from hub import port` # Modul einbinden `motor.run_for_degrees(port.F, 360, 720)` # Port, 360 Grad (1 Umdrehung) # Speed (hier 2 Umdrehungen pro Sek)
Motorsteuerung (2 Motoren) gleichzeitig	`import motor` # Modul einbinden `from hub import port` # Modul einbinden `motor.run_for_degrees(port.A, 360, 720)` # beide Motoren laufen gleichzeitig `motor.run_for_degrees(port.B, 360, 720)` # run_for_degrees() = *await*-Funktion - # - Befehl, der nicht erst abgeschlossen sein muss, bevor der nächste Befehl gestartet # wird, sondern unmittelbar nach dem Start wird auch der nächste Befehl gestartet.
Motorsteuerung (2 Motoren) Nacheinander Anlage X	`import runloop` # für Schlüsselwörter ob abwartende - `import motor` # - Befehle gleichzeitig oder nacheinander - `import port` # - ausgeführt werden sollen `async def main():` # async -> Funktion nicht gleichzeitig `  await motor.run_for_degrees(port.B, 360, 720)` # await friert den Code ein bis `  await motor.run_for_degrees(port.C, 360, 720)` # der Befehl ausgeführt wurde `runloop.run(main())` # Aufruf der Funktion main()
Fahrroboter (2 Motoren, die Zusammenarbeiten) Siehe Anlage **3**	`from hub import port` # Modul einbinden `import runloop` # Modul einbinden `import motor_pair` # Modul zum Verbinden von 2 Motoren `async def main():` # async -> Funktion nicht gleichzeitig `  motor_pair.pair(motor_pair.PAIR_1, port.D, port.F)` # Zusammenfügen d.Motoren `  motor_pair.move(motor_pair.PAIR_1, 0, velocity=1000)` # 0 -> Parameter für Lenkung geradeaus, velocity -> Geschwindigkeit (max. 1100) `  await runloop.sleep_ms(5000)` # Zeit, die die Motoren laufen `  motor_pair.stop(motor_pair.PAIR_1)` # Motoren stoppen `runloop.run(main())` # Aufruf der Funktion main() `raise SystemExit` # Progammende (immer empfohlen)
Fahrroboter genaue Strecke	`from hub import port` # Modul einbinden `import runloop` # Modul einbinden `import motor_pair` # Modul einbinden `async def main():` # async -> Funktion nicht gleichzeitig `  motor_pair.pair(motor_pair.PAIR_1, port.D, port.F)` # Zusammenfügen d.Motoren `  motor_pair.move_for_degrees(motor_pair.PAIR_1, 1028, 0, velocity=1000)` # 360 -> eine Umdrehung (Standardrad hat 17,5 cm für blaues Rad) # Beispiel für 50 cm = 50/17,5= 2,857 * 360 = 1028 (immer auf ganzen Wert) `runloop.run(main())` # Aufruf der Funktion main()
Motorposition auslesen	`import motor` # Modul einbinden `from hub import port` # Modul einbinden `while True:` # Dauerschleife `  print(motor.absolute_position(port.F))` # Ausgabe der Motorposition (0 – 360°) `  time.sleep_ms(1000)` # Pause 1 Sekunde
Kraftsensor	`import time` # Modul einbinden `import force_sensor` # Modul einbinden `while True:` # Endlosschleife `  print(force_sensor.force(port.F))` # Ausgabe der Variable (0 bis 100) `  time.sleep_ms(500)` # Pause 0,5 Sek.
Farbsensor Farbe als Zahlenwert erkennen	`import color` # Modul einbinden `import color_sensor` # Modul einbinden `while True:` # Endlosschleife

	print(color_sensor.color(port.D)) # Wert zwischen 0 und 10 time.sleep_ms(500) # Pause 0,5 Sekunden # 0=schwarz,1=pink, 2=violett, 3=blau, 4=hellblau, 6=grün, 7=gelb, 8=orange, 9=rot, # 10=weiß, -1=keine Farbe erkannt
Farbsensor als Linienverfolger (Reflektierendes Licht)	import color # Modul einbinden import color_sensor # Modul einbinden while True: # Endlosschleife print(color_sensor.reflection(port.A)) # Wert zwischen 0 und 100 time.sleep_ms(1000) # Pause 1 Sekunde
Ultraschallsensor Abstandsmessung	import distance_sensor # Modul einbinden while True: # Endlosschleife print(distance_sensor.distance(port.A)) # Ausgabe des Wertes in mm # Wert zw. 5 und 200 cm, -1=nicht erkannt time.sleep_ms(200) # Pause 0,2 Sek.
Ultraschallsensor Licht einschalten (jedes einzelne Feld)	import distance_sensor # Modul einbinden distance_sensor.set_pixel(port.B, 0, 0, 100) # 0,0 >- Position # 100 -> Helligkeit Position 0,0 1,0 0,1 1,1 

Ultraschallsensor
Licht einschalten
(alle Felder gleichzeitig)
	import distance_sensor # Modul einbinden
	pixels = [100] * 4 # Liste mit 4 gleichen Intensitätsweten
	distance_sensor.show(port.D, pixels) # Ansteuern aller 4 LED´s

Ultraschallsensor Licht ausschalten	import distance_sensor # Modul einbinden distance_sensor.clear(port.B) # alle Felder ausschalten

Anlage 2

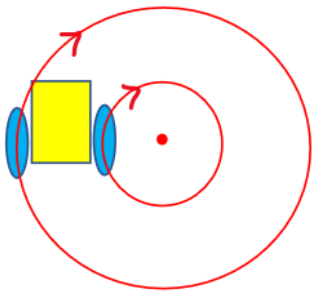
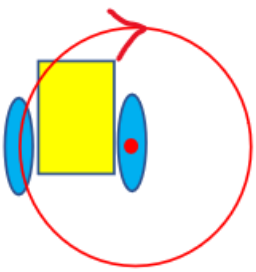
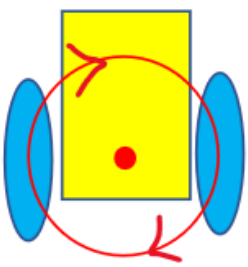
IMAGE_HEART = 1
IMAGE_HEART_SMALL = 2
IMAGE_HAPPY = 3
IMAGE_SMILE = 4
IMAGE_SAD = 5
IMAGE_CONFUSED = 6
IMAGE_ANGRY = 7
IMAGE_ASLEEP = 8
IMAGE_SURPRISED = 9
IMAGE_SILLY = 10
IMAGE_FABULOUS = 11
IMAGE_MEH = 12
IMAGE_YES = 13
IMAGE_NO = 14
IMAGE_CLOCK12 = 15
IMAGE_CLOCK1 = 16
IMAGE_CLOCK2 = 17
IMAGE_CLOCK3 = 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

Anlage 3

Lenkung	Zirkeldrehung	Kreiseldrehung
		
Wert 0 bis 50 oder 0 bis -50	Wert 50 oder -50	Wert 100 oder -100