

Lösungsblatt

Diese Fassung enthält die Lösungen.

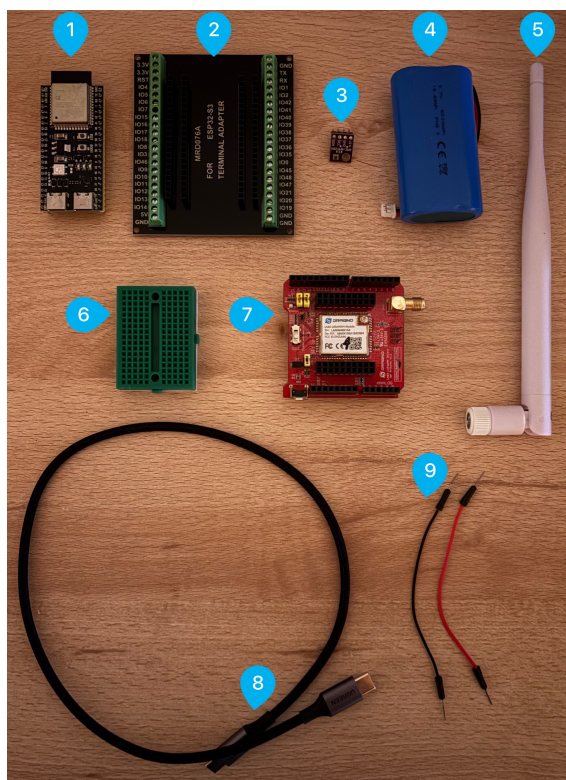
Wie die Aufgaben aufgebaut sind

Standard ist das Ziel, fang immer dort an. **Einstieg** führt hin, falls du nicht reinkommst. **Experte** geht weiter, wenn du fertig bist. Nicht jede Aufgabe hat alle drei Stufen.

Zum Nachschlagen liegen **Kärtchen** in der Tischmitte – nimm dir, was du brauchst, und leg es zurück, wenn du fertig bist.

Aufgabe 1 – Was liegt vor dir?

Das sind die Teile aus deiner Projektbox. Trag ein, wie sie heißen.



Nr.	Name
1	
2	
3	
4	
5	
6	
7	
8	
9	

Lösung

Nr.	Name
1	ESP32-S3, das Mikrocontroller-Board
2	Terminal-Adapter
3	Sensor (BME280)
4	Akku
5	Antenne
6	Breadboard
7	LA66-Shield, das Funkmodul
8	USB-C-Kabel
9	Jumperkabel

Heute gebraucht werden nur 1, 2 und 8. Sensor, Breadboard und Jumperkabel kommen an Tag 3 dazu, Shield, Antenne und Akku an Tag 4.

Sieh dir Teil 1 genauer an: Wofür sind die vielen kleinen Anschlüsse am Rand da?

Lösung

Es sind die GPIO-Pins. Über sie gibt das Board Signale aus und nimmt welche entgegen, dort werden später die Sensoren angeschlossen.

Aufgabe 2 – Die LED auf dem Board

Standard

Öffne `led-an.py`, speichere die Datei **auf dem Board** und starte sie.

```
# Die LED auf dem Board zum Leuchten bringen.  
  
import machine  
import neopixel  
  
led = neopixel.NeoPixel(machine.Pin(48), 1)  
  
led[0] = (0, 0, 30)  
led.write()
```

Die drei Zahlen in der vorletzten Zeile sind Rot, Grün und Blau. Trag ein, welche Zahlen du für die Farbe erwartest, probier es dann aus.

Farbe	Rot	Grün	Blau	Hat es gepasst?
-------	-----	------	------	-----------------

Rot

Grün

Blau

Gelb

Türkis

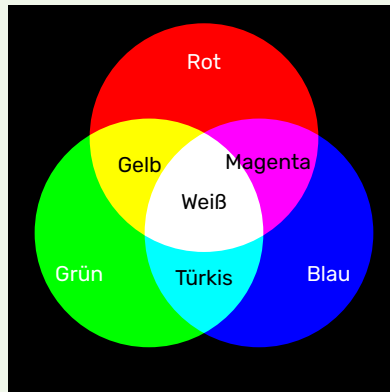
Magenta

Weiß

Lösung

Farbe	Rot	Grün	Blau	Was leuchtet
Rot	30	0	0	eine Leuchtdiode
Grün	0	30	0	eine Leuchtdiode
Blau	0	0	30	eine Leuchtdiode
Gelb	30	30	0	Rot und Grün
Türkis	0	30	30	Grün und Blau
Magenta	30	0	30	Rot und Blau
Weiß	30	30	30	alle drei

Die Zahlen gehen jeweils von 0 bis 255; kleine Werte genügen, die LED ist sehr hell.



Die Mischungen sind der interessante Teil: In der LED steckt keine gelbe Leuchtdiode. Gelb entsteht erst im Auge, weil Rot und Grün gleichzeitig leuchten – Licht wird addiert, nicht gemischt wie Farbe im Tuschkasten.

Experte

Schreib ein Programm, das nacheinander rot, gelb und grün leuchtet. Zwischen den Farben soll jeweils eine Sekunde vergehen:

```
# Ganz oben im Programm nur einmal
import time

# Immer im Programm wenn eine Sekunde pausiert werden soll
time.sleep(1)
```

Was ist daran umständlich, wenn der Farbwechsel zehnmal hintereinander laufen soll?

Lösung

farbwechsel.py

```
import machine, neopixel, time

led = neopixel.NeoPixel(machine.Pin(48), 1)

led[0] = (30, 0, 0)
led.write()
time.sleep(1)

led[0] = (30, 30, 0)
led.write()
time.sleep(1)

led[0] = (0, 30, 0)
led.write()
time.sleep(1)
```

Für zehn Durchläufe müsste man denselben Block zehnmal untereinander schreiben. Genau dafür gibt es Schleifen.

Aufgabe 3 – Werte merken

Standard

Trag ein, welche Art von Wert das jeweils ist: ganze Zahl, Kommazahl, Wahrheitswert oder Text.

Wert	Art	type(...)
23		
18.4		
"Schulgarten"		
True		
"18.4"		

Prüf deine Einträge in der Kommandozeile nach. Wie heißen die Datentypen in Python?

```
>>> type(18.4)
```

Lösung

Wert	Art	heißt in Python
23	ganze Zahl	int
18.4	Kommazahl	float
"Schulgarten"	Text	str
True	Wahrheitswert	bool
"18.4"	Text	str

Die letzte Zeile ist die interessante: Anführungszeichen machen aus einer Zahl einen Text.

Jetzt selbst: Denk dir eine Messung aus und schreib ein Programm `messung.py`. Leg jede der vier Angaben in einer eigenen Variablen ab, im Datentyp, der dahintersteht:

- a) wo gemessen wurde – Text
- b) wie warm es war – Kommazahl
- c) wie oft gemessen wurde – ganze Zahl
- d) ob es wärmer als 20 Grad war – Wahrheitswert

Gib zum Schluss alle vier Angaben mit `print` aus, jede in einer eigenen Zeile.

Lösung

`messung.py`

```
messstelle = "Schulgarten"
temperatur = 21.5
anzahl = 3
warm = True

print("Messstelle:", messstelle)
print("Temperatur:", temperatur)
print("Messungen:", anzahl)
print("Über 20 Grad:", warm)
```

Namen und Werte sind frei wählbar – die Messung ist erfunden. Es geht darum, dass jede Angabe im passenden Datentyp landet.

Experte

Was passiert bei dieser Zeile? Schreib deine Vermutung auf, bevor du es ausprobierst.

```
>>> "21.5" + 1
```

Probier es aus, notiere die Fehlermeldung und repariere die Zeile.

Lösung

`TypeError: can't convert 'int' object to str implicitly`. Links steht ein Text, rechts eine Zahl – Python addiert die nicht stillschweigend. Reparatur: `float("21.5") + 1` ergibt 22.5.

S015

Aufgabe 4 – Variablen: Namen für Werte

Standard

Tipp die Zeilen der Reihe nach in die Kommandozeile – jede einzeln, mit Enter. Trag ein, was das Board antwortet.

Du tippst	Das Board antwortet
-----------	---------------------

<code>temperatur = 21.5</code>	
--------------------------------	--

<code>temperatur</code>	
-------------------------	--

<code>temperatur + 1</code>	
-----------------------------	--

<code>temperatur</code>	
-------------------------	--

<code>temperatur = 18.0</code>	
--------------------------------	--

<code>temperatur</code>	
-------------------------	--

Lösung

Du tippst	Antwort	Warum
<code>temperatur = 21.5</code>	nichts	Zuweisen ist stilles Wegräumen, keine Ausgabe
<code>temperatur</code>	21.5	der Name wird durch seinen Wert ersetzt
<code>temperatur + 1</code>	22.5	gerechnet wird mit dem Wert, den der Name gerade hat
<code>temperatur</code>	21.5	unverändert – Rechnen ändert die Variable nicht
<code>temperatur = 18.0</code>	nichts	wieder eine Zuweisung
<code>temperatur</code>	18.0	der alte Wert ist überschrieben und weg

Beantworte mit dem, was du gerade beobachtet hast:

- a) Nach `temperatur + 1` steht in `temperatur` immer noch 21.5. Warum?
- b) Was ist mit den 21.5 passiert, nachdem du `temperatur = 18.0` getippt hast?

Lösung

- a) `temperatur + 1` rechnet nur aus und zeigt das Ergebnis. Damit sich die Variable ändert, müsste das Ergebnis wieder hineingeschrieben werden: `temperatur = temperatur + 1`.
- b) Sie ist weg. Ein Name hält genau einen Wert; die neue Zuweisung überschreibt den alten.

Wie das Gleichheitszeichen zu lesen ist

Nicht als „ist gleich“, sondern **von rechts nach links**: Rechne die rechte Seite aus und leg das Ergebnis unter dem Namen links ab. Deshalb ergibt `anzahl = anzahl + 1` Sinn, als Gleichung wäre es Unfug.

Experte

In a steht 1, in b steht 2. Vertausche die beiden Werte, sodass danach 2 in a und 1 in b steht.

```
a = 1
b = 2
```

Probier zuerst `a = b` und danach `b = a`. Was geht dabei schief?

Lösung

`a = b` überschreibt die 1 – danach steht in beiden Variablen 2, und `b = a` holt sie nicht zurück. Man braucht einen dritten Namen, der den Wert so lange aufhebt:

```
helf = a
a = b
b = helf
```

Das ist dieselbe Einsicht wie in der Standardaufgabe: Zuweisen legt ab und wirft dabei weg, was vorher da war.

S011

Aufgabe 5 – Mit Werten rechnen

Einstieg

Sag mit den Kärtchen **Rechnen** und **Vergleichen** voraus, was herauskommt. Tipp es danach in die Kommandozeile und vergleiche.

Eingabe	Deine Vorhersage	Tatsächlich
9 / 4		
9 // 4		
9 % 4		
3 ** 4		
21.5 >= 20		

Lösung

Eingabe	Ergebnis	Warum?
9 / 4	2.25	Division, das Ergebnis ist immer eine Kommazahl
9 // 4	2	ganzzahlige Division, der Rest fällt weg
9 % 4	1	nur der Rest
3 ** 4	81	Potenz, also 3 hoch 4
21.5 >= 20	True	ein Vergleich ergibt einen Wahrheitswert

Standard

Öffne `zwei-messwerte.py`:

```
# Zwei Messwerte aus dem Schulgarten.
# Ergänze das Programm unten.

morgens = 12.4
mittags = 21.8

print("Morgens:", morgens, "Grad Celsius")
print("Mittags:", mittags, "Grad Celsius")
```

Ergänze das Programm so, dass es zusätzlich ausgibt, um wie viel Grad es wärmer wurde und wie warm es im Mittel war.

Mittelwert = (erster Wert + zweiter Wert) : 2

Lösung

zwei-messwerte.py

```
unterschied = mittags - morgens
mittelwert = (morgens + mittags) / 2

print("Unterschied:", unterschied, "Grad")
print("Im Mittel:", mittelwert, "Grad")
```

Die Klammern beim Mittelwert sind nötig – ohne sie wird nur mittags halbiert.

Experte

Ergänze eine Warnung: Das Programm soll ausgeben, ob es mittags wärmer als 20 Grad war. Rechne außerdem die Mittagstemperatur in Grad Fahrenheit um.

Fahrenheit = Celsius • 1,8 + 32

Lösung

```
zu_warm = mittags > 20
print("Über 20 Grad:", zu_warm)

fahrenheit = mittags * 1.8 + 32
print("Mittags in Fahrenheit:", fahrenheit)
```

`mittags > 20` ist selbst schon der Wahrheitswert – man muss ihn nicht mit einer Abfrage herstellen. Abfragen kommen an Tag 2.

S012

S013

Aufgabe 6 – Funktionen benutzen

Einstieg

`print` hast du längst benutzt – `print` ist eine **Funktion**, und es gibt viele davon. Sag mit dem Kärtchen **Funktionen zum Rechnen** voraus, was herauskommt, und prüf es dann in der Kommandozeile.

Eingabe	Deine Vorhersage	Tatsächlich
<code>len("Schulgarten")</code>		
<code>round(21.47)</code>		
<code>abs(-3.2)</code>		
<code>max(12.4, 21.8, 18.0)</code>		

Lösung

Eingabe	Ergebnis	Was die Funktion tut
<code>len("Schulgarten")</code>	11	zählt die Zeichen
<code>round(21.47)</code>	21	rundet auf eine ganze Zahl
<code>abs(-3.2)</code>	3.2	lässt das Minus weg
<code>max(12.4, 21.8, 18.0)</code>	21.8	nimmt den größten Wert

Der Name steht vorn, in den Klammern steht, womit gearbeitet werden soll.

Standard

Erweitere `zwei-messwerte.py`: Der Mittelwert soll auf eine Nachkommastelle gerundet ausgegeben werden.

`round` nimmt eine zweite Angabe in den Klammern – probier aus, was sie bewirkt.

Lösung

```
mittelwert = (morgens + mittags) / 2
print("Im Mittel:", round(mittelwert, 1), "Grad")
```

Die zweite Angabe ist die Zahl der Nachkommastellen. Ohne sie rundet `round` auf eine ganze Zahl.

Experte

Nimm einen dritten Messwert dazu (`abends`). Lass das Programm ausgeben, wie groß die Spanne zwischen dem wärmsten und dem kältesten Wert des Tages war.

Lösung

```
abends = 15.1

waermster = max(morgens, mittags, abends)
kaeltester = min(morgens, mittags, abends)

print("Spanne:", round(waermster - kaeltester, 1), "Grad")
```

max und min vertragen beliebig viele Werte in den Klammern. Mit einer Liste ginge es kürzer – die kommt an Tag 2.

S014

S015

Anhang • Zeichen und Funktionen

Alles von den Kärtchen noch einmal, zum Mitnehmen und Nachschlagen. Auswendig lernen musst du davon nichts.

Zuweisen

Zeichen	Was es tut	Beispiel	Kurz für
=	schreibt einen Wert in eine Variable	anzahl = 3	
+=	zählt dazu	anzahl += 1	anzahl = anzahl + 1
-=	zieht ab	anzahl -= 1	anzahl = anzahl - 1
*=	vervielfacht	anzahl *= 2	anzahl = anzahl * 2
/=	teilt	anzahl /= 2	anzahl = anzahl / 2

Rechnen

Zeichen	Was es tut	Beispiel	Ergebnis
+	addieren, bei Text aneinanderhängen	3 + 4	7
-	subtrahieren	10 - 4	6
*	multiplizieren	3 * 4	12
/	teilen, Ergebnis immer Kommazahl	8 / 5	1.6
//	teilen ohne Rest	8 // 5	1
%	Modulo, nur der Rest	8 % 5	3
**	potenzieren	2 ** 5	32

Punkt vor Strich gilt auch hier, und Klammern setzen sich durch: $2 + 3 * 4$ ergibt 14, $(2 + 3) * 4$ ergibt 20.

Vergleichen

Vergleiche ergeben immer True oder False:

Zeichen	Was es tut	Beispiel	Ergebnis
==	ist gleich	20 == 20	True
!=	ist ungleich	20 != 20	False
>	ist größer als	21.5 > 20	True
<	ist kleiner als	21.5 < 20	False
>=	ist größer oder gleich	20 >= 20	True
<=	ist kleiner oder gleich	21 <= 20	False

Ein Zeichen macht den Unterschied

= schreibt einen Wert hinein, == fragt nach.

temperatur = 20 setzt die Temperatur auf 20. temperatur == 20 fragt, ob sie 20 ist.

Funktionen zum Rechnen

Die sind immer da, du musst nichts dafür einbinden:

Funktion	Was sie tut	Beispiel	Ergebnis
<code>round(x)</code>	rundet auf eine ganze Zahl	<code>round(3.7)</code>	4
<code>round(x, n)</code>	rundet auf n Nachkommastellen	<code>round(3.14159, 2)</code>	3.14
<code>int(x)</code>	schneidet die Nachkommastellen ab	<code>int(3.9)</code>	3
<code>float(x)</code>	macht eine Kommazahl daraus	<code>float(3)</code>	3.0
<code>abs(x)</code>	lässt das Vorzeichen weg	<code>abs(-7.5)</code>	7.5
<code>max(...)</code>	gibt den größten Wert	<code>max(3, 9, 5)</code>	9
<code>min(...)</code>	gibt den kleinsten Wert	<code>min(3, 9, 5)</code>	3
<code>len(...)</code>	zählt, wie viele Zeichen es sind	<code>len("Messung")</code>	7

`int` und `round` sind nicht dasselbe: `int(3.9)` ergibt 3, `round(3.9)` ergibt 4.

Funktionen für alles andere

Funktion	Was sie tut	Beispiel	Ergebnis
<code>print(...)</code>	gibt etwas in der Kommandozeile aus	<code>print("Temp:", 21.5)</code>	Temp: 21.5
<code>type(x)</code>	sagt, welche Art von Wert das ist	<code>type(18.4)</code>	<class 'float'>
<code>str(x)</code>	macht Text aus einem Wert	<code>str(21.5)</code>	"21.5"
<code>input(...)</code>	wartet, bis du etwas eintippst	<code>input("Name: ")</code>	dein Text

`print` verträgt mehrere Werte, durch Komma getrennt – dazwischen setzt es automatisch ein Leerzeichen.

Funktionen aus einer Bibliothek

`time.sleep(1)` aus Aufgabe 2 gehört nicht dazu: Es steckt in der Bibliothek `time` und muss mit `import time` erst eingebunden werden. Woher solche Funktionen kommen, klärt sich an Tag 3.