Was ist die Syntax?

Die Syntax ist die Rechtschreibung. Bei Codeblöcken müssen immer ⇥ verwendet werden. Für jeden weiteren Block, wie bei `if` oder `for` muss ein weiter Block hinzugefügt werden.

Variablen

Meistens müssen wir Variablen definieren. Variablen sind so etwas wie Namensschilder, also können wir wichtige zahlen oder anderes z.B. Pin-nummern mit einem Namen versehen um sie besser wiederzuerkennen und der Code übersichtlicher bleibt Es gibt verschiendene Variablentypen.

Variablentyp	**Beispiel**	**Beispielsinhalt**
String	var = "Inhalt"	Inhalt
Int	var = 42	42
Float	var = 3.14	3,14

Informationen ausgeben

Wenn du z.B. Informationen oder Variablen ausgeben möchtest, verwendest du den Befehl `print` .

```
1    print("Hallo Welt!")
```

Das Beispiel gibt dann **Hallo Welt!** aus.

Kommentare

Kommentare kannst du folgendermaßen eintragen.

```
1    # Gib "Hallo" aus
2    print("Hallo")
```

In diesem Beispiel ist jetzt die grüne Zeile ein Kommentar und wird nicht als Programm ausgeführt.

© 2018 Daniel Bauer, Teckids e.V.
Veröffentlicht unter der *Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 3.0 Deutschland* Lizenz.

Listen

In Listen kannst du alles speichern. Hier kannst du andere Variablen, Inhalte oder sogar andere Listen speichern. Du kannst eine leere Liste folgendermaßen definieren:

```
1       liste = []
```

Oder eine gefüllte Liste:

```
1       liste = [18, 506, 396, "Teststring"]
```

Diesen Listeninhalt kannst du folgendermaßen auslesen:

```
1       liste = [18, 506, 396, "Teststring"]
2       stelle = 3
3       var = liste[stelle];
4       print(var)
```

Das Obige würde `"teststring ausgeben"` Nun speichern wir das vierte element der Liste in der Variable `var`. Das vierte, weil der Computer beim Zählen immer mit 0 anfängt,

Generelles zu Modulen

Wenn du mehr Funktionen als die Standart-Pythonfunktionen nutzen möchtest, musst du sog. Module importieren. Dies machst du mit `import ModulName`. Gängige Module sind:

- `time`
- `machine`
- `random`

Pins Anschalten und Ausschalten

Nun willst du sicherlich einen Pin anschalten oder aus. Das machst du mit dem eben genannten Modul `machine` . Zuerst musst du aber einen Pin als Ein oder Ausgang definieren.

```
1    #Importiere das Modul machine
2    import machine
3    pinnummer = 2
4    #Deklariere Pinobjekt als Ausgang auf Pin 2
5    pinobjekt = machine.Pin(pinnummer, machine.Pin.OUT)
```

So schaltet man einen Pin an:

```
1       pinobjekt.off()
```

© 2018 Daniel Bauer, Teckids e.V.
Veröffentlicht unter der *Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 3.0 Deutschland* Lizenz.

Und so aus:

```
        pinobjekt.on()
```

Du magst dich jetzt vielleicht wundern, dass du mit `off` den Pin anschaltest, das liegt aber einfach an der internen verschaltung des Esps.

Operatoren

Im gleich folgenden Kapitel geht es um Bedingungen, also um Vergleiche. Dafür benötigen wir Operatoren.

==	beides soll gleich sein
>=	wenn die Variable größer oder gleich sein soll als der Vergleichswert
<=	wenn die Variable kleiner oder gleich sein soll als der Vergleichswert
>	wenn die Variable größer sein soll
<	wenn die Variable kleiner sein soll
!=	wenn die Variable ungleich sein soll
!>	wenn die Variable ungleich oder größer sein soll
!<	wenn die Variable ungleich oder kleiner sein soll

Bedingungen

Wenn du möchtest, dass etwas nur dann ausgeführt wird, wenn eine Bedingung erfüllt wird, kannst du die if-Bedingungen benutzten:

```
        if (variable operator vergleichswert):
                #Mach dann das hier
```

Wenn die Begingung erfüllt wird, wird der im Codeblock stehende Code ausgeführt, wenn nicht wird er übersprungen. Das könnte dann so aussehen:

```
    #Wenn der Schalter an ist
        if schalter == True:
        #Schalte die Led an
                led.off()
```

Wir können die Led auch ansonsten ausmachen wenn das nicht zutrifft und das geht mit else:

```
        if schalter == True:
        #Schalte die Led an
                led.off()
    else:
        #Schalte die Led aus
                led.on()
```

© 2018 Daniel Bauer, Teckids e.V.
Veröffentlicht unter der *Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 3.0 Deutschland* Lizenz.

Schleifen

Sicher willst du irgendwann, dass etwas immer wieder wiederholt wird wärend die Bedingung erfüllt ist. Dafür gibt es while.

```
1       while Bedingung:
2               #Mach dann das hier
```

Wenn du eine Endlosschleife programmieren willst, kannst du einfach schreiben:

```
1   while True:
2       #Mach das hier unendlich lange...
```

Zur Kurzen Erklärung: Wenn du, wie eben gezeigt `while Bedingung` schreibst, wird die Schleife so lange ausgeführt, solange die Bedingung Wahr ist, also `True` ist. So hast du quasi das Ergebnis manipuliert, sodass es direkt `True` ausgibt.

Dann gibt es noch die for-Schleife. Mit ihr können wir einen Vorgang beliebig oft wiederholen ohne, dass wir es mermals schreiben müssen:

```
1       for i in range(0,5):
2               #Mach dann das hier
3       }
```

Diese Schleife wird nun 6 mal wiederholt, denn ein Computer beginnt immer mit 0 beim zählen.

Pausen

Wenn wir z.B eine Led blinken lassen wollen, muss sie eine kurze Pause machen“ bevor sie wieder an oder aus geht. Dafür nutzen wir die Funktion `time.sleep(wartezeit)` des Moduls `time`. So lassen wir etwas blinken:

```
1   import time
2   import machine
3   led = machine.Pin(2, machine.Pin.OUT)
4   wartezeit = 1
5   #Dauerschleife
6   while True:
7       #Schalte die Led an
8       led.off()
9       #Warte 1 sec.
10      time.sleep(wartezeit)
11      #Schalte die Led aus
12      led.on()
13      #Warte 1 sec.
14      time.sleep(wartezeit)
```

© 2018 Daniel Bauer, Teckids e.V.
Veröffentlicht unter der *Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 3.0 Deutschland* Lizenz.

Drück einen Knopf

Sicher möchtest du irgendwann, dass der Esp etwas tut wenn du einen Knopf drückst. So kannst du das machen:

```
1    pin = 3
2    taster = machine.Pin(pin, machine.Pin.IN)
3        tasterstatus = taster.value();
4        if tasterstatus == True:
5        #Mach etwas
```

Hierbei brauchst du allerdings in der Elektronik einen sog. `Pullup/Pulldownwiederstand`. Wenn du dies umgehen möchtest kannst du auch einfach die `Pullupwiederstände` des Esp nutzen. Dies kannst du so realisieren.

```
1    pin = 3
2    #Hier ist das wichtige!
3    taster = machine.Pin(pin, machine.Pin.PULL_UP)
4    tasterstatus = taster.value()
5    #Wenn der Taster gedrueckt wird
6    if tasterstatus == False:
7        #Mach etwas
```

Durch die Verschaltung des Pullupwiederstandes wird bei schließen des Kontaktes der ESP-Pin auf `False` gezogen und umgekehrt.

Analog auslesen

So jetzt haben wir schon mal einen Schalter oder Taster ausgelesen. Jedoch kann man auch noch Sensoren Auslesen. Wer hätte es gedacht wir benutzen dafür die Funktion `ADC` des Moduls `machine`. Dabei werden Werte zwischen 0 und 1024 ausgegeben.
⚠ **Pass aber auf**, dass der Eingang maximal 1V Spannung bekommt!

```
1    sensorpin = 4
2    sensor = machine.ADC(pin)
3        sensordaten = sensor.read()
```

Funktionen

Nun möchtest du zum Beispiel mehrere Befehle nacheinander öfters ausführen. Dafür verwendest du Funktionen. Mit der folgenden Funktion lassen wir einen Ausgang 2 sec. an und 2 sec. aus gehen.

© 2018 Daniel Bauer, Teckids e.V.
Veröffentlicht unter der *Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 3.0 Deutschland* Lizenz.

```
def blinken(pin):
        pin.off()
        time.sleep(2)
    pin.on()
    time.sleep(2)
```

In den Klammern steht die für nur diese Funktion geltende Variable Pin. Aufrufen können wir diese Funkion nun mit:

```
        blinken(2)
```

Nun blinkt die interne LED auf Pin 2.

Wollen wir nun auch z.B. eine Funktion schreiben, die x und y multipliziert machen wir das so.

```
x = 2
y = 3

def multplizieren(x, y):
        ergebnis = x * y
        return ergebnis
}
```

Wie du merkst wird nun, da wir einen Rückgabewert haben (Ergebnis), statt dem void ein Variablentyp hingeschrieben. Diese Funktion rufen wir nun so auf:

```
    #Rechne aus
        washabenwirnunraus = multiplizieren(x, y)
    #Gib Ergebnis aus
    print(washabenwirnunraus)
```

Nun hat die Variable washabenwirnunraus den Wert 2*3 also 6. Was in den Klammern für Variablen stehen ist egal (in unserem Fall müssen es zahlen sein). Das x und y in der Funktionsklammer steht für die interne Variablen. Deshalb können wir die Funktion auch so aufrufen:

```
        washabenwirnunraus = multiplizieren(4, 6)
    print(washabenwirnunraus)
```

In diesem Fall haben wir die Werte mit denen gerechnet werden soll von Hand und nicht per Variable reingeschrieben. In diesem Beispiel muss 4*6 = 24 rauskommen.

© 2018 Daniel Bauer, Teckids e.V.
Veröffentlicht unter der *Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 3.0 Deutschland* Lizenz.

Led dimmen

So, wie können wir bisher mit unserer Umgebung interagieren? Wir können digitale und analoge Sensoren auslesen und digitale Ausgänge ansteuern. Nun fehlt es nur noch, dass wir unsere IO-Pins analog ansteuern können und so z.B. eine Led dimmen können. Und das geht z.B. so:

```
1   import machine
2   import time
3   #Pinobjekt erstellen
4   led = machine.Pin(2)
5   #PWMObjekt mit einer Frequenz von 1000Hz erstellen
6   ledpwm = machine.PWM(led, freq=1000)
7   #Die Led auf 100% aufdrehen
8   ledpwm.duty(1023)
9   #Eine sec warten
10  time.sleep(1)
11  #Die Led auf 50% runterdimmen
12  ledpwm.duty(512)
13  #Eine sec warten
14  time.sleep(1)
15  #Die Led ausschalten
16  ledpwm.duty(0)
```

So, wie du gesehen hast können wir die Led in einem Bereich von **0** (0%) - **1023** (100%) dimmen. Die Technik dafür heißt PulseWideModulation. Dabei wird mit einer festgelegten Frequenz (im Beispiel 1000Hz) die Led ganz schnell an und aus geschaltet. Dies regeln wir über den `Dutycycle`. Der Dutycycle gibt den Anteil der An- und Ausphasen in dieser festgelegten Zeit an. Durch die Trägheit des Auges wirkt dies, wie das dimmen der Led. Natürlich kannst du auch mit dieser Technik und einer RGB-Led alle beliebigen Farben mischen.

© 2018 Daniel Bauer, Teckids e.V.
Veröffentlicht unter der *Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 3.0 Deutschland* Lizenz.