

Lektion 6: Arbeitsblatt 6.1 - Blinken der LED als Reaktion auf ein Klatschen

Bei dieser Aufgabe musst du ein Programm schreiben, das den Klatschsensor von Edison verwendet, um den Roboter dazu zu bringen, eine LED-Lampe aufleuchten zu lassen, wenn er ein Klatschen erkennt.

Als Erstes muss das Programm geplant werden.





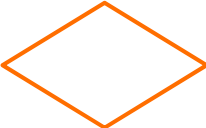
Flussdiagramme in der Programmierung

Professionelle Programmierer planen in der Regel ihr Programm, bevor sie mit dem Schreiben des Codes beginnen. Die Verwendung von Flussdiagrammen ist eine Möglichkeit für Programmierer, ihre Programme zu organisieren und zu planen.

Die Idee eines Flussdiagramms ist es, grafisch zusammenzufassen, was im Code passiert, ohne auf alle Details eingehen zu müssen. Mithilfe von Flussdiagrammen kann ein Programmierer visualisieren und kommunizieren, wie der "Fluss" des Programms funktionieren wird.

In einem Flussdiagramm wird ein Programm durch verschiedene Formen und Pfeile dargestellt. Jede Form steht für ein anderes Element im Programm, und die Pfeile zeigen, wie die Elemente zusammenarbeiten.

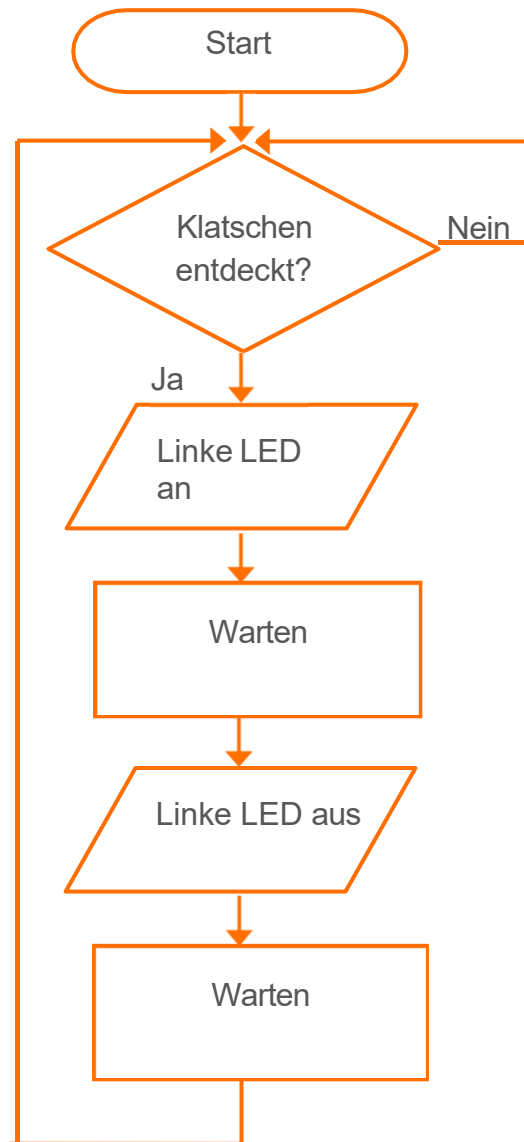
Es gibt fünf Hauptsymbole, die in Flussdiagrammen verwendet werden:

Symbol	Name	Funktion
	Terminator (Anfang/Ende)	Ein Oval stellt einen Start- oder Endpunkt dar
	Pfeil	Eine Linie, die als Verbindungselement dient und die Beziehungen zwischen repräsentativen Formen aufzeigt
	Eingabe/Ausgabe	Ein Parallelogramm stellt eine Eingabe oder eine Ausgabe dar
	Prozess	Ein Rechteck stellt einen Prozess oder eine Aktion dar
	Entscheidung	Ein Diamant steht für eine Entscheidung

Bei der Erstellung eines Flussdiagramms zur Planung eines Programms werden oft Wörter innerhalb der Formen oder neben den Pfeilen hinzugefügt. Diese Wörter sind kurze Zusammenfassungen des Prozesses oder der Entscheidung.

Schauen wir uns ein Beispiel-Flussdiagramm an, in dem das Programm, das wir erstellen wollen, zusammengefasst ist.

Hier ist ein Flussdiagramm für ein Programm, das deinen Edison-Roboter anweist, auf ein Klatschen zu warten und dann die linke LED aufleuchten zu lassen:



Dieses Programm verwendet den Geräuschsensor von Edison, um festzustellen, ob ein Klatschen stattgefunden hat oder nicht. Das Ergebnis bestimmt, wie der Code weiterläuft.

Wenn du dir dieses Flussdiagramm ansiehst, fällt dir vielleicht auf, dass es keinen "End"-Terminator hat. Das liegt daran, dass dieses Programm eine "while"-Schleife verwendet, die so aufgebaut ist, dass das Programm unbegrenzt fortgesetzt werden kann.

Herstellung einer Endlosschleife

Manchmal möchten Sie vielleicht ein Programm schreiben, das kein Ende hat, sondern eine Endlosschleife bildet. In der Programmierung wird dies oft als Endlosschleife bezeichnet.

Du kannst eine Endlosschleife verwenden, um das Programm im Flussdiagramm zu planen.

Sieh dir das folgende Programm an:

```
1
2 #-----Setup-----
3
4 import Ed
5
6 Ed.EdisonVersion = Ed.V2
7
8 Ed.DistanceUnits = Ed.CM
9 Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #-----Your code below-----
12 while True:
13     Ed.ReadClapSensor()
14     while Ed.ReadClapSensor() != Ed.CLAP_DETECTED:
15         pass
16     Ed.LeftLed(Ed.ON)
17     Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
18     Ed.LeftLed(Ed.OFF)
19     Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
20
```

Dieses Programm wird durch unser Flussdiagramm dargestellt und enthält eine Endlosschleife. Schau dir die Zeile 12 des Programms an, in der eine "while"-Schleife verwendet wird.

While"-Schleifen benötigen immer eine Bedingung. Die Schleife wiederholt jeden eingerückten Code, solange die Bedingung "wahr" ist.

Wenn wir wollen, dass die 'while'-Schleife unendlich oft wiederholt wird, geben wir keine Bedingung an, die das Programm auswerten muss, sondern wir ersetzen die Bedingung durch 'True'.

"True" wird immer zu "True" aufgelöst. Indem wir die Bedingung auf "True" setzen, haben wir die Bedingung unserer "while"-Schleife fest auf "True" kodiert.

Du bist dran:

Schreibe den obigen Code, um deinen Edison-Roboter so zu programmieren, dass die linke LED blinkt, wenn du klatschst. Lade ihn herunter und teste, wie er funktioniert.

Name _____

1. Wie weit kannst du von deinem Edison entfernt sein, ohne dass der Roboter merkt, wenn du klatschst?

2. Welchen Zweck haben die TimeWait()-Funktionsaufrufe im Code? Was würde passieren, wenn wir sie nicht hätten? *Tipp:* Versuche, das Programm ohne die TimeWait()-Funktionsaufrufe auszuführen.

3. Schau dir das Programm und das Flussdiagramm an und vergleiche, wie sich der Code im Programm und das Flussdiagramm zueinander verhalten. Erkläre, was im Code passiert, wenn das Ergebnis der im Flussdiagramm dargestellten Entscheidung "nein" ist.

Versuchen Sie es!

Kannst du das Programm so ändern, dass beide LEDs aufleuchten, wenn du klatschst?

Lektion 6: Arbeitsblatt 6.2 - Fahren als Reaktion auf ein Klatschen

Bei dieser Aufgabe musst du ein Programm schreiben, das deinen Edison-Roboter auf ein Klatschen hin vorwärts fahren lässt.

Du bist dran:

Aufgabe 1: Fahre vorwärts, wenn ein Klatschen erkannt wird. Schreibe das folgende Programm und führe es aus:

```
1  #-----Setup-----
2
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #-----Your code below-----
12 Ed.ReadClapSensor()
13 while Ed.ReadClapSensor() == Ed.CLAP_NOT_DETECTED:
14     pass
15 Ed.Drive(Ed.FORWARD, Ed.SPEED_8, 10)
16
```

1. Warum ist es notwendig, in Zeile 12 eine erste Abfrage des Klatschsensors durchzuführen? Was bewirkt dies? *Tipp:* Schau noch einmal auf Arbeitsblatt 2.5.

Aufgabe 2: Fahre vorwärts und dann rückwärts, wenn ein Klatschen erkannt wird.

Der Geräuschsensor des Edison-Roboters reagiert nicht nur auf Klatschen. Die Sensoren können auf jedes laute Geräusch reagieren. Deshalb kannst du in der Nähe des Sensors auf den Roboter tippen, um den Tonsensor auszulösen.

Edisons Motoren, Zahnräder und Räder machen Geräusche, wenn sie sich drehen, was den Geräuschsensor auslösen kann. Um zu verhindern, dass das Geräusch des fahrenden Roboters den Geräuschsensor auslöst, musst du das Programm ändern.

Du musst einen `TimeWait()`-Funktionsaufruf mit einem Eingabeparameter von etwa 350 Millisekunden hinzufügen, um den Motoren des Roboters Zeit zum Anhalten zu geben.

Schreibe das folgende Programm in EdPy:

```
1
2 #-----Setup-----
3
4 import Ed
5
6 Ed.EdisonVersion = Ed.V2
7
8 Ed.DistanceUnits = Ed.CM
9 Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #-----Your code below-----
12 Ed.Drive(Ed.FORWARD,Ed.SPEED_8,10)
13
14 Ed.TimeWait(350,Ed.TIME_MILLISECONDS)
15 Ed.ReadClapSensor()
16 while Ed.ReadClapSensor() == Ed.CLAP_NOT_DETECTED:
17     pass
18 Ed.Drive(Ed.BACKWARD,Ed.SPEED_8,10)
19
```

Lade das Programm herunter und teste es.

2. Normalerweise schreiben wir ein Flussdiagramm, das uns bei der Planung unseres Codes hilft. Zeichne zum Üben ein Flussdiagramm, das dem Code entspricht, den du gerade geschrieben hast. Zeichne dein Flussdiagramm unten ein oder erstelle ein Flussdiagramm am Computer.

Lektion 6: Arbeitsblatt 6.3 - Entwirf deine eigene Funktion

In dieser Aktivität wirst Du deine eigene Funktion entwerfen und sie verwenden, um ein Programm für Edison zu schreiben.

Was genau sind Funktionen?

Nachdem Du nun eine Weile mit Edison und EdPy programmiert hast, hast du eine Reihe von verschiedenen Funktionen aus der EdPy-Bibliothek verwendet.

Wie du weisst ist eine Funktion ein Teil des Codes, der eine bestimmte Rolle oder Aufgabe im Programm übernimmt, je nachdem, welche Eingabeparameter verwendet werden.

Aber vielleicht hast du nicht bemerkt, dass alle Funktionen, die du bisher verwendet hast, tatsächlich mehrere Codezeilen ausgeführt haben, wenn sie vom Programm ausgeführt wurden.

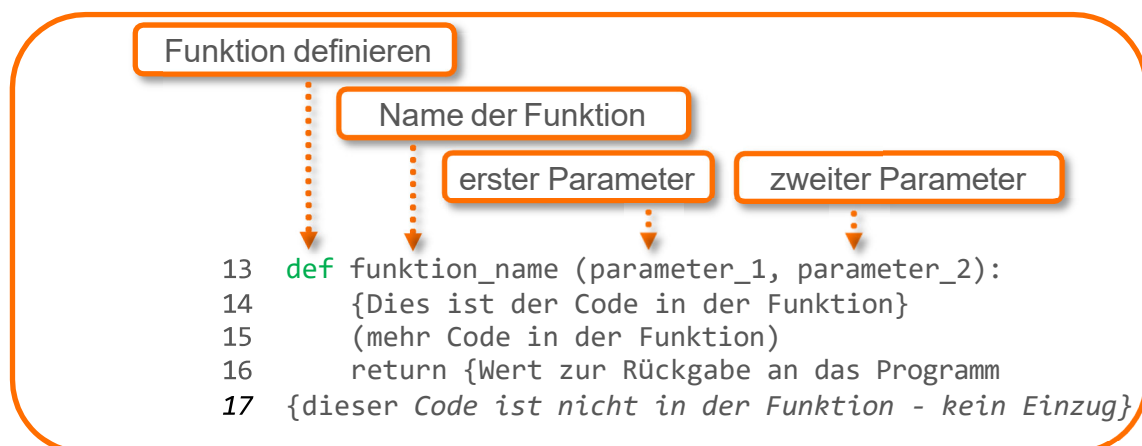
Das liegt daran, dass eine Funktion ein Block von organisiertem, wiederverwendbarem Code ist, der zur Durchführung einer einzigen, zusammenhängenden Aktion verwendet wird.

Funktionen sind sehr hilfreich, weil sie es uns ermöglichen, modular zu programmieren und denselben Codeblock an verschiedenen Stellen im Programm zu verwenden. Damit dein Programm all diese Codezeilen ausführen kann, musst du nur eine Zeile eingeben: den Aufruf der Funktion. Funktionen machen die Wiederverwendung von Code einfacher und effizienter, wenn wir programmieren.

Eigene Funktion erstellen

Bis jetzt haben wir jedes Mal, wenn wir eine Funktion in EdPy verwendet haben, diese Funktion aus der Ed-Bibliothek aufgerufen. Du kannst auch Ihre eigenen Funktionen erstellen.

In Python sehen Funktionen wie folgt aus:



Alles, was eingerückt ist, ist ein Teil der Funktion. Alles, was nicht eingerückt ist, ist nicht Teil der Funktion, sondern die nächste Codezeile im Programm.

Es ist wichtig zu beachten, dass die von Ihnen erstellten Funktionen sich nicht von den Funktionen unterscheiden, die Sie bisher in der Ed-Bibliothek verwendet haben.

Wenn du deine Funktion (mit oder ohne Parameter) aufrufen, springt das Programm vom Aufruf zum Code der Funktion (dem eingerückten Code). Das Programm führt dann diesen Code aus, bevor es zu der Zeile zurückkehrt, in der Sie den Aufruf gemacht haben.

Wenn du deine Funktion so programmierst, dass sie einen Wert zurückgibt, wird der Funktionsaufruf durch den Wert ersetzt, den die Funktion zurückgegeben hat, wenn das Programm zu der Zeile zurückkehrt, in der Sie den Aufruf gemacht haben.

Das ist wichtig, weil das Programm immer erst die Funktionen, dann die mathematischen Befehle und dann die Ausdrücke in dieser Reihenfolge auflösen wird.

Organisiere deinen Code

Die Konvention in EdPy ist es, die Definition Ihrer Funktionen am Ende deines Codes zu schreiben, nachdem du deine Funktion bereits in Deinem Programm verwendet haben.

Dadurch wird dein Code schön und übersichtlich. Wenn du deinen Code auf diese Weise organisierst, kannst du alle deine Funktionen, ob du nun deine eigenen verwendest oder Funktionen aus einer Bibliothek aufrufst, im Hauptteil deines Programms auf eine optisch saubere, organisierte Weise schreiben. Deine Funktionsdefinitionen können außerhalb dieses Bereichs, weiter unten im Programm, stehen.

Du bist dran:

Aufgabe 1: Übe die Definition einer Funktion

Sieh dir das folgende Programm an:

```
1
2 #-----Setup-----
3
4 import Ed
5
6 Ed.EdisonVersion = Ed.V2
7
8 Ed.DistanceUnits = Ed.CM
9 Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #-----Your code below-----
12 directionToMove=Ed.SPIN_LEFT
13 speedToMoveAt=Ed.SPEED_8
14 distanceToMove=360
15
16 #call the function
17 moveOnClap(directionToMove, speedToMoveAt, distanceToMove)
18
19
20 #user defined functions
21 def moveOnClap(direction, speed, distance):
22     Ed.ReadClapSensor()
23     while Ed.ReadClapSensor()==Ed.CLAP_NOT_DETECTED:
24         pass
25     Ed.Drive(direction, speed, distance)
26
27
```

Zeile 17 im Programm ist der Funktionsaufruf. Die Zeilen 21 bis 25 dienen der Definition der Funktion.

Schreibe das Programm und lade es auf deinen Edison-Roboter herunter. Führe das Programm aus, um zu sehen, wie es funktioniert.

1. Wir könnten eine Funktion schreiben, um den Edison-Roboter in einer quadratischen Form fahren zu lassen. Ergänze die fehlenden Wörter in der Funktion unten, um diese Funktion fertig zu schreiben.

```
def driveInaSquare():
    for i in range( ):
        Ed.Drive(_____,_____,_____)
        Ed.Drive(_____,_____,_____)
```

2. Schreibe die Syntax für den Code zum Aufruf dieser Funktion. Was würdest du in deinem Programm eingeben, um diese Funktion aufzurufen?

Schreibe ein Programm, das Edison mehrere Quadrate fährt. Du musst die Funktion "driveInaSquare" definieren, und dein Programm muss diese Funktion mehr als einmal aufrufen. Lade das Programm herunter und führe es aus, um zu sehen, wie es funktioniert.

3. Macht das Programm das, was du von ihm erwartet hast? Wenn nicht, beschreibe, was du erwartet hast und was das Programm tatsächlich getan hat. Beschreibe alle Probleme, die du hattest, um dein Programm zum Laufen zu bringen.

Aufgabe 2: Entwirf eine eigene Funktion

Schreibe deine eigene Funktion, die Edison etwas tun lässt, wenn du klatschst. Du könntest Edison tanzen lassen, eine LED blinken lassen, sich im Kreis drehen oder was auch immer du willst!

Schritt 1: Entwurf

Erstelle zunächst ein Flussdiagramm, das dein Programm grafisch zusammenfasst. Entweder zeichnest du dein Flussdiagramm auf Papier oder verwendest ein Programm wie Google Slides. Achte darauf, dass du die richtigen Formen in deinem Flussdiagramm verwendest, um den Start, die Prozesse, die Verzweigungspunkte und den Fluss des Programms darzustellen.

Die Formen, die du benötigst, hängen von deinem Flussdiagramm ab. Du benötigst mindestens die ovale Startform, die rechteckige Prozessform und die rautenförmige Entscheidungsform in deinem Flussdiagramm.

Schritt 2: Code

Programmieren deine Idee in der EdPy-App. Verwenden dein Flussdiagramm als Leitfaden und codiere deine Funktion so, dass sie jeden Schritt enthält, den du in deinem Flussdiagramm festgelegt hast.

Schritt 3: Test

Schreibe ein Testprogramm, das deine Funktion und zusätzlichen Code zum Ausprobieren Ihrer Funktion enthält. Schau dir an, ob deine Funktion so funktioniert, wie du es geplant und erwartet hast. Wenn nicht, überarbeite dein Flussdiagramm und deinen Code und passe ihn gegebenenfalls an. Experimentiere, um zu sehen, was funktioniert!

4. Beschreibe, was deine Funktion bewirkt hat.

5. Beschreibe alle Probleme, die du hattest.
