```
wait_for_seconds(3)
hub.light_matrix.show_image('SURPRISED')
wait_for_seconds(3)
hub.light_matrix.show_image('SKULL')
wait_for_seconds(3)
```

4.3 Schleifen, Schleifenabbruch und Unterbrechung

Schleifen sind ein wichtiges Konstrukt in der Programmierung, um Programmschritte mehrmals auszuführen. Somit lässt sich eine Schleife folgendermaßen beschreiben:

Eine Schleife ...

- wiederholt die Aktionen in ihrem »Bauch« (Programmsequenzen)
- hat eine Abbruchbedingung
  - Abbruchbedingung führt bei erfolgreicher Prüfung zum Beenden der Schleife
  - Beispiele für mögliche Abbruchbedingungen: unendlich, Anzahl an Durchläufen, Sensorwerte
- kann eine weitere Schleife im »Bauch« beinhalten (Schachtelung von Schleifen)
- wird benötigt, um Aktionen (Programmsequenzen) mehrfach auszuführen, ohne diese im Programm kopieren zu müssen

Eine Schleife endet typischerweise mit einer Abbruchbedingung. In der Programmierung sind Schleifen mit vorgestellter oder nachgestellter Abbruchbedingung möglich, sodass entweder die zu wiederholende Programmsequenz mindestens einmal (bei nachgestellter Abbruchbedingung) oder gegebenenfalls gar nicht ausgeführt wird (bei vorgestellter Abbruchbedingung). Eine Abbruchbedingung wird immer erst geprüft, wenn das Programm den Prüfschritt erreicht. Somit kann man nicht davon ausgehen, dass eine Schleife sofort bei Erreichen des Abbruchereignisses beendet wird.

Es gibt auch die Möglichkeit, eine Schleife an jeder beliebigen Stelle abzubrechen. Dies ist allerdings von der jeweiligen Programmiersprache abhängig, da ein solcher Abbruch nicht in jeder Sprache ohne Probleme beziehungsweise ohne tiefere Kenntnis von nebenläufiger Programmierung (zum Beispiel Thread) möglich ist.

Diese verschiedenen Möglichkeiten und Einsatzgebiete für Schleifen werden bei der Lösung der folgenden Aufgaben deutlich:

ID	Aufgabenstellung
A431	Roboter fährt ein Viereck (Umsetzung mit einer Schleife) mit circa 20 cm Seitenlänge.
A432	Roboter fünfmal ein Stück (20 Grad Motordrehung) vorwärtsfahren und nach jeder Fahrsequenz 1 Sekunde warten lassen.
A433	Erweiterung des zweiten Programms, sodass nach dem fünften Mal ein Ton ausgegeben wird und dies alles solange wiederholt wird, bis der Ultraschallsensor eine Entfernung kleiner als 30 cm gemessen hat.
A434	Darstellung einer Animation von Punkten auf der Lichtmatrix, indem die Eckpunkte zum Zentrum animiert werden. Die Animation soll 5-mal durchlaufen werden.

Für die erste Aufgabe A431 kommen dabei das erste Mal die Überlegungen zum Tragen, wie der Roboter eine feste Strecke und eine konkrete Drehung vollziehen kann. Dies ist mit einfacher Physik oder mit erweiterten Sensoren zu beantworten. Deshalb gehen wir zunächst auf die Physik ein.

Die Strecke, die ein fahrendes Objekt zurücklegt, hängt von der Drehung und der Größe der Räder ab. Die gefahrene Strecke lässt sich einfach berechnen, indem die Anzahl der Drehungen mit dem Radumfang multipliziert wird. Der Radumfang berechnet sich aus Durchmesser × Kreiszahl PI (ungefähr 3,14). LEGO druckt die Reifengröße auf die Reifen auf, sodass diese wie beim Fahrrad oder Auto abgelesen werden können. Die Angabe ist dabei in Durchmesser × Breite in Millimeter angegeben. Das im Mindstorms/Spike-Standardkasten enthaltene Rad hat einen Durchmesser von 57 mm. Um somit eine Strecke von 20 cm zurückzulegen, sind 20 cm/(5,7 cm × π) Umdrehungen notwendig. Dies sind circa 1,1169 Umdrehungen oder 402 Grad. Grundsätzlich sind diese Überlegungen für weitere Varianten sinnvoll und notwendig, allerdings bietet der Mindstorms und Spike bereits die Möglichkeit, eine feste Fahrstrecke in Zentimeter oder Inch zurückzulegen. Dies funktioniert jedoch nur, wenn keine Übersetzung mittels Zahnräder verbaut ist.

Im Unterricht wird eher das Prinzip »Trial and Error« (sprich: Versuch und Irrtum) angewendet werden, da dies in der Praxis oftmals die detaillierteren Ergebnisse liefert. Denn die tatsächlich gefahrene Strecke hängt nicht nur von den berechneten Werten ab, sondern auch von der Reibung, dem Untergrund, dem Batterieladestand und mit wie viel Leistung die Motoren angesteuert werden. Die LEGO-Motoren sind zwar ziemlich genau, aber je schneller die Motoren drehen, umso ungenauer werden auch die Messwerte und die Ansteuerung. Somit ist eine große Geschwindigkeit bei genauen Aufgaben meist der falsche Ansatz.

Bei Drehungen eines Roboters können ähnliche Überlegungen angestellt werden, die dabei von der Physik des Roboters abhängig sind und auch davon, wie der Roboter für eine Drehung angesteuert wird. Der Beispielroboter kann auf der Stelle wie ein Panzer drehen. In dem Fall laufen die beiden Motoren gegenläufig (das heißt, das eine Rad dreht vorwärts, das andere Rad dreht rückwärts), oder aber der Roboter bleibt mit einem Rad stillstehen und dreht nur durch eine Radbewegung. Dabei ändert sich der Mittelpunkt der Drehung und damit auch die notwendige Drehbewegung je Rad. Dies wird in den folgenden Diagrammen dargestellt.

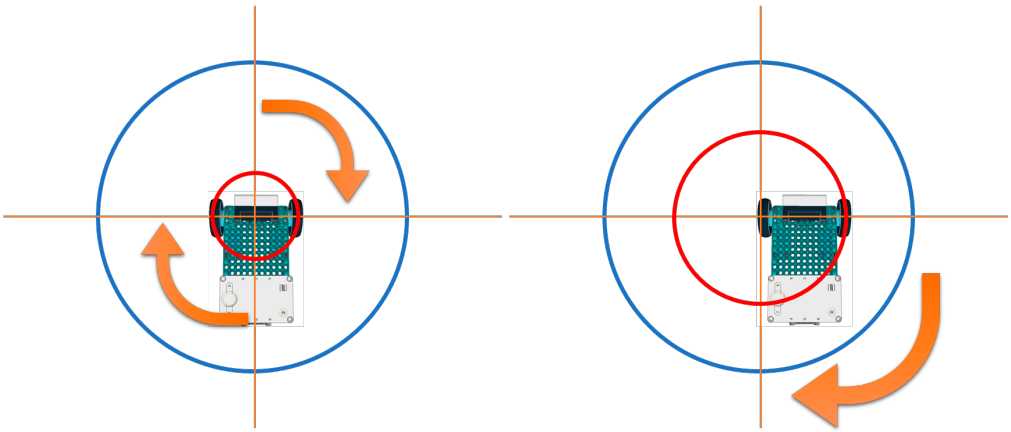


Abb. 4–10 // Unterschiedliche Drehalgorithmen bei fahrenden Robotermodellen

Der rote Kreis verdeutlicht die Strecke, die das jeweilige Rad zurücklegen muss, um eine entsprechende Drehung zu vollziehen. Bei einer Panzerdrehung ist somit eine geringere Drehbewegung eines Rads notwendig, da sich die zurückgelegte Strecke je Rad addiert. Die Strecke pro Rad berechnet sich aus dem Durchmesser des Drehkreises und der gewünschten Drehung – somit Durchmesser × π × ¼ für eine 90-Grad-Drehung.

Der Abstand der Räder im Beispielroboter, die den Durchmesser des Drehkreises bei der Panzerdrehung sowie den Radius des Drehkreises bei der Drehung mit einem Rad darstellen, beträgt circa 11 cm.

Aus der Überlegung der Streckenberechnung zuvor ergeben sich somit die folgenden Werte für die Berechnung einer 90-Grad-Kurve:

	Strecke für die Drehung	Motordrehung
Panzerdrehung	11 cm × π × ¼ = 8,64 cm	8,64 cm/(5,7 cm × π) = 0,48 = 174 Grad
Einraddrehung	2×11 cm × π × ¼ = 17,28 cm	17,28 cm/(5,7 cm × π) = 0,96 = 347 Grad

Mit diesem Ansatz sind die Strecken und Drehungen immer noch mit festen Werten programmiert. Umgebungsbedingte Rahmenbedingungen wie ein rutschiger Untergrund können immer dazu führen, dass die Strecke oder die Drehung ungenau erfolgt.

Um dies besser zu gestalten, ist ein alternativer Ansatz über weitere Sensoren ebenfalls möglich.

Ein Ultraschallsensor kann die Entfernung von einer Wand messen. Wenn somit immer eine Wand für die Entfernungsmessung vorhanden ist, kann der Unterschied zur letzten Messung für eine Fahrt von 20 cm genutzt werden.

Für die Drehung bietet sich der Kreiselsensor (Gyrosensor) im Hub an. Dieser misst die Winkelgeschwindigkeit und kann diese in einen Drehwinkel umrechnen. Auch hier ist die Fahrgeschwindigkeit von entscheidender Bedeutung für die Genauigkeit der Drehung. Dieser wird in späteren

Kapiteln intensiver genutzt, um die Drehung des Roboters unabhängig von seiner Bauart korrekt durchzuführen. Um die Komplexität in diesem Abschnitt geringer zu halten, wird erstmal darauf verzichtet.

4.3.1 Umsetzung mit Textblöcken

Eine Schleife bei den Textblöcken wird durch einen der folgenden Klammerblöcke repräsentiert. Diese befinden sich bei den orangen Elementen, die die Ablaufsteuerung im Programm darstellen.

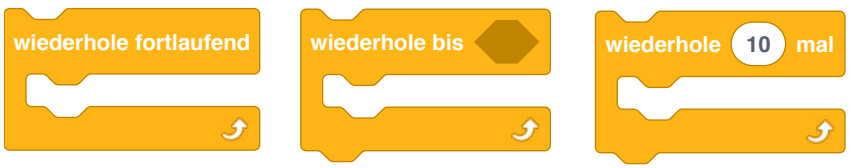


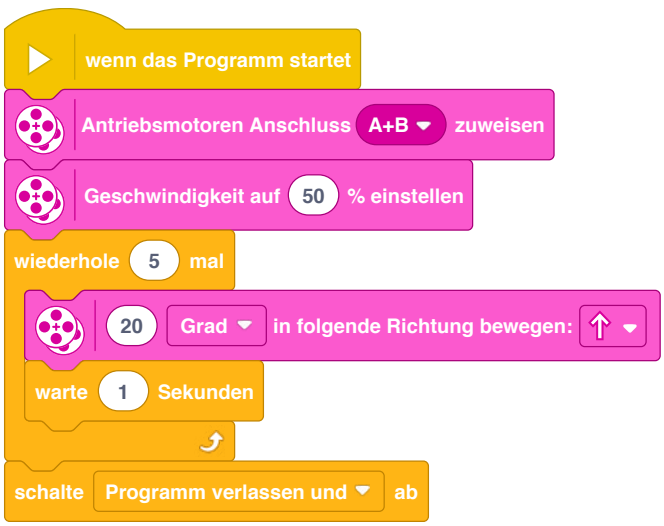
Abb. 4–11 // Unterschiedliche Blöcke für Schleifen als Klammerblöcke

Textblöcke unterstützen nur Schleifen mit vorgelagerter Abbruchbedingung. Somit wird die Programmsequenz innerhalb der Schleife bei Erreichen der Abbruchbedingung nicht mehr ausgeführt und die Abbruchbedingung immer zu Beginn eines Schleifendurchgangs geprüft. Wenn die Prüfung der Abbruchbedingung wahr (true) ist, wird die Schleife beendet. Solange sie falsch (false) ist, wird die Schleife erneut durchlaufen (»Führe die Schleife so lange aus, bis das Ereignis eintritt!«). Dies entspricht einer sogenannten Repeat-Until-Schleife. Dies ist sehr wichtig, da die Schleifenprüfung im Gegensatz zu der in diesem Buch diskutierten textuellen Programmierung umgedreht verstanden wird.

Die Abbruchbedingung ist bei den Textblöcken grundsätzlich beliebig möglich. Eine vorgefertigte Abbruchbedingung stellt einen Zähler dar und wird durch den Textblock »wiederhole x-mal« repräsentiert. Weiterhin gibt es eine sogenannte Endlosschleife, welche die enthaltene Programmsequenz unendlich oft ausführt (»wiederhole fortlaufend«). Ein wichtiger Aspekt ist die Abbruchbedingung aufgrund eines logischen Werts (wahr oder falsch). Damit lässt sich die Abbruchbedingung innerhalb der Schleife oder bei der Prüfung aufgrund einer Logik berechnen und als Abbruchbedingung nutzen. Dafür kann z. B. ein Vergleich mit einem Sensorwert genutzt werden, wie er in Aufgabe A433 notwendig wird.

Die Aufgabe A431 ist dabei über eine Schleife, die 4-mal durchlaufen wird, zu lösen und den Roboter immer ein Stück vorwärtsfahren und die 90-Grad-Drehung durchführen lässt. Hierbei müssen die ersten Fragen einer sinnvollen Drehung um 90 Grad beachtet werden, die in dem Fall durch einfache Versuche ermittelt werden können. Das folgende Programm liefert dabei ein passendes Ergebnis für den Beispielroboter. Die Abbruchbedingung ist dabei ein Zähler und wird nach dem vierten Durchlauf erreicht – die Schleife wird damit beendet.

Für die Aufgabe A432 wird die Vorwärtsbewegung des Roboters auf Grad umgestellt, wobei 20 Grad Motordrehung genutzt werden. Anschließend wird mit dem Warteblock eine Sekunde gewartet und dies mithilfe der Schleife 5-mal durchgeführt.



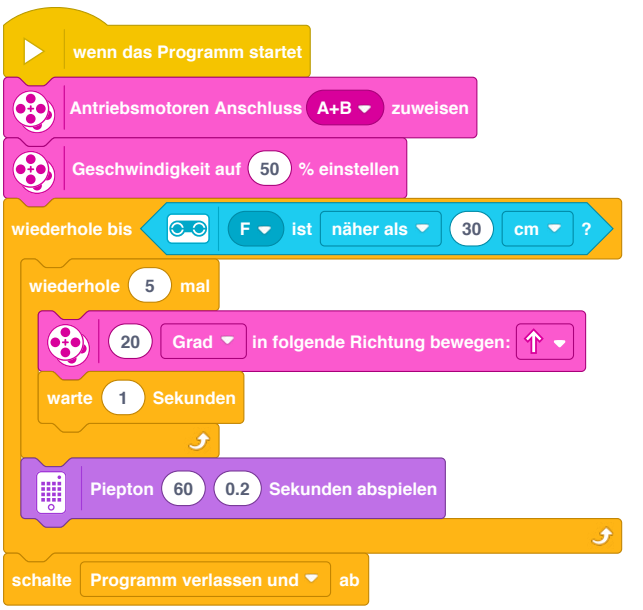
Die Aufgabe A433 ist etwas komplexer und erfordert verschachtelte Schleifen. Dabei gibt es eine innere Schleife, die das Programm aus der zweiten Aufgabe darstellt. Die äußere Schleife beinhaltet den Aufruf der inneren Schleife und zusätzlich den Tonblock zur Ausgabe des Tons. Die Abbruchbedingung der äußeren Schleife wird dabei auf die Entfernungsmessung des Ultraschallsensors umgestellt. Dabei soll die Schleife so lange laufen, bis die Entfernungsmessung weniger als 30 cm misst. Somit muss die Kontrolle auf kleiner 30 cm gestellt werden. Für die Aufgabe A414 wurde bereits die Abbruchbedingung für eine Farbe genutzt. Im türkisen Bereich findet sich auch

eine Ereignisabfrage für den Ultraschallsensor. Dabei lässt sich neben dem Port zusätzlich die Bedingung (»näher als«, »weiter als«, »genau bei«) sowie die Maßeinheit (»cm«, »%«, »Zoll«) wählen. Die Prozentangabe bezieht sich damit auf die maximale Entfernungsmessung von 200 cm.

Abb. 4–12 // Auswahl des Vergleichsoperators bei Textblöcken

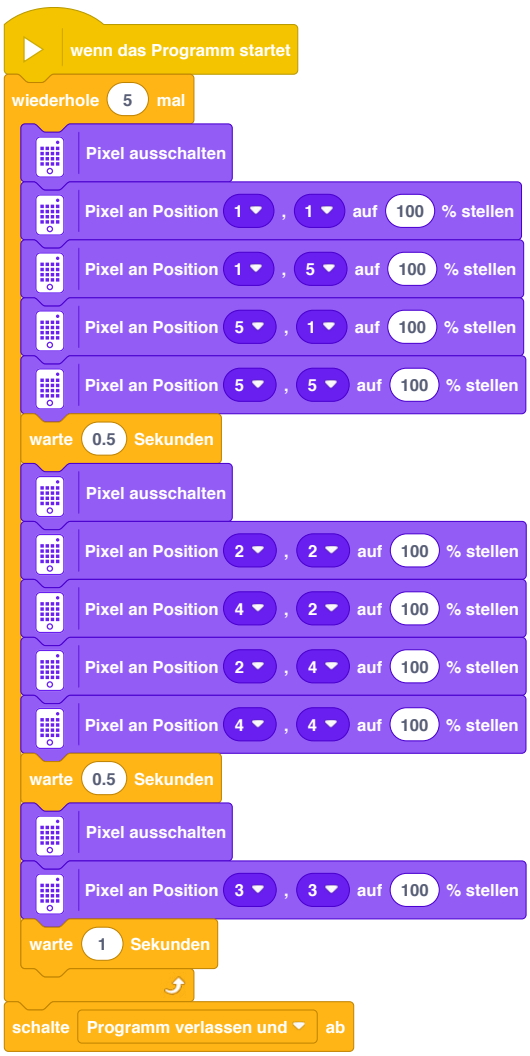
Für die Bedingung ist somit folgende Konfiguration notwendig.

Aufbauend auf diesen Angaben sieht eine mögliche Lösung der Aufgabe A433 folgendermaßen aus:



Die Abbruchbedingung wird immer beim Schleifenbeginn geprüft. Im Beispiel der dritten Aufgabe bedeutet dies, dass die Entfernungsmessung erneut erst nach Abspielen des Klangs stattfindet. Der Roboter bleibt nicht stehen, wenn er innerhalb der inneren Schleife (20-Grad-Schritte) die 30 cm Entfernung zu einer Wand unterschreitet. Dies ist eine wichtige Erkenntnis, da für eine permanente Kontrolle von Ereignissen andere Techniken notwendig sind (zum Beispiel parallele Ausführung von Abfragen oder Abfragen zu dedizierten Zeitpunkten).

Die Aufgabe A434 baut auf der Aufgabe A423 auf und zeigt eine kleine Animation auf der Lichtmatrix mithilfe einer Schleife. Dabei wird nach jedem Aktivieren bestimmter Pixel eine Wartezeit eingeplant, die Lichtmatrix gelöscht und anschließend der nächste Animationsschritt gezeigt. Dabei laufen die Punkte von außen nach innen und simulieren so einen Animationsablauf.



Eine Schleife kann bei den Textblöcken theoretisch auch abgebrochen werden. Dafür steht der Textblock zum Abschalten eines Stapels im orangen Bereich zur Verfügung. Dadurch kann eine Endlosschleife sofort bei Erreichen dieses Textblocks beendet werden.

Allerdings beendet dies den gesamten Stapel inklusive aller darin befindlichen Programmsequenzen sofort. Somit muss, vor allem bei parallelen Programmierpfaden, sehr darauf geachtet werden, wann ein Stapelabbruch erfolgt, da dies das Programm auch in einen undefinierten Zustand bringen kann, wenn die Blöcke in der Schleife nicht korrekt zu Ende gebracht wurden.

Ein kontrollierter Abbruch einer konkreten Schleife und Fortsetzen des Stapels nach der Schleife wird mit Textblöcken nicht unterstützt. Dies müsste mit einem parallel gestarteten Stapel (siehe Kapitel 4.7 und 5.2) umgesetzt werden.

Im grünen Bereich der Palette finden sich weitere Wahrheits- und Wertblöcke, die Bedingungen berechnen können, welche dann in Schleifen oder Schaltern genutzt werden können. Dabei geht es konkret um Vergleiche (größer, kleiner, gleich) von Werten, Bereichsprüfungen (liegt ein Wert zwischen x und y) sowie logische Operationen (UND-, ODER-Verknüpfungen). Diese werden im Kapitel 4.7 detaillierter betrachtet.

4.3.2 Umsetzung mit Python

In diesem Abschnitt wird das Schleifenkonstrukt von Python erläutert und verschiedene Einsatzmöglichkeiten vertieft. Eine Schleife stellt dabei eine grundlegende Anweisung in Python dar. Durch die Eigenschaft von Python, auch in sich verschachtelte Aufrufe zu ermöglichen, bietet Python eine wesentlich größere Bandbreite für Schleifenkonstrukte an, als es mit Textblöcken möglich ist. Im Rahmen dieses Abschnitts werden dabei die wichtigsten angesprochen, um auch die Aufgaben sinnvoll zu lösen.

Es werden in Python zwei Schleifenanweisungen unterschieden:

Python-Anweisungen	Kurzerläuterung
for ... in ... else	Schleife, welche über einen Inhalt iteriert
while ... else	Schleife mit einer generischen Abbruchbedingung

Die for-Schleife in Python unterscheidet sich von anderen textuellen Programmiersprachen, weil die for-Schleife keine Schleife für eine zählbare Wiederholung einer Programmsequenz darstellt. Vielmehr wird mit der for-Schleife über einen Inhalt (z. B. ein Feld, ein Text o. ä.) iteriert und ist

eher mit Iteratoren aus anderen Programmiersprachen vergleichbar. Der Inhalt stellt dabei eine Sequenz von Daten dar. Die folgenden Beispiele sollen dies grundsätzlich verdeutlichen. Beim ersten Beispiel wird jeder Buchstabe des Worts »LEGO« einzeln ausgegeben. Im zweiten Beispiel wird zuerst der Text »R2D2« und anschließend »C3PO« ausgegeben:

```
for c in "LEGO":
    print(c)

robots = ["R2D2", "C3PO"]
for f in robots:
    print(f)
```

Für die vorliegenden Aufgaben ist die for-Schleife von Python nur bedingt einsetzbar.

In Python ist auch eine andere Schleifenart möglich – die sogenannte While-Schleife. Diese stellt die generische und universell einsetzbare Schleifenart in Python dar, bei der eine Abbruchbedingungen notwendig ist, um die Schleife zu beenden. Eine While-Schleife wird durchlaufen, solange eine vorgegebene Bedingung erfüllt ist. Dabei unterstützt Python nur die vorgestellte Prüfung der Abbruchbedingung. Dies bedeutet, dass die Prüfung erfolgt, bevor der Schleifeninhalt ausgeführt wird.

Ein Beispiel verdeutlicht der folgende Programmcode, der aufsteigend eine Zahl ausgibt und die Schleife so lange durchläuft, solange der Zähler kleiner als 5 ist:

```
i=1
while i<5:
    print(i)
    i=i+1
else:
    print("ist größer als 5")
```

Dabei werden das erste Mal auch Variablen eingesetzt, die erhöht werden müssen. Die Details zur Verwendung von Variablen werden in Kapitel 4.5 näher erläutert.

Darüber hinaus wird eine Besonderheit in der Python-Sprache aufgezeigt. Python unterstützt einen else-Block bei einer while-Schleife. Dieser Block wird ausgeführt, wenn die Bedingung nicht mehr erfüllt ist. Der else-Block wird allerdings auch ausgeführt, wenn der Schleifenrumpf nie ausgeführt wurde.

Eine Abbruchbedingung muss den Wert wahr (true) erfüllen. Somit kann darin auch beliebiger alternativer Programmcode ausgeführt werden. Dies erweitert die Einsatzmöglichkeiten der While-Schleife, um zum Beispiel auf ein bestimmtes Ereignis des Hubs oder eines Sensors zu warten:

```
while colorSensor.get_color() != 'red':
    # do something
```

Bei den Textblöcken wird die Schleife so lange wiederholt, bis eine Bedingung erfüllt ist (»wiederhole bis«). In Python wird die Schleife so lange wiederholt, solange die Bedingungsprüfung wahr (true) liefert. Somit sind hier die Abfragen genau umgedreht zu den Textblöcken.

Die ersten beiden Aufgaben lassen sich nach diesen Ausführungen ohne weitere notwendige Erklärungen lösen.

Die Lösung für Aufgabe A431 sieht folgendermaßen aus:

```
motor_pair = MotorPair('A', 'B')
motor_pair.set_default_speed(50)
i = 0
while i < 4:
    motor_pair.move_tank(20, 'cm')
    motor_pair.move_tank(10, 'cm', -50, 50)
    i = i + 1
```

Die Lösung für Aufgabe A432 kann folgendermaßen umgesetzt werden:

```
motor_pair = MotorPair('A', 'B')
motor_pair.set_default_speed(50)
i = 0
while i < 5:
    motor_pair.move_tank(20, 'degrees')
    wait_for_seconds(1)
    i = i + 1
```

Für die Aufgabe A433, bei der die Entfernung des Ultraschallsensors gemessen werden soll, kommt erneut die Klasse `DistanceSensor` zum Einsatz. Nach Initialisierung mit dem korrekten Port kann der Entfernungswert ausgelesen und als Abbruchbedingung in der While-Schleife genutzt werden. Hierbei ist zu beachten, dass die Abbruchbedingung im Vergleich zu den Textblöcken umgedreht werden muss. Bei den Textblöcken wird eine Schleife solange ausgeführt, bis ein Ereignis eintritt und damit die Schleife beendet wird. Bei einer While-Schleife wird die Schleife solange durchgeführt, wie die Abbruchbedingung den Wert wahr (true) liefert. In der Beispiellösung muss somit die Prüfung in der While-Schleife auf eine Entfernung größer 30 cm erfolgen, da die Schleife solange ausgeführt werden soll. Es kann hierbei zu einem Programmfehler kommen, wenn der Ultraschallsensor keinen gültigen Wert liefert. Eine sinnvolle Umgehung dieses Problems wird konkret in Abschnitt 4.7.2 vorgestellt.

```
hub = MSHub()

motor_pair = MotorPair('A', 'B')
motor_pair.set_default_speed(50)
```