Name_____

Lektion 2: Arbeitsblatt 2.1 - Den Roboter vorwärts fahren

In dieser Aufgabe musst du ein Programm schreiben, um deinen Edison-Roboter vorwärts zu bewegen. Schau 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
13 Ed.Drive(Ed.FORWARD,Ed.SPEED_5,8)
14
```

Schritt 1: Gebe "Ed.Drive(" in Zeile 13 ein, und wähle die Funktion "Ed.Drive()".

Ed.Drive() ist eine Funktion in Python, die vom Setup-Code aus dem Edison-Bibliotheksmodul "Ed" importiert wird.

Eine Funktion ist ein Teil des Codes, der je nach den eingegebenen Parametern eine bestimmte Funktion oder Aufgabe ausführt. Alle Funktionen, die aus der Bibliothek "Ed" importiert werden, müssen mit "Ed" beginnen. Damit wird dem Programm mitgeteilt, in welcher Bibliothek es diese Funktion finden kann.

Schritt 2: Fülle die Eingabeparameter aus.

Wenn eine Funktion Eingabeparameter hat, musst du für jeden einzelnen einen Wert eingeben. Die Funktion Drive() hat drei Eingabeparameter:

- **Richtung** - die Richtung, die Edison einschlagen wird
- **Geschwindigkeit** - die Geschwindigkeit, mit der Edison fahren wird
- **Entfernung** - die Anzahl der Entfernungseinheiten, die Edison zurücklegt

Verschiedene Eingabeparameter nehmen unterschiedliche Werte an. Für "Geschwindigkeit" beispielsweise ist Ed.SPEED_ eine Zahl zwischen 1 und 10 (10 ist der Höchstwert).

Name_____

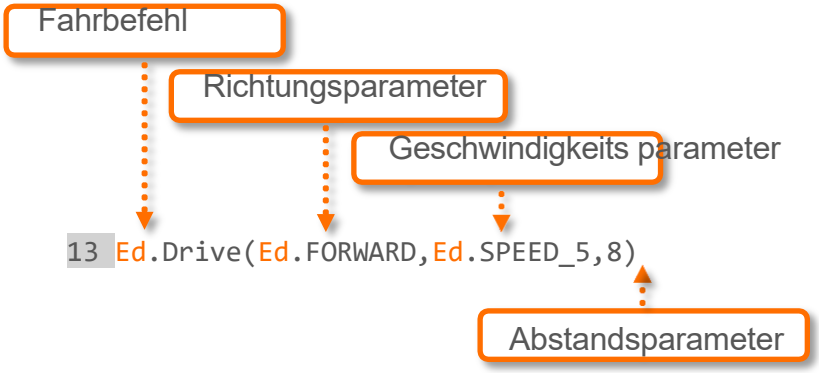
Die Art der Entfernungseinheiten wird durch die Konstante 'Ed.DistanceUnits' im Setup-Code bestimmt. Es gibt drei Entfernungseinheiten, die Sie verwenden können:

- Zentimeter, geschrieben als Ed.CM
- Zoll, geschrieben als Ed.INCH
- Zeit, geschrieben als Ed.TIME

Hinweis:.Die Zeiteinheit für die Entfernung ist Millisekunden, d.h. um 2 Sekunden zu fahren, musst du den Eingabeparameter für die Entfernung auf 2000 einstellen (da 2000 Millisekunden = 2 Sekunden).

Schritt 3: Überprüfe Dein Programm.

Schauen wir uns die Antriebsfunktion und die Parameter in diesem Programm genauer an.



Klicke auf die Schaltfläche "Code prüfen" und schau dir das Fenster "Compiler-Output" an, um sicherzustellen, dass du beim Tippen keine Fehler gemacht hast.

Tippfehler werden als "Syntaxfehler" bezeichnet. Wenn du ein Wort eingibst, das nicht Teil der Sprache des EdPy-Programms ist, auch Syntax genannt, dann kann der EdPy-Compiler nicht verstehen, was es tun soll. Dies erzeugt einen Syntaxfehler.

Schritt 4: Lade Dein Programm herunter und teste es.

1. Schalte den Edison ein und drücke dann einmal die Aufnahmetaste (rund) von Edison.
2. Verbinde Edison über das EdComm-Kabel mit dem Computer und stelle sicher, dass die Lautstärke voll aufgedreht ist.
3. Drücke die Schaltfläche "Edison programmieren" in der oberen rechten Ecke der EdPy-App
4. Folge den Schritten im Pop-up-Fenster und drücke dann auf 'Edison programmieren'.

Sobald das Programm heruntergeladen ist, ziehe das EdComm-Kabel ab. Markieren mit Hilfe des Aktivitätsblatt 2.1 oder eines farbigen Klebebands die Start- und Ziellinie auf einem Schreibtisch oder dem Boden als Testbereich für Dein Programm. Drücke einmal auf die Wiedergabetaste (Dreieck), um das Programm auszuführen und zu beobachten, was passiert.

Name_____

Schritt 5: Experimentiere mit Deinem Programm.

Messe die Entfernung zwischen der Startlinie und der Ziellinie. Versuche, Dein Programm so zu ändern, dass der Edison-Roboter kurz vor der Ziellinie ins Ziel fährt.

Sie sind dran:

1. Auf welche Konstante hast du "Ed.DistanceUnits" im Setup-Code gesetzt?

   _____

2. Welche Zahl musstest du als Eingabeparameter für die Entfernung eingeben, damit Edison vom Start bis zur Ziellinie fährt?

   _____

3. Versuche, den Roboter Edison mit verschiedenen Geschwindigkeiten zu fahren. Wie verhält sich der Roboter, wenn du Edison mit Geschwindigkeit 10 fährst? Bemerkst du irgendwelche Veränderungen in der Genauigkeit von Edison, wenn er mit Geschwindigkeit 10 fährt?

_____

_____

_____

_____

Name_____

Lektion 2: Arbeitsblatt 2.2 - Den Roboter rückwärts fahren

Bei dieser Aufgabe musst du ein Programm schreiben, um Deinen Edison-Roboter rückwärts zu fahren.

Sie sind dran:

Aufgabe 1: Rückwärts fahren
Schreibe das folgende Programm:

```
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
13 Ed.Drive(Ed.BACKWARD,Ed.SPEED_5,8)
14
```

Markiere mit Hilfe des Aktivitätsblatts 2.1 oder farbigem Klebeband eine Start- und eine Ziellinie auf einem Tisch oder auf dem Boden als Testbereich für Dein Programm. Experimentiere, um zu sehen, ob du Deinen Edison dazu bringen kannst, von der Start- zur Ziellinie rückwärts zu fahren.

Aufgabe 2: Verwenden Sie die Konstante 'Ed.DISTANCE_UNLIMITED'.

Es gibt mehrere Möglichkeiten, den Edison so zu programmieren, dass er vorwärts und rückwärts fährt. Eine andere Möglichkeit ist die Verwendung von 'Ed.DISTANCE_UNLIMITED' für den Abstandsparameter. Diese Konstante schaltet beide Antriebsmotoren des Edison ein.

Anders als bei der Verwendung eines Zahlenwerts für den Entfernungsparameter wird bei Ed.DISTANCE_UNLIMITED kein genauer Wert angegeben, nach dem die Motoren anhalten. Stattdessen schaltet er die Motoren ein und fährt dann mit der nächsten Codezeile fort. Ein Anhalten der Motoren wird später im Code benötigt.

Name_____

Die Verwendung von Ed.DISTANCE_UNLIMITED kann nützlich sein, wenn du ein Programm schreiben willst, bei dem die Motoren nur dann anhalten, wenn ein anderes Ereignis eintritt, z. B. wenn ein Hindernis erkannt wird.

Sehen Sie sich 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
13 Ed.Drive(Ed.BACKWARD, Ed.Ed.SPEED_6, Ed.DISTANCE_UNLIMITED)
14 Ed.TimeWait(200, Ed.TIME_MILLISECONDS)
15 Ed.Drive(Ed.STOP, Ed.SPEED_10, 0)
16
```

Dieses Programm schaltet die Edison-Motoren ein, um rückwärts zu fahren, und wartet dann 200 Millisekunden, bevor es die Motoren ausschaltet.
Schreibe ein Programm, das den Parameter Ed.DISTANCE_UNLIMITED verwendet, um rückwärts zu fahren.
Markiere mit Hilfe des Aktivitätblatts 2.1 oder farbigem Klebeband eine Start- und eine Ziellinie auf einem Tisch oder auf dem Boden als Testbereich für Dein Programm. Experimentiere, um zu sehen, ob du Deinen Edison dazu bringen kannst, von der Start- zur Ziellinie rückwärts zu fahren.

1. Setze die Geschwindigkeit in Deinem Programm auf Ed.SPEED_6. Wie viele Millisekunden musst du in die Funktion TimeWait() eingeben, damit Edison von der Ziellinie rückwärts zur Startlinie fährt?

   _____

2. Experimentiere mit verschiedenen Geschwindigkeits- und TimeWait()-Eingabeparametern. Was sind die schnellsten und die langsamsten Zeiten, die du Edison rückwärts vom Ziel zur Startlinie fahren lassen kannst?

   Schnellste: _____ Langsamste: _____

Name_____

Lektion 2: Arbeitsblatt 2.3 - Ausdrücke in Python

In dieser Aktivität lernst du ein wichtiges Element des Codes kennen, das wir bei der Programmierung in Python verwenden: Ausdrücke.

Was sind Ausdrücke?

Ein Ausdruck ist eine Frage, die entweder als "wahr" oder "falsch" beantwortet werden kann. Zum Beispiel: "Ist A kleiner als B?" oder "Ist A gleich B?".

Im Code werden Ausdrücke mit mathematischen Bezeichnungen anstelle von Wörtern geschrieben. Im Setup-Code haben Sie gesehen, dass die =-Notation verwendet wird. Zum Beispiel Ed.DistanceUnits = Ed.CM. Die Notation "A = B" bedeutet "setze A auf den gleichen Wert wie B".

Ausdrücke verwenden auch Notationen. Dies sind einige der grundlegenden Notationen in Ausdrücken, die wir in Python verwenden können:

Ausdruck	Bedeutung
A == B	Ist A dasselbe wie B?
A != B	Ist A nicht gleich B?
A > B	Ist A größer als B?
A >= B	Ist A größer als oder gleich B?
A < B	Ist A kleiner als B?
A <= B	Ist A kleiner als oder gleich B?

Ausdrücke vergleichen die linke Seite mit der rechten Seite der Notation im Ausdruck.

Du kannst "A" und "B" in der obigen Liste von Ausdrücken durch einen beliebigen Wert oder eine Funktion ersetzen, die einen Wert zurückgibt. du kannst auch mit diesen Werten rechnen. Zum Beispiel bedeutet "(A + 2) > B": "Ist A plus 2 größer als B?

Im Code funktionieren Ausdrücke in einer bestimmten Reihenfolge. Wenn Dein Ausdruck mathematische Werte enthält oder eine Funktion aufruft, löst der Ausdruck zuerst die mathematischen Werte oder die Funktion auf. Dann wird die linke Seite des Ausdrucks mit der rechten Seite verglichen und entweder als "wahr" oder "falsch" aufgelöst.

Wofür werden Ausdrücke in Python verwendet?

Ausdrücke werden zusammen mit anderen Codeelementen wie "while"-Schleifen und "if"-Anweisungen verwendet, um den Codefluss zu verändern. Diese Elemente ermöglichen es, dass der Code anders abläuft als nur im sequentiellen Fluss von Zeile 1 zu Zeile 2 zu Zeile 3.

Name_____

Sie sind dran:
Übe das Auflösen von Ausdrücken. Schreibe zunächst auf, was jeder Ausdruck bedeutet, und lösen Sie ihn dann in wahr oder falsch auf.

Wenn A = 2 und B = 4 ist, was bedeutet dann jeder der folgenden Ausdrücke und was ergibt sich daraus (wahr oder falsch)?

1. (A*2) == B

   Bedeutung: _____

   Wird ausgewertet zu: _____

2. A >= B

   Bedeutung: _____

   Wird ausgewertet zu: _____

3. (A+A) != B

   Bedeutung: _____

   Wird ausgewertet zu: _____

4. (A-1) < (B-3)

   Bedeutung: _____

   Wird ausgewertet zu: _____

Name_____

Lektion 2: Arbeitsblatt 2.5 - Tastaturgesteuertes Fahren

In dieser Aufgabe musst du ein Programm schreiben, das deinen Edison-Roboter nur dann vorwärts fährt, wenn entweder die runde oder die dreieckige Taste gedrückt wird. Dazu werden wir Ausdrücke und die "while"-Schleife verwenden.

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 Ed.ReadKeypad()
13 while Ed.ReadKeypad() == Ed.KEYPAD_NONE:
14     pass
15 Ed.Drive(Ed.FORWARD, Ed.SPEED_6,8)
16
```

Dieses Programm verwendet eine 'while'-Schleife mit einem Ausdruck.

In Python wiederholt eine 'while'-Schleife eine Anweisung oder eine Gruppe von Anweisungen, solange eine bestimmte Bedingung WAHR ist. Sie testet die Bedingung, die als Ausdruck geschrieben ist, bevor sie den Schleifenkörper ausführt.
Solange der Ausdruck TRUE ergibt, wiederholt das Programm die Befehle in der Schleife. Wenn der Ausdruck zu FALSE ausgewertet wird, geht das Programm zur nächsten Codezeile außerhalb der Schleife weiter.

Einrückung verwenden

Python verwendet Einrückungen, um Anweisungen oder Befehle zusammenzufassen.

In Python werden alle Anweisungen, die um die gleiche Anzahl von Leerzeichen eingerückt sind, als ein einziger Codeblock betrachtet.

Sieh dir Zeile 14 des Programms an. Da "pass" eingerückt ist, befindet sie sich innerhalb der Schleife. Zeile 15 im Programm ist jedoch nicht eingerückt, also befindet sich Zeile 15 nicht innerhalb der Schleife.

Funktionen und Konstanten in diesem Programm

Ed.ReadKeyPad() - diese Funktion liest den Zustand der Edison-Tastatur. Mit anderen Worten, sie ermittelt, ob eine der Tasten von Edison gedrückt wurde oder nicht.
Ed.ReadKeyPad() gibt einen Wert zurück, der angibt, welche Taste gedrückt wurde:

Ed.KEYPAD_NONE, Ed.KEYPAD_TRIANGLE oder Ed.KEYPAD_ROUND.

Name_____

Diese Funktion funktioniert nicht für die quadratische Taste. Das liegt daran, dass die quadratische Schaltfläche nur zum Anhalten eines Programms gedacht ist. Die quadratische Schaltfläche hält immer ein laufendes Programm an, wenn sie gedrückt wird.

Besonderer Hinweis: Verwendung von 'read'-Funktionen innerhalb einer Schleife
Einige Arten von Daten werden vorübergehend im Speicher von Edison gespeichert. So kann die Funktion Ed.ReadKeyPad() einen Tastendruck auslesen, bevor die Lesefunktion in Deinem Code aufgerufen wird.

In diesem Programm wollen wir sicherstellen, dass Ed.ReadKeyPad() in der while-Schleife wartet, bis eine Taste gedrückt wurde, und keine Tastendrücke berücksichtigt, die vor der while-Schleife erfolgen. Deshalb fügen wir Ed.ReadKeyPad() in die Zeile oberhalb der while-Schleife ein (Zeile 12). Dadurch werden alle vorherigen Tastendrücke vor der Schleife gelöscht.

Dieses Verfahren solltest du immer befolgen, wenn du eine Lesefunktion innerhalb einer Schleife verwendest.

Du bist dran:
Schreibe das Programm.

Achten Sie beim Schreiben der while-Schleife auf den Doppelpunkt und die richtige Einrückung

```
13  while Ed.ReadKeypad() == Ed.KEYPAD_NONE:
14      pass
```

Lade das Programm herunter und drücke die Dreieckstaste, um das Programm zu starten. Warten Sie ein wenig und versuchen Sie dann, entweder die Dreieckstaste oder die runde Taste zu drücken.

1. Was hat der Roboter gemacht, als du einige Sekunden nach dem Start des Programms die Dreieck- oder Rundtaste gedrückt hast?

_____

_____

Name_____

2. Starte das Programm erneut und versuche, die quadratische Taste anstelle der runden oder dreieckigen Taste zu drücken. Was ist passiert? Warum ist das passiert? Erkläre.

3. Versuche nun, am Ende des Programms weiteren Code hinzuzufügen. Der neue Code, den du schreibst, sollte Edison dazu bringen, rückwärts zu fahren, nachdem du entweder die runde oder die dreieckige Taste erneut gedrückt haben. Dein Programm sollte Edison jetzt also anweisen, beim ersten Drücken einer Taste vorwärts zu fahren, und dann rückwärts, wenn eine Taste erneut gedrückt wird. Vergiss nicht, den Doppelpunkt einzufügen und den Code innerhalb der while-Schleife einzurücken. Wie sieht Dein neues Programm aus? Schreibe Deinen Ihren Code unten auf.

Name_____

Lektion 2: Aktivitätsblatt 2.1

ZIEL

START

www.edpyapp.com www.meetedison.com