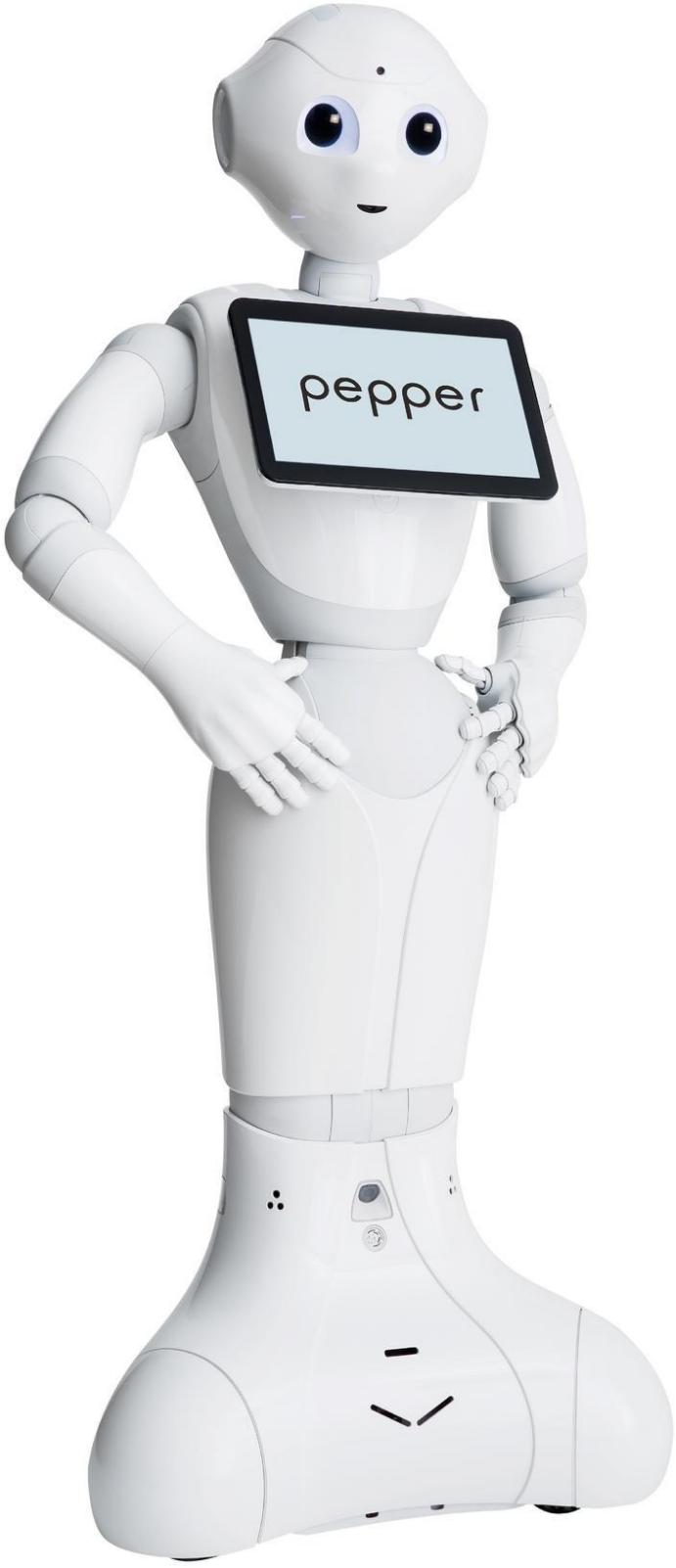
Übersicht über die Pepper RoboBlocks Software und Einleitung



Vorwort

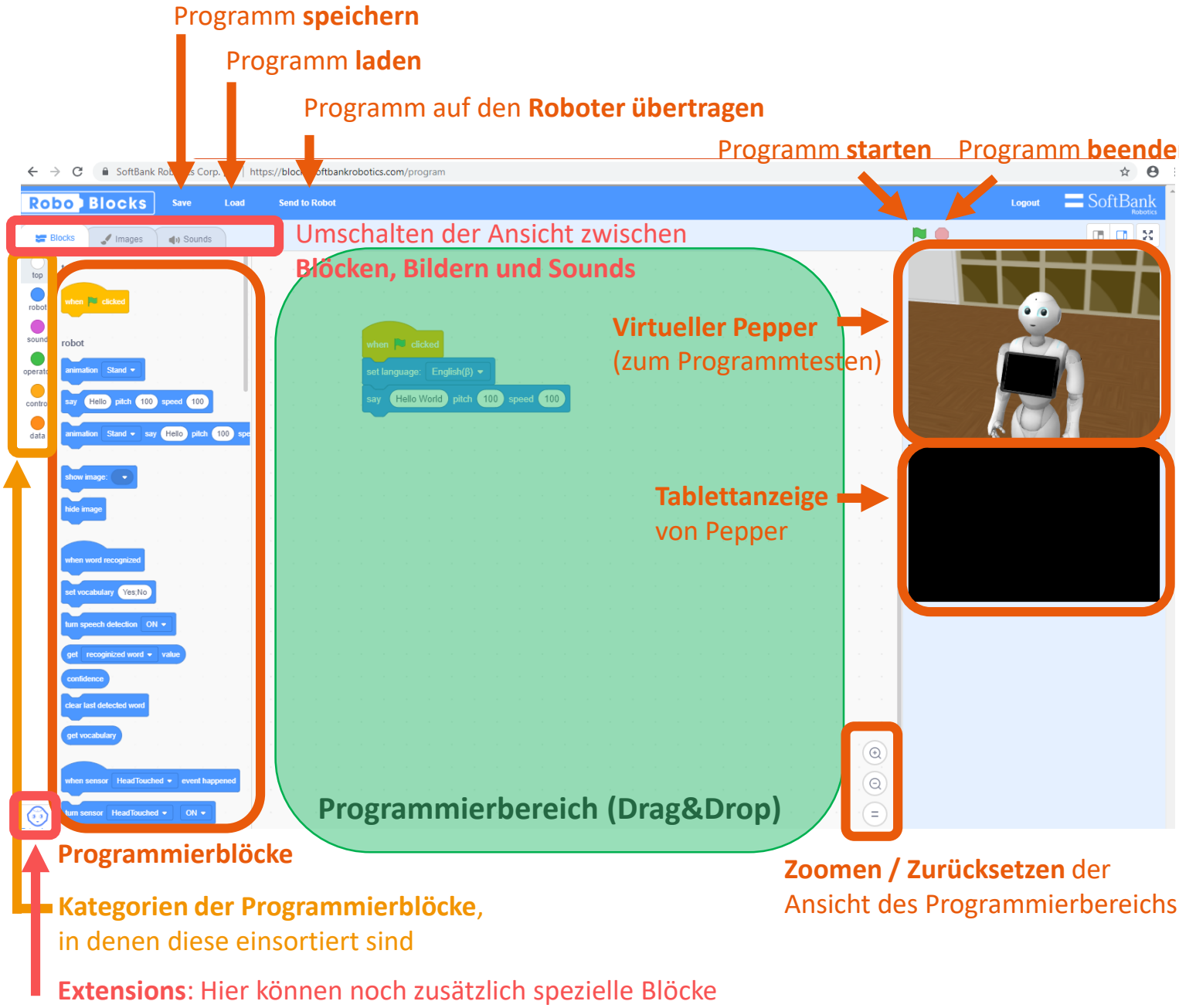
Dies ist eine Einführung in die RoboBlocks Software von SoftBank Robotics (https://www.softbankrobotics.com/), für den Workshop „Mein smarter Roboter“. Dieser wurde im Rahmen des smile Projektes entwickelt und durchgeführt. Hier werden einige Beispielaufgaben beschrieben, wie sie im Rahmen des Workshops mit den Teilnehmerinnen durchführbar sind. Sie vermitteln ein Grundverständnis von Informatik und erste Programmiergrundkonzepte.

Da die hier verwendete Software extern entwickelt wird, können aufgrund von zukünftigen Updates Veränderungen oder Unstimmigkeiten in den Blöcken oder der Benutzeroberfläche auftreten. Wir geben uns Mühe, dies so gut es geht aktuell zu halten. Die grundlegenden Funktionen werden aber erhalten bleiben. Es empfiehlt sich dies nochmal vor der Durchführung des Workshops abzugleichen und sich Änderungen ggf. zu notieren.

Diese Anleitung ist auch an die Teilnehmerinnen selbst adressiert und kann zum eigenständigen Durcharbeiten oder als Referenz verteilt werden.

Änderungsvorschläge oder Aufgabenideen können gerne in GitHub https://github.com/projekt-smile/Mein-smarter-Roboter als "Issues" hinzugefügt werden.

Übersicht der Benutzeroberfläche



Dies ist eine grobe Übersicht über die Benutzeroberfläche und deren einzelne Bereiche und Ansichten.
Die Kategorien der Programmierblöcke erleichtern das finden spezifischer Blöcke. Man kann aber auch in den Blöcken herunter scrollen, bis man bei dem gesuchten Block ankommt.
Extensions: Hier findet man Blöcke, die spezifische Unterrichtsthemen behandeln und eher speziellere Anwendungsgebiete haben.

Beispielprogramme

Im Folgenden werden Beispiele und Übungen dargestellt. Meistens gibt es eine Aufgabe, die ein neues Konzept einführt. Danach gibt es weiterführende Aufgaben zum gleichen Konzept, die entweder deren Stärken, Besonderheiten oder Grenzen aufzeigen. Die Programme sind jeweils als Json Datei hinterlegt oder können hier anhand der Bilder nachgebaut werden. Es werden außerdem Anregungen zu Experimenten mit den entsprechenden Blöcken aufgezeigt oder häufige Fehler aufgelistet.

Die Teilnehmerinnen sollen dazu ermutigt werden, selbst zu experimentieren und herauszufinden, was die einzelnen Blöcke tun.

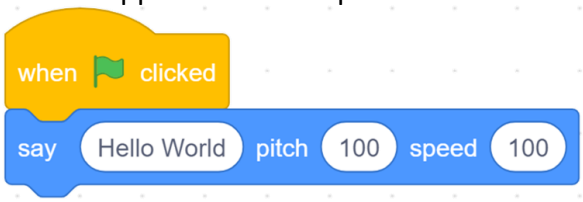
Es wird außerdem den Teilnehmerinnen ein Handout gegeben, in dem die wichtigsten Blöcke nochmal übersetzt und kurz erklärt sind. Dies ist insbesondere wichtig für mehrtägige Workshops.

Der Spaß am Lernen soll aber stets im Vordergrund stehen.

1. Hello World

Jeder Programmierer in jeder Programmiersprache beginnt mit einem sogenannten „Hello World“ Programm. Das heißt, dass man dem Programm und in unserem Fall dem Roboter sprechen beibringt. Klassischerweise kann ein Computer nicht sprechen, sondern zeigt uns den Text nur auf seinem Bildschirm. Pepper aber kann sprechen und so soll er uns auch Hello World sagen.

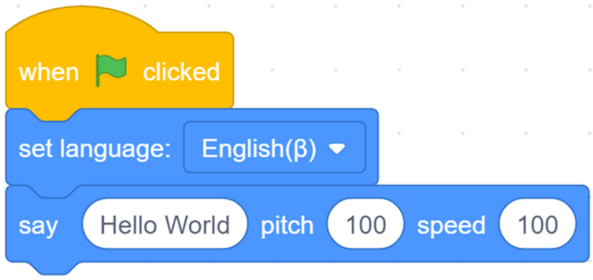
1a)



Der erste Versuch.
Dafür brauchen wir den **„When clicked“** Block. Damit weiß das Programm, dass es starten soll, wenn wir die **grüne Fahne** drücken. (Der Block startet auch, wenn man per Doppelklick drauf klickt.)

Dann muss der Roboter ja noch etwas sagen. Dafür brauchen wir den **„Say“** (Sprechen) Block. Den können wir, wie bei einem Puzzle, darunter ansetzen. Beide Blöcke sollten dann am Ende wie oben gezeigt, verbunden sein.
Jetzt können wir das ganze mit einem Klick auf die **grüne Fahne** oder einem Doppelklick auf die Blöcke ausprobieren. Was passiert? Der Roboter sagt etwas. Aber es klingt komisch. Das liegt daran, dass der Roboter versucht, englische Wörter Japanisch auszusprechen. Wir möchten aber, dass er mit uns auf Englisch spricht. Deshalb brauchen wir einen weiteren Block:

1b)



Wir haben einen zweiten Block hinzugefügt, den ihr durch weiteres Herunterscrollen finden könnt. Der Block heißt **„set language“** und lässt uns eine Sprache zwischen Japanisch und Englisch auswählen. Wir wählen Englisch aus und testen das ganze, wie auch bei der vorherigen Aufgabe nochmal.
Jetzt können wir verstehen, was Pepper sagt.

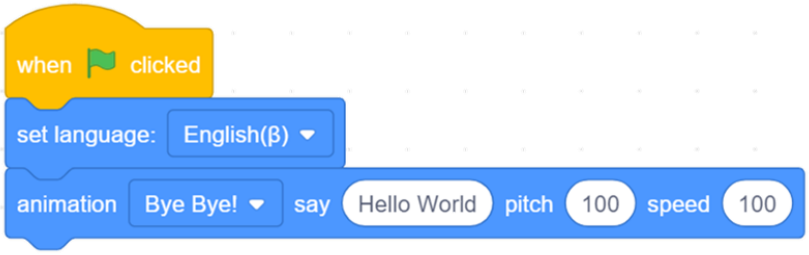
BONUSAUFGABE: Wir haben bei dem vorherigen **„say“** Block bisher zwei Sachen ignoriert: und zwar die beiden Felder **„pitch“** und **„speed“.** Was passiert, wenn ihr die Werte in diesen Blöcken verändert? Probiert es aus!

LÖSUNG: „pitch“ gibt die Tonhöhe der Stimme an und **„speed“** die Geschwindigkeit, mit der der Text ausgesprochen werden soll.

1.c) Hello World – mit Bewegung

Wir möchten, dass der Roboter nicht nur spricht, sondern sich auch dabei bewegt. Am besten winkt er uns zu. Hierfür ersetzen wir den **„say“** Block mit einem **„animation say“.**

1c)



Dieser Block sieht fast genauso aus wie der **„say“** Block aus der vorherigen Aufgabe, aber er hat noch eine kleine Besonderheit: man kann sich eine **Animation** aussuchen. Wählt die Animation „Bye Bye!“ aus, wenn ihr möchtet, dass der Roboter euch zuwinkt.
Wenn ihr nun auf die **grüne Fahne** klickt, dann winkt euch der animierte Pepper Roboter (oben rechts) zu.

BONUSAUFGABE: Probiert weitere Bewegungen aus. Was macht der Roboter? Lasst ihn auch ruhig verschiedene Sachen sagen. Vielleicht kann er euch ja mit Namen begrüßen? Vielleicht könnt ihr ihm aber auch beibringen mehrere Sachen zu sagen und sich dabei unterschiedlich zu bewegen? Probiert es aus!

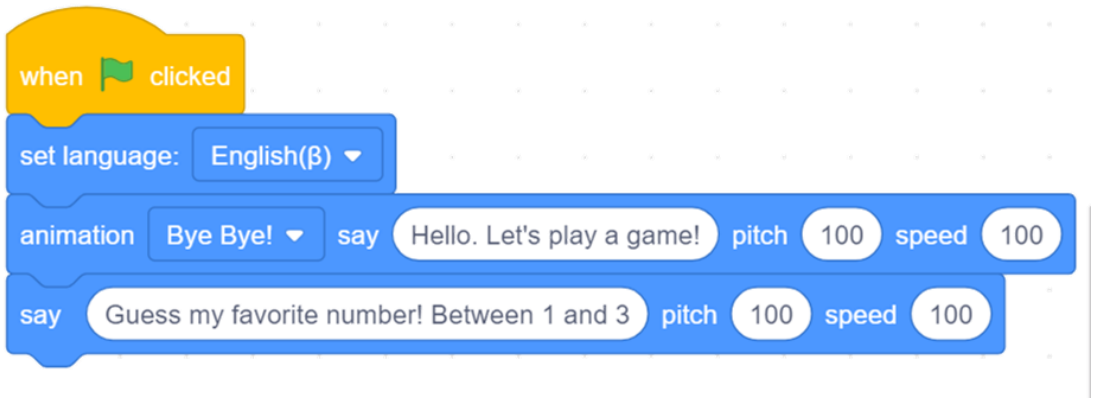
2. Zahlen raten

Wir möchten mit Pepper auch interagieren und haben uns überlegt, dass wir ein Ratespiel spielen wollen. Da Pepper ein Roboter ist und Roboter sehr gerne Zahlen mögen, werden wir versuchen, die Lieblingszahl von Pepper zu erraten.

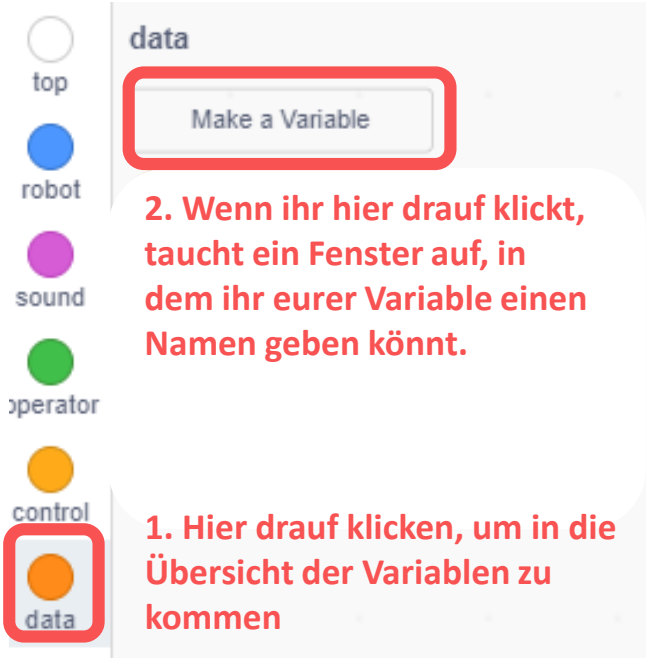
Bevor wir das machen können, sollte Pepper uns aber sagen, dass er ein Ratespiel mit uns machen möchte.

AUFGABE 2a: Wie kann Pepper uns **sagen**, dass er ein Spiel mit uns spielen möchte?

LÖSUNG:



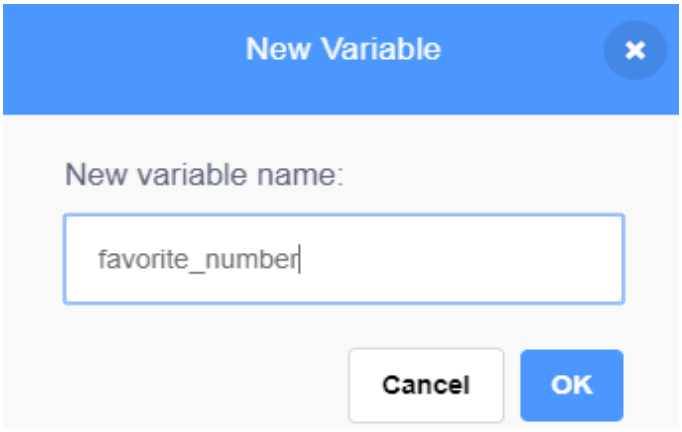
Jetzt muss sich der Roboter eine Zahl ausdenken und merken. Damit sich Pepper etwas merken kann, brauchen wir eine sogenannte **Variable**. In einer **Variable** kann sich Pepper eine Zahl oder auch ein Wort merken. Das ist wie ein Gedächtnis für den Roboter. Um eine Variable zu erstellen, müsst ihr oben links in der Leiste der Blockkategorien auf **data** klicken und dann auf make Variable. Dann könnt ihr euch einen Namen für die Variable aussuchen. Ich habe meine **„favorite number“** genannt.



2. Wenn ihr hier drauf klickt, taucht ein Fenster auf, in dem ihr eurer Variable einen Namen geben könnt.

1. Hier drauf klicken, um in die Übersicht der Variablen zu kommen

3. Gebt in dem Fenster einen Namen für eure Variable ein und klickt auf OK.



4. Dann tauchen für diese Variable folgende Blöcke auf:

Schauen, was gerade in der Variable gespeichert ist.

Etwas neues in die Variable einspeichern.

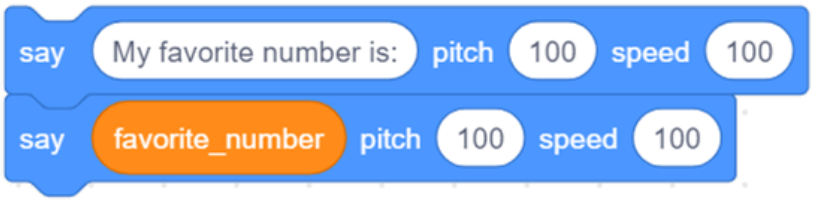
Den Wert der Variable um 1 verändern.

2. Zahlen raten - Variablen

Jetzt kann sich Pepper eine Zahl merken, wenn ihr diesen Block verwendet und eine Zahl reinschreibt. Für dieses Beispiel nehmen wir eine 2.

Jetzt lassen wir Pepper seine Lieblingszahl aussprechen. Wie könnte das gehen? Es gibt zwei Möglichkeiten. Könnt ihr euch da eine vorstellen?

Möglichkeit 1:



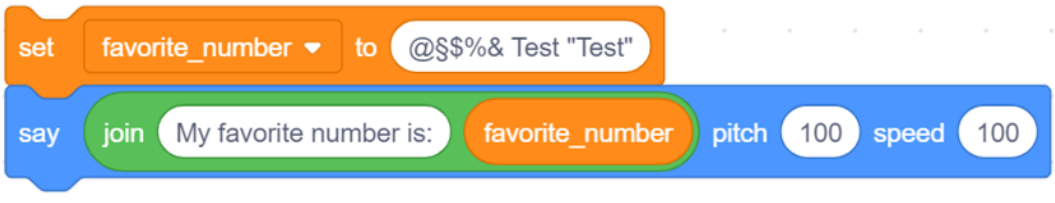
In dieser Möglichkeit verwenden wir zwei „say“ Blöcke. In den ersten geben wir einen Text ein, z.B. „My favorite number is“ und in den zweiten dann die Zahl. Man könnte die Zahl direkt eingeben, aber das wäre ja langweilig. Deshalb schauen wir, ob sich Pepper an die Zahl **erinnern** kann. Dafür nutzen wir den runden Block mit dem Namen unserer Variable.
BONUSAUFGABE: Was fällt euch dabei auf, wenn wir das ausführen?
LÖSUNG: Der Roboter spricht abgehakt. Kann man das irgendwie verhindern oder lösen? Ja klar!

Möglichkeit 2:

In dieser Möglichkeit verwenden wir einen neuen runden Block, nämlich **„join“.** Diesen Block findet ihr in der Blockübersicht unter dem Begriff **„operator“.** Dieser Block vereinigt den Text, den ihr eingegeben habt, mit der Zahl, die sich Pepper gemerkt hat. Weil diese beiden Teile zusammengefasst werden, kann Pepper sie flüssiger aussprechen.

TIPP: Blöcke, die ihr gerade nicht braucht, die ihr aber für später behalten möchtet, könnt ihr einfach beiseite schieben. Wenn ihr das Programm startet, werden sie einfach ignoriert. So könnt ihr bestimmte kleinere Teile eures Programms separat testen.

AUFGABE 2b: Definiert noch weitere Variablen. Zum Beispiel eine Zahl die Pepper gar nicht mag? Oder vielleicht muss es ja auch gar keine Zahl sein, die sich Pepper merken soll? Experimentiert ein wenig herum!
Beispiel:

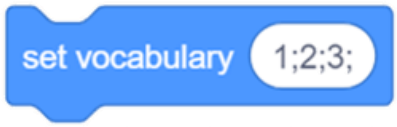


Pepper kann das meiste was man ihm vorsetzt tatsächlich in irgendeiner Art und Weise aussprechen. Text kann auch in einer Variable gespeichert werden.

Aber setzten wir mal wieder Peppers Lieblingszahl auf 2. Jetzt wäre es doch schön, wenn wir die Zahl raten könnten und Pepper uns dafür sagen kann, ob die Zahl richtig ist oder nicht. Dafür muss Pepper uns erstmal zuhören.

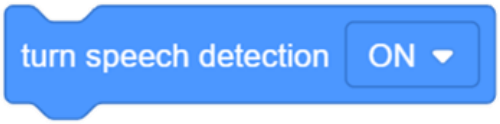
2. Zahlen raten – Zuhören (1)

Bevor uns Pepper zuhören kann, müssen wir ihm mitteilen auf welche Wörter er hören soll. Es gibt in unserer Sprache sehr sehr viele Wörter. Wenn ein Roboter auf alle hören soll, ist es etwas viel. Außerdem wollen wir ja ganz klein anfangen ☺ Deshalb teilen wir Pepper mit, dass er erstmal nur auf die Zahlen 1,2,3 hören soll. Dafür brauchen wir den **„set vocabulary“** (Setze die Vokabeln) Block.

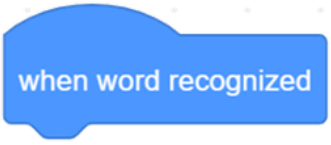


In diesem Block tragen wir erstmal „1;2;3“ ein, die jeweils durch ein **Semikolon (;)** getrennt werden. Es ist sehr wichtig, dass die Wörter oder Zahlen durch ein Semikolon getrennt werden und nicht durch ein Leerzeichen! Auch keine Leerzeichen zwischen Semikolon und dem Wort!

Jetzt darf Pepper uns endlich zuhören. Hierfür gibt es zwei wichtige Blöcke:

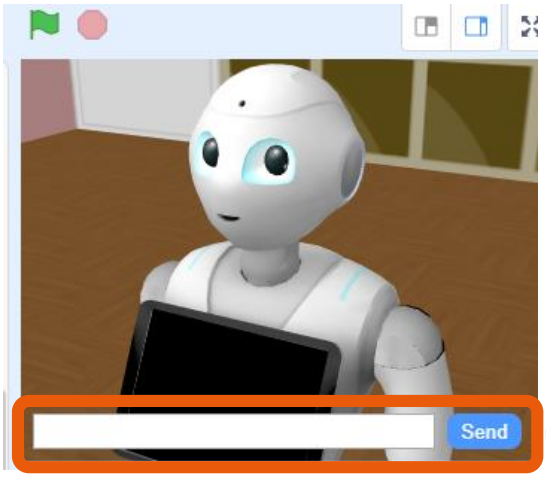


Dieser Block **„turn speech detection on/off“** sagt Pepper, dass er **zuhören** soll oder auch nicht. Jetzt müssen wir aber auch sagen können, wie Pepper reagieren soll, wenn er ein Wort richtig erkannt hat.



Alles was unter diesen Block kommt, wird nur ausgeführt, wenn Pepper ein **Wort erkannt** hat. Es ist erstmal egal welches Wort, sondern es geht darum, dass er überhaupt eins erkennt. Wie ihr sehen könnt, passt dieser Block mit seiner Rundung auch nicht unter die anderen Blöcke sondern sieht aus, wie der **Start** Block. So funktioniert er auch. Man kann sich das so vorstellen, als würde ein neuer Programmabschnitt ausgeführt werden.

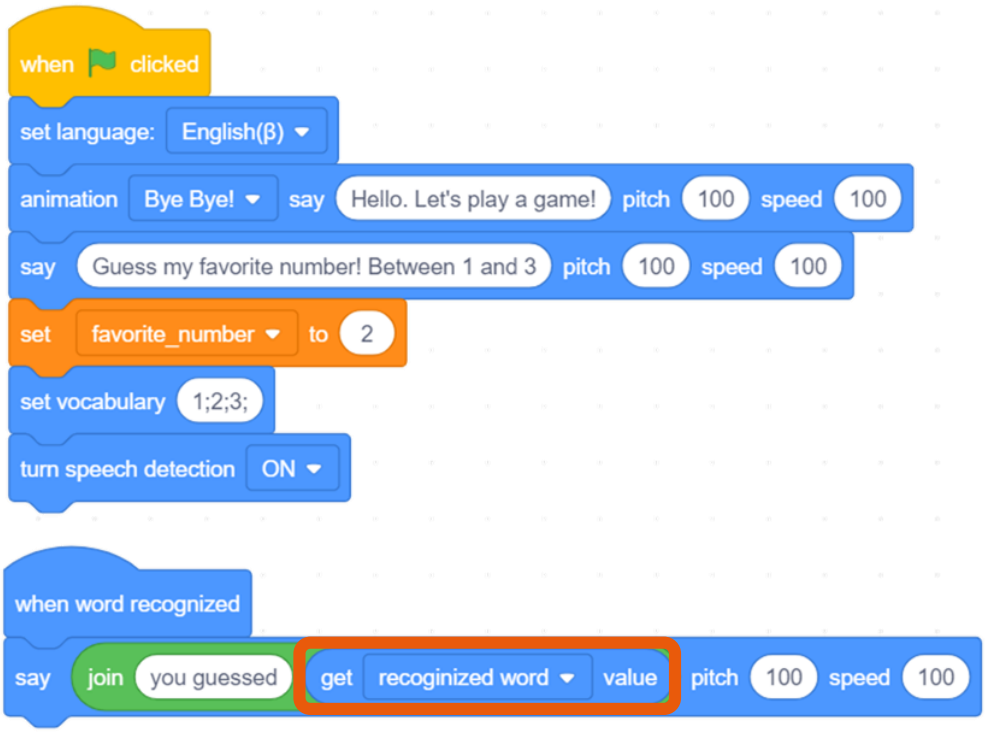
Ein **BEISPIEL:** Wir lassen Pepper das sagen, was wir geraten haben. Könntet ihr euch vorstellen, wie man das machen könnte?
(Was wir raten, können wir oben rechts eingeben, wo man auch den virtuellen Pepper sehen kann.)



Durch dieses Feld könnt ihr mit dem Roboter „sprechen“. Alles was ihr eingebt, „hört“ Pepper.

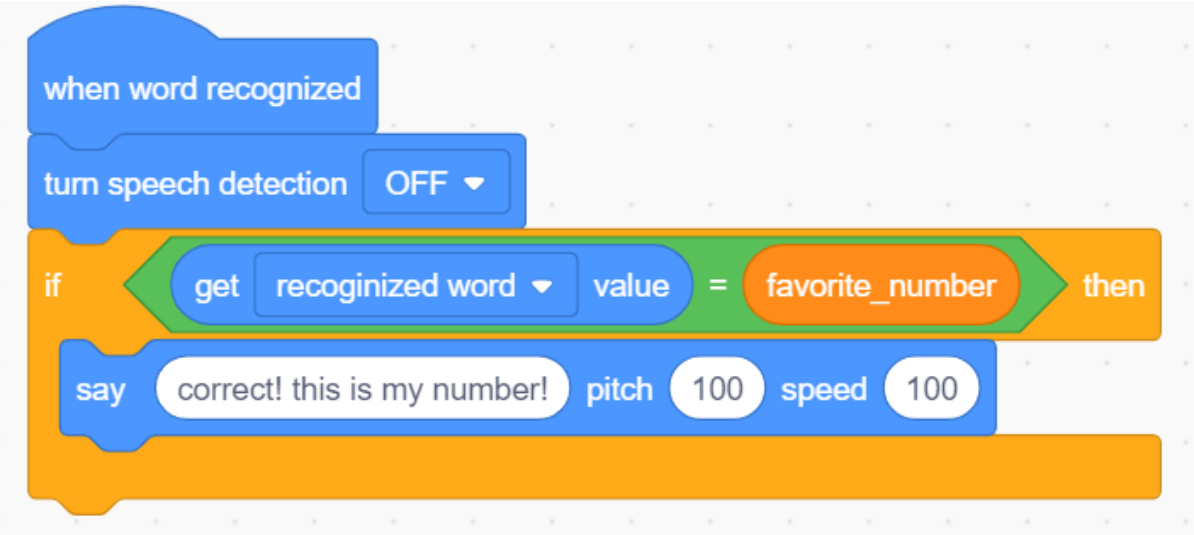
2. Zahlen raten – Zuhören (2)

Pepper kann uns sagen, was er gehört hat, wenn wir das Programm wie folgt aussehen lassen:



Der neue Block hier ist der **„get recognized word value“** Block. Wie ihr sicher erraten könnt, gibt dieser Block uns das erkannte Wort. Intern merkt sich Pepper das Wort genauso in einer **Variable**, wie wir das vorhin mit der Lieblingszahl selbst ihm beigebracht haben.

Jetzt möchten wir aber, dass wir Peppers Lieblingszahl erraten können. Dafür muss Pepper uns sagen können, ob die Zahl, die wir geraten haben richtig oder falsch ist. Das heißt, er muss das, was wir gesagt haben, mit dem **Vergleichen**, was er sich gemerkt hat. Außerdem muss er unterschiedlich reagieren. Wenn wir die Zahl erraten haben, soll er uns das sagen. Wenn wir falsch geraten haben, sollte er uns das auch sagen. Dafür gibt es eine eigene Kategorie von Blöcken, nämlich die „Control“ Blöcke.

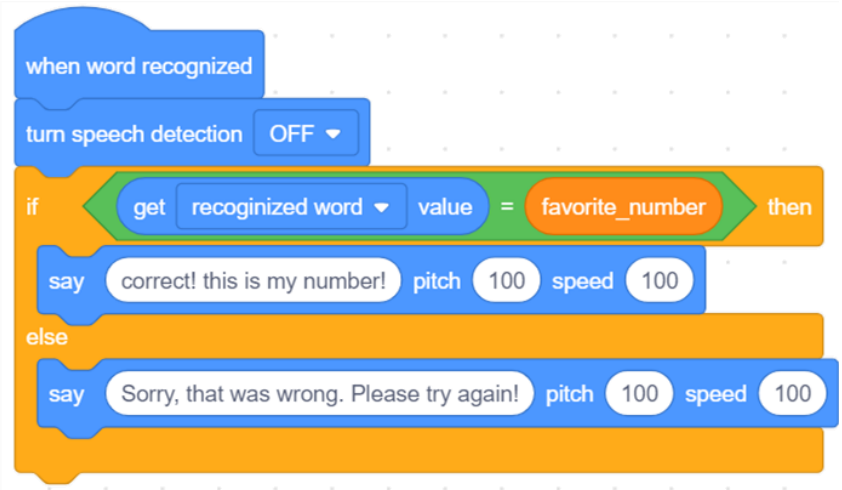


Der neue Block hier ist der **„if“** Block. „If“ heißt soviel wie „wenn“. Wenn eine **Bedingung** eintrifft, soll etwas passieren. In unserem Fall möchten wir das Wort, was Pepper gehört hat, mit dem vergleichen, was wir ihm vorhin in sein Gedächtnis eingespeichert haben. Seine Lieblingszahl. Diese beiden Werte befinden sich in Variablen. Einmal in der „get recognized word value“ Variable und einmal in der „favorite number“ Variable. Damit wir schauen können, ob beide gleich sind, brauchen wir den **„=„** Block, den wir uns aus der „Operator“ Kategorie holen können.

2. Zahlen raten – Zuhören (3)

Probiert aus, was passiert.

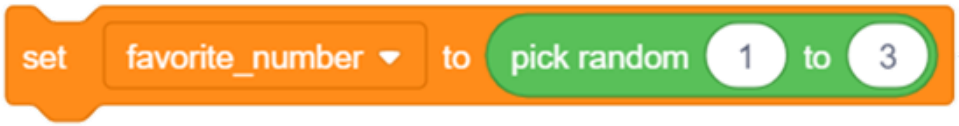
Der Roboter sollte kurz das Spiel vorstellen und dann darauf warten, dass man eine Zahl eingibt. Wenn man seine Lieblingszahl erraten hat, sagt Pepper uns Bescheid. Aber er sagt noch nichts, wenn die Zahl falsch ist. Das möchten wir als nächstes ändern. Dafür gibt es einen Block, der fast genauso aussieht, wie der **if** Block, aber noch eine kleine Erweiterung hat. Nämlich das **„else“:**



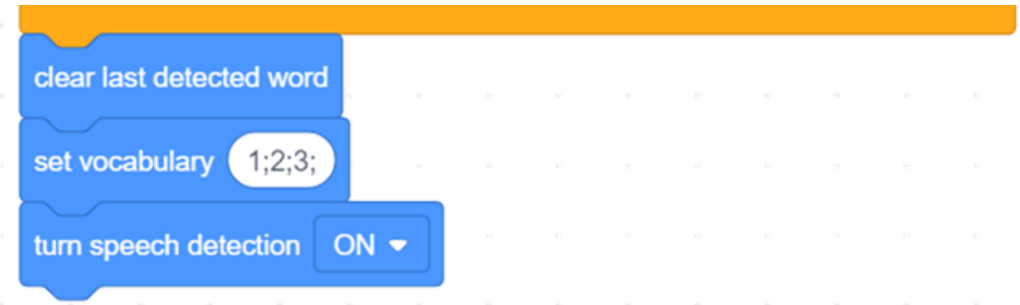
Der **if-else** Block verhält sich fast genauso wie der **if** Block, nur dass wir in diesem Fall Pepper Bescheid sagen lassen, wenn wir falsch geraten haben. Also, der Roboter kann auch reagieren, wenn die Bedingung nicht eintrifft. Man kann den Block lesen wie: „Wenn die Bedingung stimmt, mache dies, wenn sie nicht stimmt, mache das.“
Damit Pepper uns aber sofort wieder zuhört, müssen wir ihm das sagen. Also fügt den „turn speech detection on“ Block nach dem else des if-else wieder ein. (Das funktioniert nur auf dem echten Roboter…?)

AUFGABE: Erratet Peppers Lieblingszahl. Lasst ihn sich auch mal eine andere Zahl merken. Funktioniert das auch?

Jetzt wollen wir aber richtig raten. Pepper soll sich eine Zahl **zufällig** aussuchen. Dafür findet ihr in der grünen **„Operator“** Kategorie einen Block, der **„pick random 1 to 10“** heißt. Mit diesem Block könnt ihr Pepper sagen, dass er sich eine zufällige Zahl zwischen 1 und 10 aussuchen kann. Wir sollten das wieder auf 1 bis 3 beschränken. Dieser Block kann in den **„set favorite_number“** Block eingefügt werden:



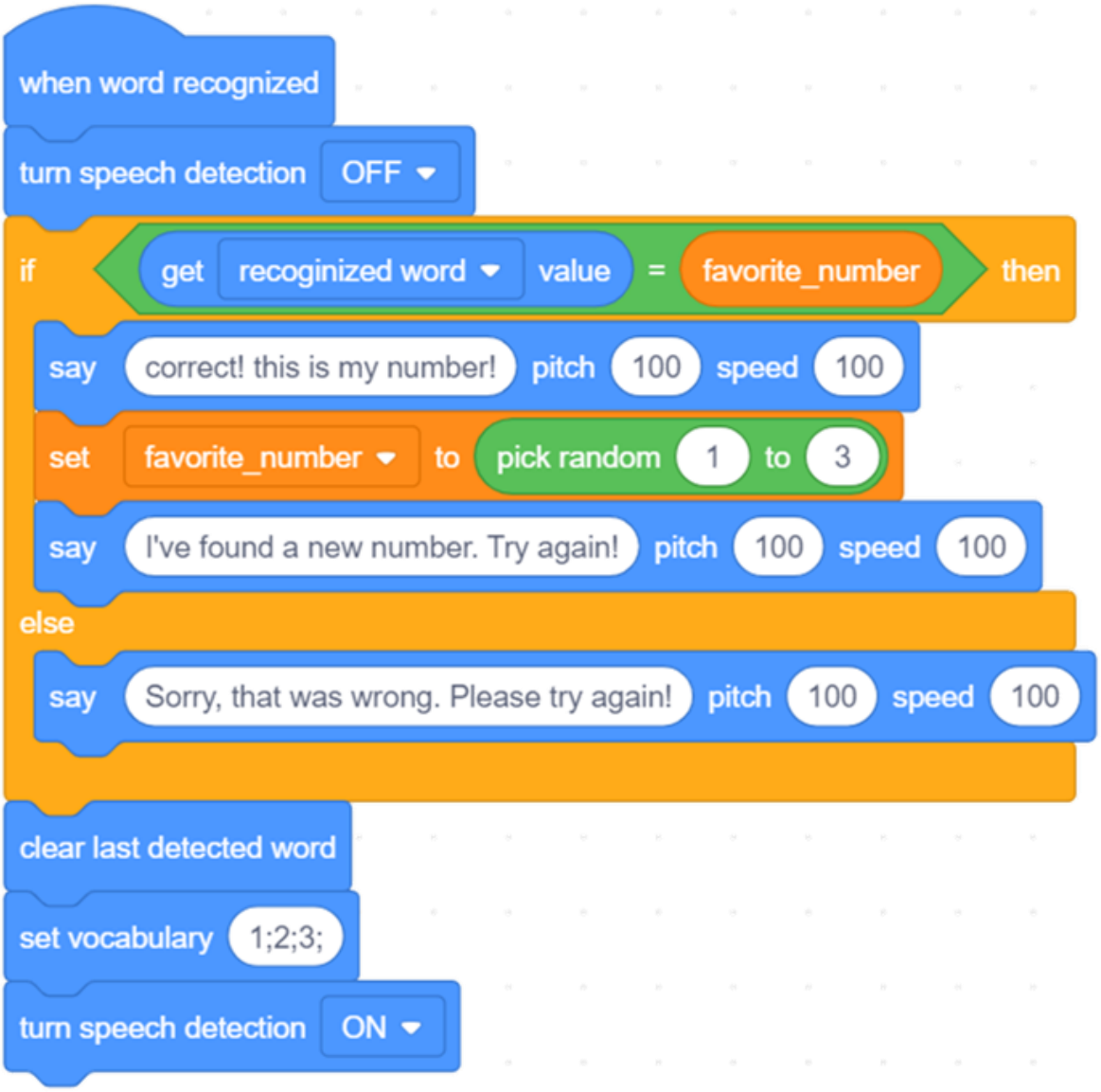
Außerdem möchten wir, dass wir nicht nur einmal raten dürfen, sondern mehrmals. Hierfür können wir Pepper einfach sagen, dass nachdem er ein Wort erkannt hat, dies wieder vergessen soll, noch einmal die Vokabeln durchgehen und danach uns wieder zuhören soll. Das sieht dann so aus:



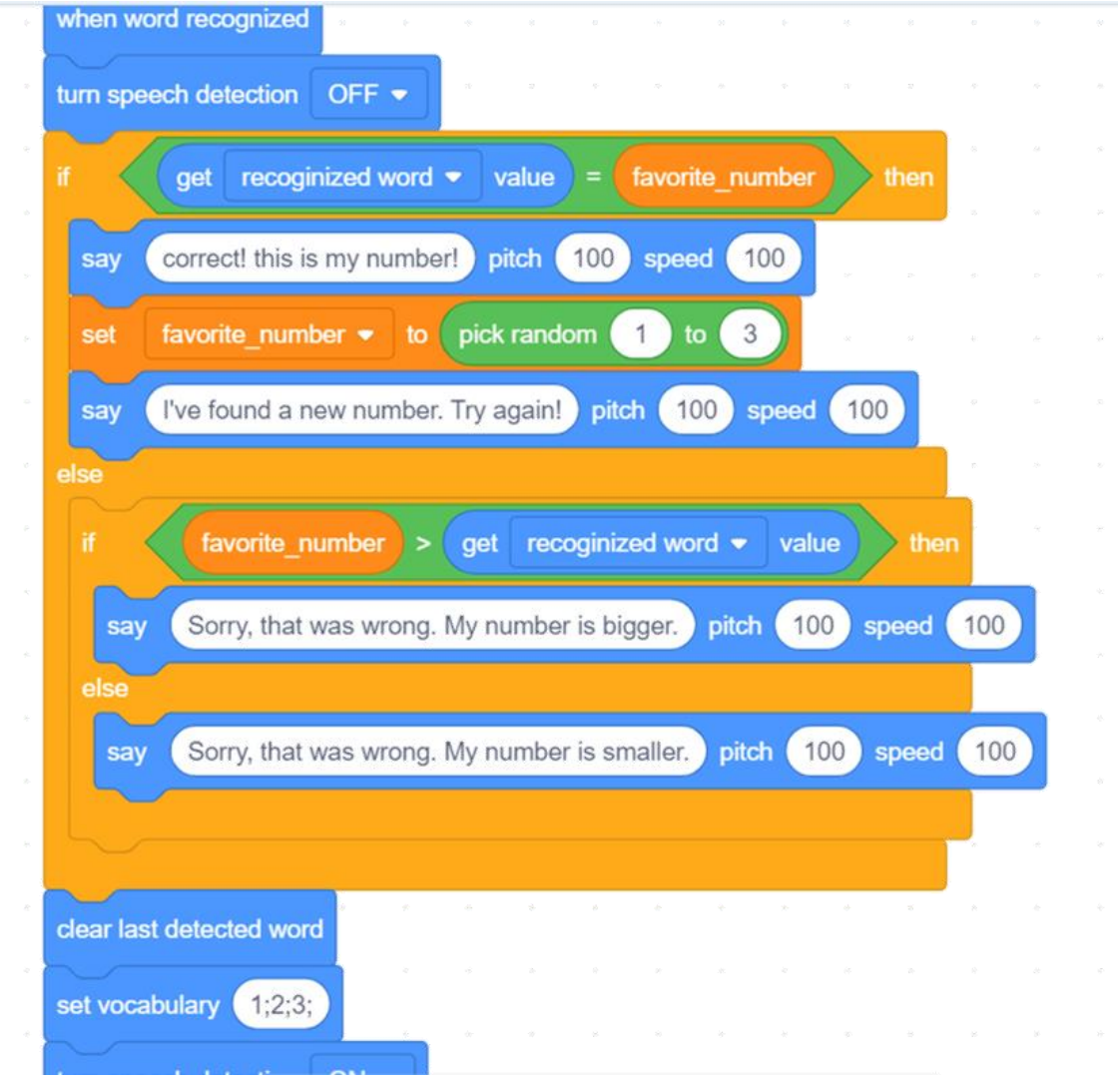
2. Zahlen raten – Zuhören (4)

Noch die letzten Feinschliffe für das Zahlenraten.

Pepper soll sich eine neue Zahl überlegen, nachdem wir die richtige erraten haben:



2. Zahlen raten – größer kleiner

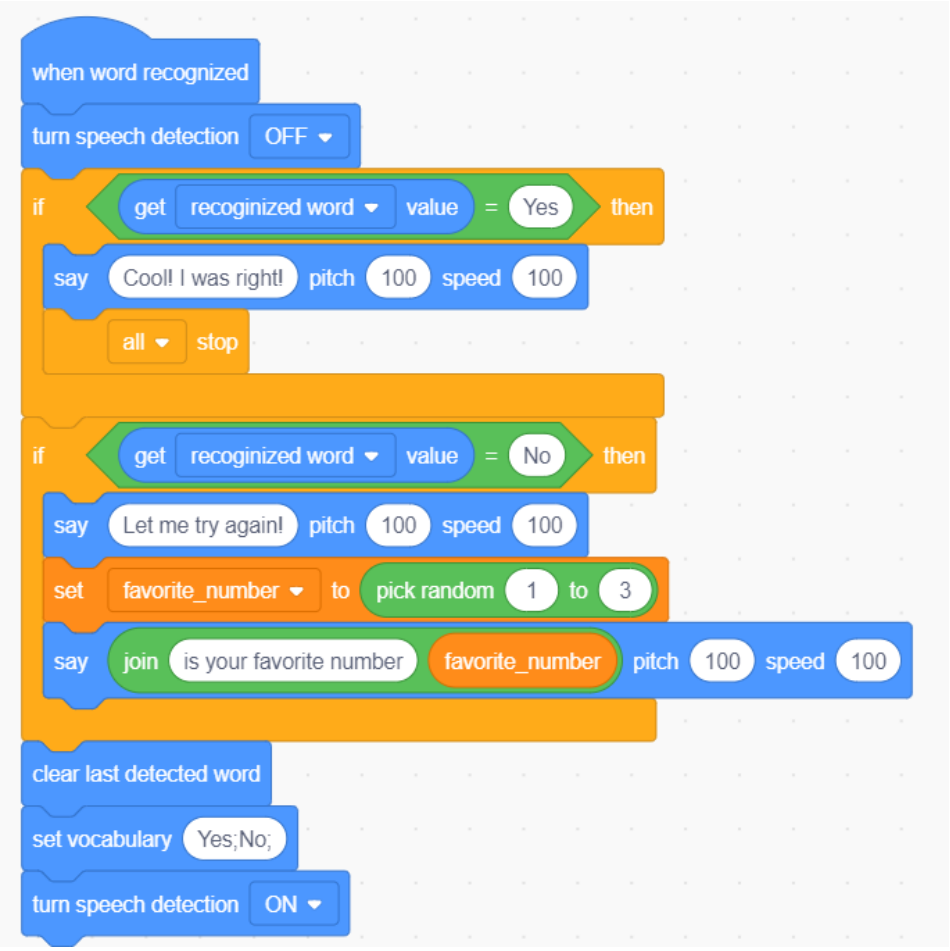


Zum Selberausprobieren. Große Aufgabe: Pepper soll Tiere erkennen. Also braucht Pepper ein Lieblingstier. Wisst ihr, wie wir das machen müssen?

Pepper soll sich unterschiedlich bewegen, je nachdem, ob die Zahl oder das Tier richtig erraten wurde. Baut noch Bewegungen ein.

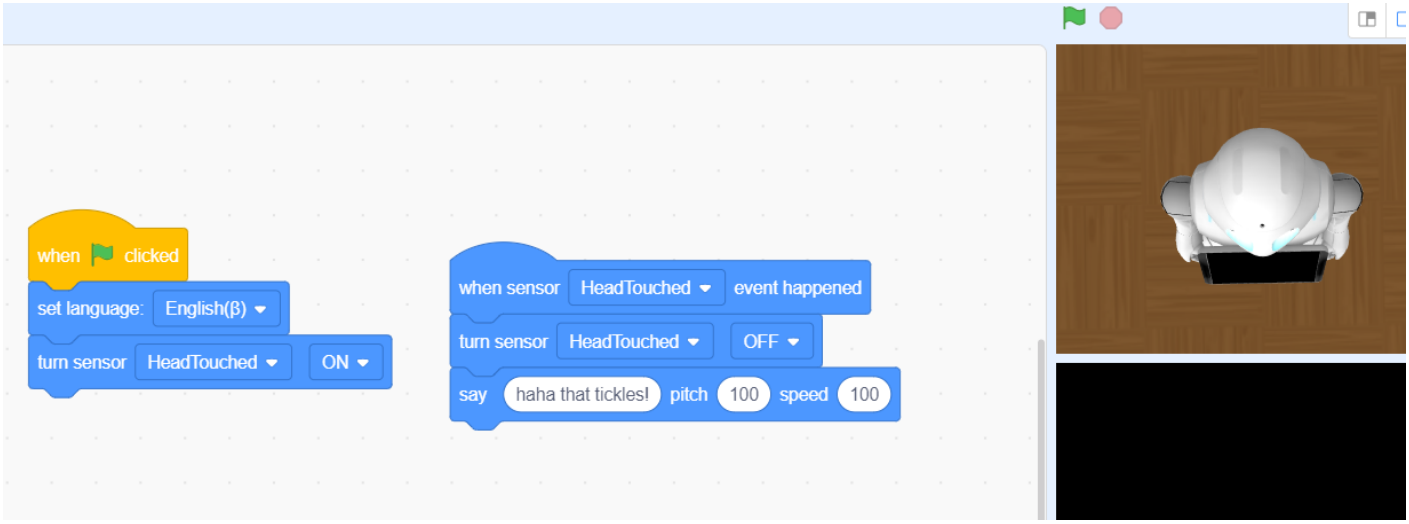
Zahlen raten – Pepper rät unsere Lieblingszahl

Erst als Aufgabe stellen.



Sensoren

Ein Roboter kann seine Umwelt wahrnehmen. Dafür hat er Sensoren. Er kann dies aber nicht ganz so gut wie wir Menschen. Wir werden uns zuerst seine Touchsensoren (Berührungssensoren) anschauen. Dies sind Sensoren, die es dem Roboter erlauben zu merken, ob er gerade am Kopf oder an den Handrücken berührt wird.



Weiterführend: Zählen, wie oft der Sensor am Kopf berührt wurde.