



# Girls Who Code At Home

Nicht perfekt, sondern mutig  
Scratch Debugging

## Übersicht über die Aktivität

Bei dieser Aktivität werden wir uns genauer anschauen, was es bedeutet, beim Programmieren **nicht perfekt, sondern mutig** zu sein! Nur, weil ein Programm nicht so funktioniert wie erwartet, heißt das nicht, dass man gescheitert ist. Es gibt viele Fähigkeiten, die wir erlernen und anwenden können, um Fehler zu beseitigen und unsere Produkte und Projekte zu verbessern.

**Debugging** ist eine Strategie, die Informatiker nutzen, um Probleme oder **Bugs** in ihrem Programm aufzuspüren und zu beseitigen. In dieser Aktivität wirst du dabei helfen, drei Programme in Scratch zu debuggen und in Ordnung zu bringen! Bevor du mit dieser Aktivität startest, empfehlen wir dir einen Blick auf unser Spotlight zu Frauen in der Technikbranche – Ayanna Howard. Ayannas erstes Projekt mit der NASA war es, einen Roboter zu bauen, der die Umwelt auf dem Mars analysiert. Erfahre mehr darüber, wie Ayanna Mut und Selbstbewusstsein einsetzt, um innovativ zu sein.

## Materialien

- [Online Scratch](#) oder [Offline Scratch](#)
- [Debugging-Challenge 1](#)
- [Debugging-Challenge 2](#)
- [Debugging-Challenge 3](#)
- Wahlweise: Handout zur Debugging-Challenge
- Lösungen zur Debugging-Challenge

## Spotlight zu Frauen in der Technikbranche: Ayanna Howard



Bildquelle: [Black Sci-Fi](#)

Vorbilder können in allen Formen auftreten. Tatsächlich war Ayanna Howards erstes Vorbild nicht einmal ein Mensch! Bionic Woman, eine Roboter-Superheldin, inspirierte Dr. Howard bereits im Kindesalter zu ihrer Leidenschaft für das Entwickeln von Robotern und für das Ingenieurwesen. Später studierte sie Elektrotechnik und Informatik an der Universität und erhielt einen Wirtschaftsabschluss.

Dr. Howard widmet sich weiterhin ihrer Leidenschaft für das Lernen, die Robotik und Elektrotechnik und arbeitet aktuell als Professorin für Biotechnologie des pädagogischen Robotik-Unternehmens Zyrobotics, von dem sie auch Mitgründerin und Chief Technology Officer ist. Darüber hinaus hat Dr. Howard in Zusammenarbeit mit der NASA Roboter entwickelt, die lernen, sich auf dem Mars zu bewegen.

Schau dir dieses [Video](#) an, um mehr über Dr. Howards erste Stelle bei der NASA zu erfahren. Sie spricht außerdem darüber, warum Diversität am Arbeitsplatz so wichtig ist und wie sie sich einer unangenehmen Situation bei der Arbeit gestellt hat.

## Überlegung

Informatikerin zu sein bedeutet wesentlich mehr, als nur gut programmieren zu können. Nimm dir etwas Zeit und denke darüber nach, was Ayannas Arbeit mit den Stärken zu tun hat, die du als IT-Wissenschaftlerin brauchst – Mut, Belastbarkeit, Kreativität und sinnvolle Ziele.



Dr. Howard zeigte Mut, als sie sich in einer unangenehmen Situation bei der Arbeit zu Wort meldete. Was würdest du in diesem Moment zu dir selbst sagen?

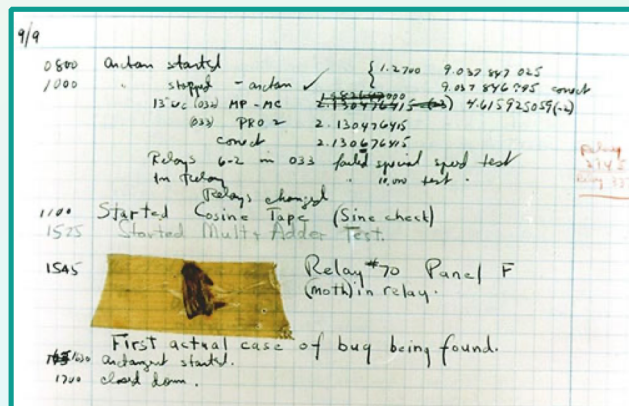
Besprich deine Ergebnisse mit einem Familienmitglied oder im Freundeskreis. Ermutige andere, mehr über Ayanna zu lesen und sich am Gespräch zu beteiligen.

## Schritt 1: Was ist Debugging? (5 Min.)

Denke an einen Moment, in dem du ein Problem lösen musstest, sei es für eine schwierige Aufgabe in der Schule oder beim Suchen eines verlorenen Objekts. Konntest du das Problem sofort lösen? Wenn wir versuchen, ein Problem zu lösen, passiert es oft, dass wir mehrere Misserfolge erleben, bevor wir das gewollte Ergebnis erzielen. Programmierer verbringen die meiste Zeit ihres Tages damit, Probleme in ihrem Code zu finden und zu beseitigen! Misserfolge sind ein großer Teil der Informatik 🐞.

Ein Fehler in einem Computerprogramm oder ein Hardware-Fehler wird **Bug** genannt. Der Prozess, diese Fehler oder Bugs zu finden und sie vom Computer, der Hardware oder Software zu entfernen, nennt man **Debugging**. Die Ursprünge dieser Begriffe gehen auf Grace Hopper zurück, die als eine der Ersten Pionierarbeit im Bereich Informatik leistete. Während sie an einem ihrer ersten Computer arbeiteten, fand Grace Hoppers Team eine Motte in dem Computer, die einen Fehler verursachte – eine echte Motte! Graces Tagebucheintrag mit der eingeklebten Motte gilt als die erste Aufzeichnung eines Computer-Bugs in der Geschichte. Er ist in dem Smithsonian Museum of American History in Washington, D.C. ausgestellt.

Die erste Aufzeichnung eines Computer-Bugs



Bildquelle: [Atlas Obscura](#)

Nimm dir einen Moment Zeit, um darüber nachzudenken, was die Aussage **Nicht perfekt, sondern mutig** für dich bedeutet. Warum ist es wichtig, beim Ausprobieren von etwas Neuem **mutig** zu sein, anstatt immer alles sofort **perfekt** zu machen? Wenn wir etwas Neues ausprobieren, kommt es oft vor, dass wir uns neuen Herausforderungen stellen müssen. Es ist wichtig, **belastbar** zu sein und zu versuchen, Fehler und Herausforderungen als Möglichkeit zum Lernen anzusehen. Debugging ist eine Möglichkeit, um aus Fehlern zu lernen. Diese Strategien helfen dir dabei, in der Zukunft mit größeren und komplizierteren Herausforderungen umzugehen.



## Schritt 2: Melde dich bei Scratch an und erkunde die Oberfläche (5–10 Min.)


Scratch ist eine Programmierplattform und eine blockbasierte Programmiersprache, die vom MIT entwickelt wurde und mit der du interaktive Geschichten, Spiele und Animationen programmieren kannst.

### 1. Registriere dich oder melde dich bei [Scratch](#) an.

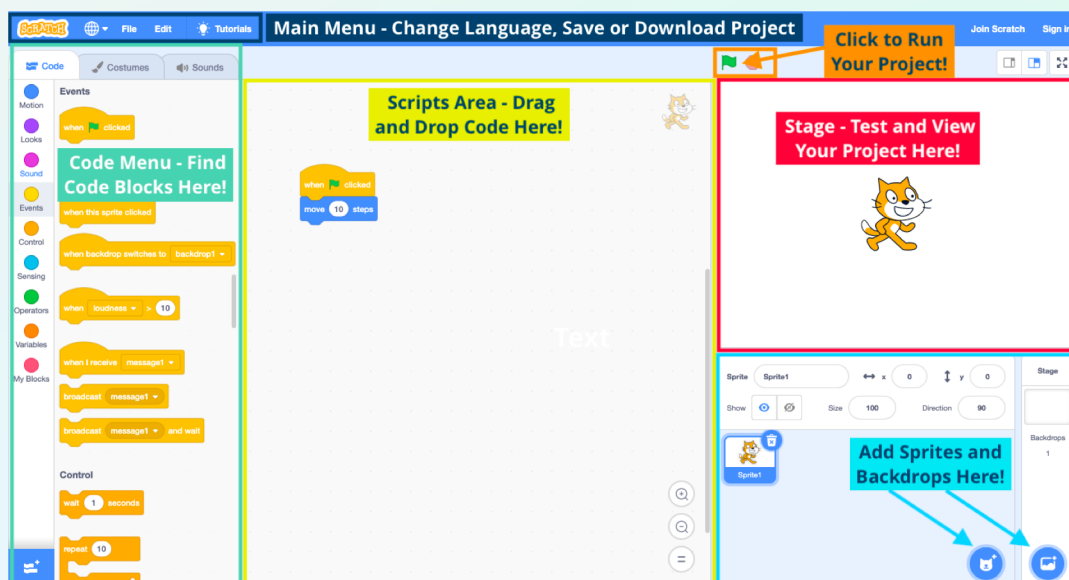
Um deine Arbeit auf der Online-Plattform von Scratch speichern zu können, musst du ein Konto erstellen, falls du noch keines hast. Folge bei der Registrierung den Anweisungen zur Erstellung eines Kontos. Wenn du unter 13 Jahre alt bist, brauchst du für die Registrierung die E-Mail-Adresse deiner Eltern. Wenn du kein Konto erstellen möchtest, kannst du auch die [Offline-Version von Scratch 3.0](#) herunterladen.

### 2. Erkunde die Oberfläche von Scratch.

Wenn du Scratch zum ersten Mal verwendest, nimm dir ein paar Minuten Zeit, um **ein**

**neues Projekt** zu erstellen , damit du die Scratch-Oberfläche kennenlernst. Du kannst dir auch dieses [Getting Started](#)-Tutorial von Scratch anschauen!

In der Abbildung unten findest du einige wichtige Abschnitte der Scratch-Plattform, mit denen du interagieren wirst.



## Schritt 3: Wiederholung der Debugging-Strategien von Scratch (5 Min.)

Bevor du mit dem Debugging beginnst, nimm dir Zeit, um die vorgeschlagenen Strategien zu wiederholen, um einen Bug in deinem Programm aufzuspüren und zu beseitigen.

1. **Beschreibe den Fehler (oder Bug).** Wie hast du bemerkt, dass es einen Bug im Programm gab? Denke darüber nach, was du erwartet hast und was tatsächlich passiert ist.
2. **Prüfe deinen Programmcode und denke darüber nach, welche Fehler du eventuell gemacht hast.** Lies deinen Code Zeile für Zeile durch. Du kannst sogar jede Codezeile laut vorlesen! Während du deinen Code noch einmal durchliest, denke darüber nach, welche Codezeilen (oder Blocks, wenn du Scratch verwendest) die Quelle des Fehlers oder des unerwarteten Ergebnisses sein könnten.
3. **Platziere in deinem Code Markierungen, die als Kontrollpunkte dienen.** Wenn dein Code lang ist, kann es hilfreich sein, ihn in verschiedene Abschnitte einzuteilen. In Scratch kannst du einen **Say**-Block verwenden, damit du weißt, an welcher Stelle in deinem Code sich dein Programm befindet. Diese Strategie kann dir dabei helfen, die möglichen Standorte des Bugs einzugrenzen.





Wenn beispielsweise deine erste Markierung erreicht ist (wenn dein Sprite den Text im Say-Block aufsagt, den du als Marker verwendest) und dein Programm weiter wie erwartet funktioniert, weißt du, dass sich der Bug wahrscheinlich nicht vor der ersten Markierung befindet. Wenn du den Bug in deinem Programm bemerkst, bevor du die zweite Markierung erreichst, ist es wahrscheinlich, dass sich der Bug in dem Teil des Codes zwischen der ersten und zweiten Markierung befindet.

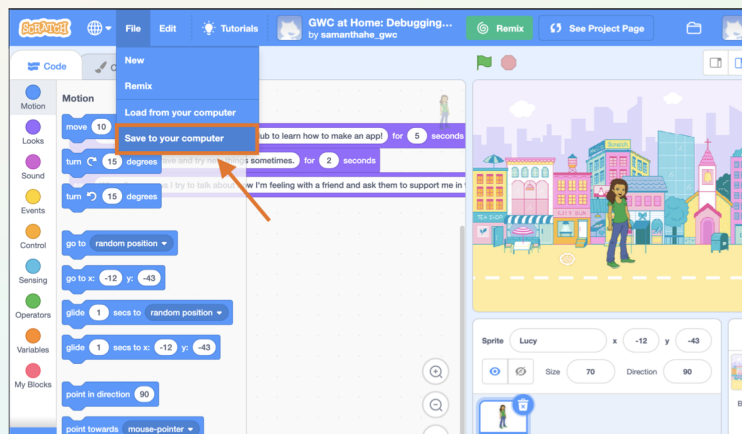
4. **Wenn du den Bug gefunden hast, versuche zu verstehen, warum der Code ein Bug ist.** Denke darüber nach, warum die Codezeile oder der Codeblock einen Fehler in deinem Programm verursacht. Einige häufige Fehler sind Rechtschreibfehler, das Platzieren von Codeblocks in der falschen Reihenfolge oder die Verwendung des falschen Blocks. Einige Programme haben Fehler-Hinweise, die beschreiben, warum eine bestimmte Codezeile ein Bug ist. Scratch verfügt nicht über diese Fehler-Hinweise, daher wirst du etwas mehr nachdenken müssen!
5. **Bringe deinen Code in Ordnung und teste ihn.** Versuche, die Codezeile in Ordnung zu bringen und dein Programm dann erneut auszuführen. Existiert der Bug immer noch? Wenn ja, versuche es mit einer anderen Lösung und probiere es nochmal! Tritt ein Bug in einem anderen Bereich auf? Dann gibt es vielleicht einen zweiten Bug im Programm. Befolge die gleichen Schritte, um deinen neuen Bug zu finden. Wenn es keine Fehler mehr in deinem Programm gibt, bedeutet das, dass du deinen Bug beseitigt hast!

## Schritt 4: Die Debugging-Challenge 1 angehen (5–10 Min.)

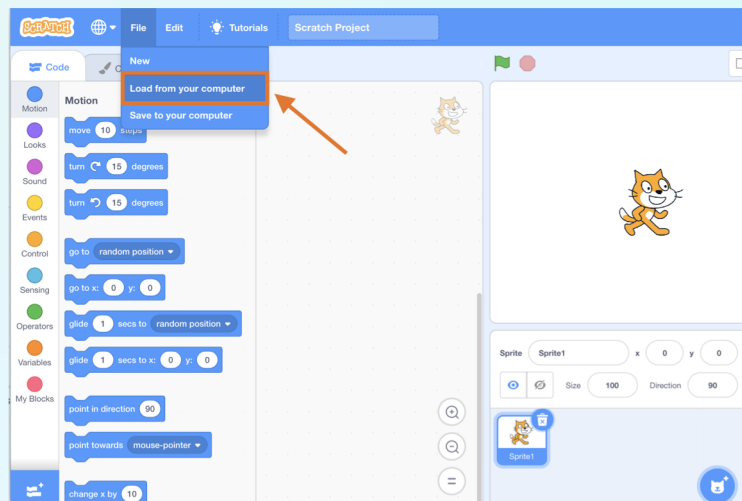
**Vorwissen:** Um diese Challenge zu lösen, musst du die Blöcke [Event](#) und [Looks](#) kennen. Schau dir dieses [Getting Started](#)-Tutorial an, um dich mit diesen Blöcken vertraut zu machen.




1. **Erstelle den Startercode neu.** Klicke auf die  Schaltfläche oben rechts, um eine Kopie des Projekts zu erstellen.
  - Wenn du den Scratch Offline Editor verwendest, musst du eine Kopie deines Projekts auf deinem Computer speichern. Klicke auf die  Schaltfläche, um den Projektcode einzusehen.
  - Klicke auf **File** oben in der Navigationsleiste und wähle die Option **Save to your computer** im Dropdown-Menü aus.



- Öffne den Scratch Offline Editor auf deinem Computer. Klicke auf **File** oben in der Navigationsleiste und wähle die Option **Load to your computer** im Dropdown-Menü. Suche deine gespeicherte Projektdatei auf deinem Computer und klicke auf **Open**.




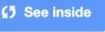

## Schritt 4: Die Debugging-Challenge 1 angehen (Fortsetzung)

2. **Teste das Programm und identifiziere den Bug.** In diesem Projekt stellt sich Lucy, unser Sprite (das Objekt, die Figur) dem Nutzer vor. Allerdings passiert nichts, wenn man auf die grüne Flagge klickt! Führe das Programm aus, indem du auf die grüne Flagge  klickst. Überprüfe den Code und versuche, den Bug zu identifizieren. Nutze das **Challenge-Handout** auf den Seiten 12–13, das dir dabei hilft, den Bug zu beseitigen und ihn zu verstehen.
3. **Beseitige den Bug und überprüfe deine Lösung [hier](#).**  
Auf Seite 14 findest du weitere Informationen zu diesem Bug!
4. **Denke über den Debugging-Prozess nach.** Denke darüber nach, was der Bug war und inwiefern er das Programm beeinflusst hat. Viele Programmierer haben einen Spickzettel mit häufigen Fehlern, während sie das Programmieren lernen. Das hilft ihnen, aus ihren eigenen Fehlern zu lernen, wenn sie andere Projekte programmieren.

## Schritt 5: Die Debugging-Challenge 2 angehen (5–10 Min.)

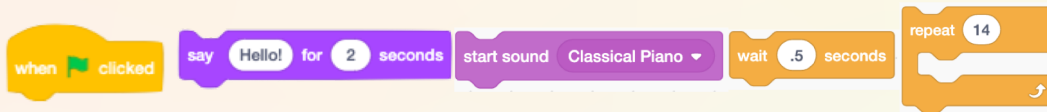
**Vorwissen:** Um diese Challenge lösen zu können, musst du die Blöcke [Event](#), [Looks](#), [Motion](#), [Sensing](#) und [Loops](#) kennen. Schaue dir das [Code a Cartoon](#)-Tutorial an, um dich mit diesen Blöcken vertraut zu machen.






1. **Erstelle den [Startercode](#) neu.** Klicke auf die  Schaltfläche oben rechts, um eine Kopie des Projekts zu erstellen.
  - Wenn du den Scratch Offline Editor verwendest, musst du eine Kopie deines Projekts auf deinem Computer speichern. Klicke auf die  Schaltfläche, um den Projektcode einzusehen.
  - Klicke auf **File** oben in der Navigationsleiste und wähle die Option **Save to your computer** im Dropdown-Menü aus.
  - Öffne den Scratch Offline Editor auf deinem Computer. Klicke auf **File** oben in der Navigationsleiste und wähle die Option **Load to your computer** im Dropdown-Menü. Suche deine gespeicherte Projektdatei auf deinem Computer und klicke auf **Open**.
2. **Teste das Programm und identifiziere den Bug.** In diesem Programm soll Avery in der Mitte beginnen und sich vorstellen. Dann geht sie nach rechts, bis sie die Ecke erreicht, um zum Programmierclub zu gelangen. Es funktioniert, wenn wir zum ersten Mal die grüne Flagge anklicken,  aber wenn wir sie noch einmal anklicken, ist Avery immer noch rechts, anstatt wieder in der Mitte des Bildschirms zu beginnen. Führe das Programm aus, indem du auf die grüne Flagge klickst. Überprüfe den Code und versuche, den Bug zu identifizieren. Nutze das **Challenge-Handout** auf den Seiten 12–13, das dir dabei hilft, den Bug zu beseitigen und ihn zu verstehen.
3. **Behebe die Bugs und überprüfe deine Lösung [hier](#).** Du hättest auch programmieren können, dass Avery von einer anderen Position startet als von denen, die in den Lösungen aufgelistet sind. Auf Seite 15 findest du weitere Informationen zu diesem Bug!
4. **Denke über den Debugging-Prozess nach.** Denke darüber nach, was der Bug war und inwiefern er das Programm beeinflusst hat. Viele Programmierer haben einen Spickzettel mit häufigen Fehlern, während sie das Programmieren lernen. Das hilft ihnen, aus ihren eigenen Fehlern zu lernen, wenn sie andere Projekte programmieren.

## Schritt 6: Die Debugging-Challenge 3 angehen (5–15 Min.)

**Vorwissen:** Um diese Challenge zu lösen, musst du die Blöcke [Event](#), [Looks](#), [Sounds](#), [Wait](#) und [Loops](#) kennen. Schaue dir das [Code a Cartoon](#)-Tutorial an, um dich mit diesen Blöcken vertraut zu machen.



1. **Erstelle den [Startercode](#) neu.** Klicke auf die  Schaltfläche oben rechts, um eine Kopie des Projekts zu erstellen.
  - Wenn du den Scratch Offline Editor verwendest, musst du eine Kopie deines Projekts auf deinem Computer speichern. Klicke auf die  Schaltfläche, um den Projektcode einzusehen.
  - Klicke auf **File** oben in der Navigationsleiste und wähle die Option **Save to your computer** im Dropdown-Menü aus.
  - Öffne den Scratch Offline Editor auf deinem Computer. Klicke auf **File** oben in der Navigationsleiste und wähle die Option **Load to your computer** im Dropdown-Menü. Suche deine gespeicherte Projektdatei auf deinem Computer und klicke auf **Open**.
2. **Teste das Programm und identifiziere den Bug.** Wenn du auf die grüne Flagge  klickst, sollte Ada dir etwas über sich erzählen und dann zu der Musik tanzen. Wenn Ada zu tanzen beginnt, läuft etwas mit der Musik schief! Sie beginnt jedes Mal, wenn Ada sich bewegt, immer wieder von vorn. Wie können wir den Code in Ordnung bringen, sodass die Musik immer weiterspielt, während Ada tanzt? Führe das Programm aus, indem du auf die grüne Flagge klickst. Überprüfe den Code und versuche, den Bug zu identifizieren. Nutze das **Challenge-Handout** auf den Seiten 12–13, das dir dabei hilft, den Bug zu beseitigen und ihn zu verstehen.
3. **Behebe die Bugs und überprüfe deine Lösung [hier](#).** Du hättest auch programmieren können, dass Avery von einer anderen Position startet als von denen, die in den Lösungen aufgelistet sind. Auf Seite 16 findest du weitere Informationen zu diesem Bug!
4. **Denke über den Debugging-Prozess nach.** Denke darüber nach, was der Bug war und inwiefern er das Programm beeinflusst hat. Viele Programmierer haben einen Spickzettel mit häufigen Fehlern, während sie das Programmieren lernen. Das hilft ihnen, aus ihren eigenen Fehlern zu lernen, wenn sie andere Projekte programmieren.

## Schritt 7: Teile dein Werk (5 Min.)

### 1. Teile dein Projekt auf Scratch.

Wenn du die Challenge gelöst hast, klicke auf die „Share“-Schaltfläche in Scratch. Erzähle im Abschnitt „Anleitungen“, wie du den Bug gelöst hast!

### 2. Zeige der Welt, wie du mit Girls Who Code at Home Herausforderungen bewältigst!

Vergiss nicht, deine Projekte in den Sozialen Medien zu teilen. Tagge @girlswhocode und #codefromhome. Vielleicht werden wir dich sogar auf unserem Profil vorstellen!



# Handout zur Debugging-Challenge

**Anleitungen:** Während du die Challenges durchgehst, denke über die folgenden Fragen nach.

- Was ist der Bug? Was sollte passieren? Gibt es verschiedene Bugs?
- Wie hat sich der Bug auf das Programm ausgewirkt und wieso?
- Wie hast du den Bug beseitigt?

## Debugging-Challenge 1 (5–10 Min.)

Startercode: <https://scratch.mit.edu/projects/387548698/>

**Bug:**

**Wie er sich auf das Programm ausgewirkt hat und wieso:**

**Wie hast du den Bug beseitigt?**

# Handout zur Debugging-Challenge

## Debugging-Challenge 2 (5–10 Min.)

Startercode: <https://scratch.mit.edu/projects/387549618/>

**Bug:**

**Wie er sich auf das Programm ausgewirkt hat und wieso:**

**Wie hast du den Bug beseitigt?**

## Debugging-Challenge 3 (5–15 Min.)

Startercode: <https://scratch.mit.edu/projects/387851932/>

**Bug:**


**Wie er sich auf das Programm ausgewirkt hat und wieso:**

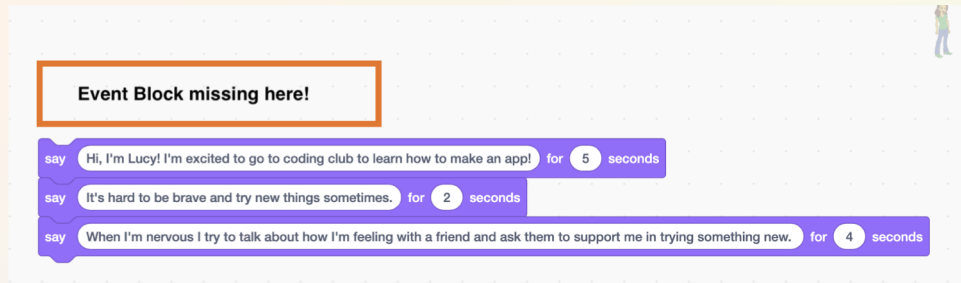
**Wie hast du den Bug beseitigt?**

## Debugging-Challenge 1 Lösung


Startercode mit Bug: <https://scratch.mit.edu/projects/387548698/>

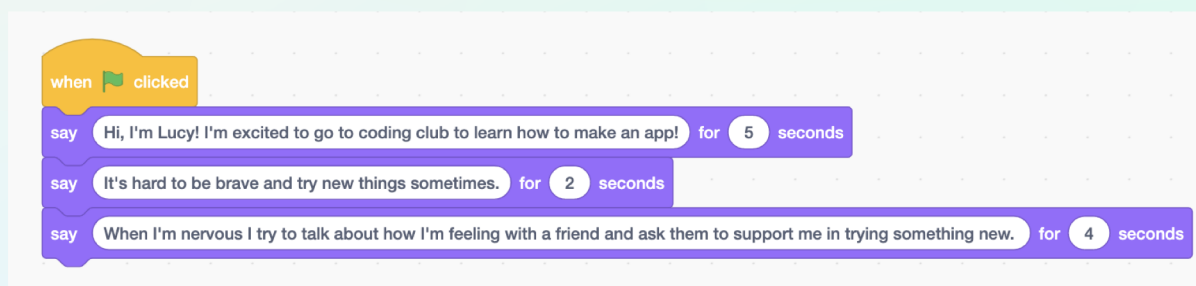
Beispiel für Lösungscode: <https://scratch.mit.edu/projects/388218006/>

**Bug:** Wenn man auf die grüne Flagge  klickt, sollte Lucy ihre Vorstellung beginnen. Allerdings passiert nichts. Der Grund war, dass kein Event-Block im Code enthalten war.



**Wie er sich auf das Programm ausgewirkt hat und wieso:** Ohne einen Event-Block weiß der Computer nicht, wann er unseren Code ausführen soll. Es passiert also nichts, wenn man auf die grüne Flagge klickt, da wir dem Computer nicht sagen, dass er etwas tun soll.

**Wie hast du den Bug beseitigt?** Wir müssen dem Computer sagen, dass der Code ausgeführt werden soll, **wenn auf die grüne Flagge geklickt wird**. Wir haben den  Block verwendet und ihn vor die anderen Codeblocks gesetzt.

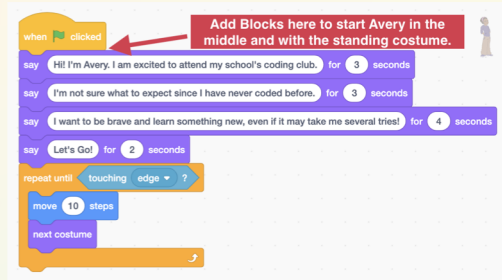


## Debugging-Challenge 2 Lösung

Startercode mit Bug: <https://scratch.mit.edu/projects/387549618/>

Beispiel für Lösungscode: <https://scratch.mit.edu/projects/388333942/>

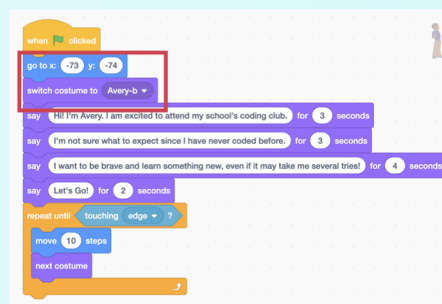
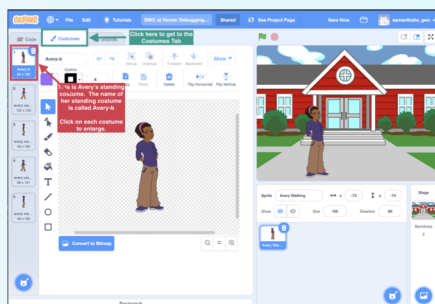
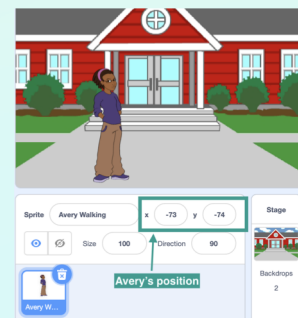
**Bug:** Dieses Programm enthält **zwei** Bugs. Nachdem du ein zweites Mal auf die grüne Flagge geklickt hast, startet Avery nicht mehr von demselben Startpunkt in der Mitte des Bildschirms. Stattdessen startet sie von dort aus, wo sie beim letzten Mal aufgehört hat – an der rechten Seite des Bildschirms und in ihrem Spazieroutfit. Der erste Bug ist ihre Position und der zweite Bug ist ihre Kleidung (oder Look).



**Wie er sich auf das Programm ausgewirkt hat und wieso:** In diesem Programm soll Avery in der Mitte beginnen und sich vorstellen. Dann geht sie nach rechts, bis sie die Ecke erreicht, um zum Programmierclub zu gelangen. Da wir keinen Code haben, der dazu führt, dass Avery beim Klicken auf die grüne Flagge von einer festen Position startet, beginnt Avery ihre Codesequenz immer dort, wo sie beim letzten Mal aufgehört hat. In diesem Fall ist dies der rechte Bildschirmrand.

**Wie hast du den Bug beseitigt?** Damit Avery immer von derselben Position aus startet, müssen wir `go to x: -73 y: -74` benutzen. Der einfachste Weg, um Averys Startposition auszuwählen, ist es, deine Maus zu nutzen und Avery dorthin zu ziehen, von wo aus sie starten soll. Achte darauf, wie sich die Nummern hinter „x:“ und „y:“ Averys Position entsprechend verändern. Du kannst dies in der Sprite-Beschreibung unter der Scratch-Bühne sehen. Wir nutzen den Block „go to“ als ersten Block nach unserem Event-Block `when green flag clicked`.


Nun wollen wir, dass Avery mit der Kleidung beginnt, die sie stehend darstellt. Zuerst müssen wir `switch costume to Avery-b` benutzen, nachdem wir ihre Startposition festgelegt haben, um sicherzustellen, dass sie immer in ihrem Steh-Outfit startet. Wenn du dir nicht sicher bist, welches Outfit ihre Standposition ist, schau im Tab **Costumes** nach und merke dir den Namen des Outfits, mit dem sie starten soll. Wir programmieren diese zwei neuen Blöcke zuerst, da wir unsere Figur an dieselbe Stelle setzen möchten, wenn auf die grüne Flagge geklickt wird, bevor Avery zu sprechen und zu laufen beginnt!

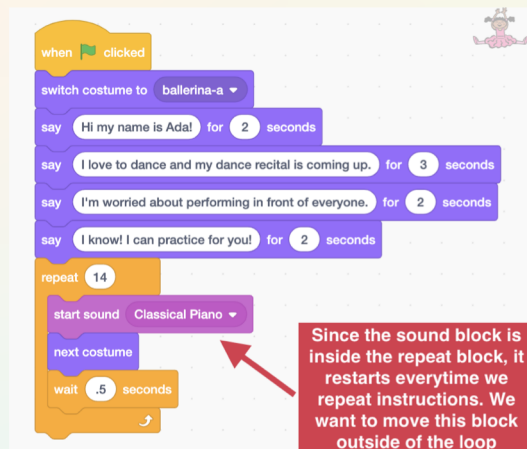


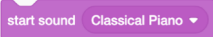
## Debugging-Challenge 3 Lösung

Startercode mit Bug: <https://scratch.mit.edu/projects/387851932/>

Beispiel für Lösungscode: <https://scratch.mit.edu/projects/388346112/>

**Bug:** Wenn du auf die grüne Flagge  klickst, sollte Ada dir etwas über sich erzählen und dann zu der Musik tanzen. Wenn Ada zu tanzen beginnt, läuft etwas mit der Musik schief! Sie beginnt jedes Mal, wenn Ada sich bewegt, immer wieder von vorn. Der Bug hat etwas mit der Sequenz oder der Reihenfolge zu tun, in der die Blöcke angeordnet sind.



**Wie er sich auf das Programm ausgewirkt hat und wieso:** Du siehst, dass unser Sound-Block  in einer Wiederholungsschleife programmiert ist. Das bedeutet, dass die Musik jedes Mal, wenn unser Code eine Schleife dreht, wieder von vorn beginnt.

**Wie hast du den Bug beseitigt?** Wir müssen unseren Sound-Block aus der Schleife entfernen, aber wohin? Wir müssen uns noch einmal überlegen, was wir erreichen wollen. Experimentiere ein wenig und platziere den Sound-Block vor und hinter dem Repeat-Block. Wenn wir den Sound-Block hinter der Wiederholungsschleife platzieren, spielt die Musik, nachdem Ada getanzt hat, und das wollen wir nicht. Wenn wir den Sound-Block vor der Wiederholungsschleife platzieren, spielt zuerst die Musik und dann tanzt Ada. Das wollen wir! Du kannst entscheiden, ob die Musik starten soll, wenn sie tanzt oder direkt am Anfang, wenn sie sich vorstellt.

