



# Girls Who Code At Home

Kann ich dir helfen?  
Chatbot in Python

## Übersicht über die Aktivität

Bei dieser Aktivität wirst du grundlegende Informatikkonzepte in Python erkunden, während du lernst, wie du deiner Gemeinschaft helfen kannst, indem du einen Chatbot, nämlich ein automatisiertes Hilfe-System, erstellst! Vermutlich hast du bereits mit einem Chatbot interagiert – eventuell sogar, ohne dass du das bemerkt hast. Zu den üblichen Chatbots gehören: Apples Siri, Amazons Alexa, der Google Assistant, Lyft und seit Neuestem der von der World Health Organization für [COVID-19-Fakten](#) erstellte Chatbot. Bevor du beginnst, zu entwerfen und zu programmieren, empfehlen wir dir, hier unsere Sonder-Info Frauen im Tech-Rampenlicht: Erica Kochi anzuschauen. Als Mitgründerin der Innovation Unit (Innovationseinheit) von UNICEF konstruierte Erica einen Chatbot, der Betreuung und Beistand für Menschen in Nigeria und Ruanda bereitstellt.

## Materialien

- [Trinket Editor](#)
- [Starter Trinket Code](#)
- [Beispiel Chatbot-Projekt](#)
- Wahlweise: Planungshilfe
- Wahlweise: Kugelschreiber/Bleistifte/Marker

## Frauen im Rampenlicht der Technik: Erica Kochi



Erica Kochi ist Mitbegründerin von und leitet gemeinsam mit anderen [UNICEFs](#) Innovation Unit, die darauf abzielt, die weltweite Gesundheit durch die Nutzung technologischer Innovationen zu optimieren. Ihr Projekt [RapidSMS](#) nutzt Handys und SMS-Mitteilungen, um die Datensammlung für Gesundheits- und Aufklärungsdienste zu ermöglichen.

Die in Zusammenarbeit mit Partnern erfolgende Arbeit von Erica Kochi half bei der Entwicklung von Open-Source-Technologien, die mehr als 7 Millionen Geburten in Nigeria registrierten und Fürsorge für Tausende schwangere Frauen in ganz Ruanda bereitstellte. Im Jahr 2013 wurde sie in der Liste der 100 „World's Most Influential People“ (einflussreichsten Menschen der Welt) des TIME Magazins aufgeführt!

Schau dir dieses [Video](#) an, um mehr über Erica Kochi und einige der Projekte, denen sie bei der [UNICEF](#) nachgeht, zu erfahren.

### Zeit zu reflektieren

Informatikerin zu sein, bedeutet wesentlich mehr, als nur gut programmieren zu können. Nimm dir etwas Zeit, um darüber nachzudenken, inwiefern Erica und ihre Arbeit jene Stärken ausdrücken, um deren gezielten Aufbau sich bedeutende IT-Wissenschaftler\*innen bemühen – Mut, Hartnäckigkeit, Kreativität und sinnvolle Ziele.



UNERSCHROCKENHEIT

Ein Großteil von dem, was Erica Kochi tut, besteht darin, leidenden Personen zu helfen. Inwiefern könnte dies im Rahmen einer Karriere herausfordernd sein?

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

## Schritt 1: Erkundungen (5 Min.)

In diesem Tutorium werden wir einen personalisierten Chatbot in Python erstellen! Ein **Chatbot** ist ein Computerprogramm, das Gespräche mit einer realen Person simuliert. Es ist eine sehr simple Form Künstlicher Intelligenz, oder „KI“. Zu den Chatbots, die du eventuell schon benutzt, gehören: [Starbucks Barista](#), [Apples Siri](#), [Amazons Alexa](#) und [der Google Assistant](#).

Nimm dir 5 Minuten Zeit dafür, einige der Eigenschaften [dieses Chatbots](#), den wir hinsichtlich Frauen in der Welt der Technologie erstellt haben, zu erkunden. Für unser Beispiel haben wir jeweils folgendes Thema, Publikum und Ziel gewählt:

- **Thema des Projekts:** Frauen/Technologie
- **Publikum:** Personen, die etwas über Frauen, die in der Welt der Technologie arbeiten, erfahren möchten
- **Ziel:** Auf Technologie basierende Witze teilen und Informationen über Frauen in der Welt der Technologie bereitstellen

Während du das Beispielprojekt erkundest, denke über folgende Punkte nach:

- Wie wird der Chatbot vorgestellt?
- Welche Arten von Fragen werden gestellt? Erfüllt dies die Zielsetzung?
- Welche Funktionen aus dem Beispiel würdest du gerne in dein eigenes Projekt integrieren? Was würdest du in deinem eigenen Projekt gerne anders machen?

## Schritt 2: Mach ein Brainstorming zu den Funktionen und plane dein Projekt (10 Min.)

Nachdem du nun Gelegenheit hattest, das Beispielprojekt zu erkunden, wäre es eine gute Idee, dir als Nächstes etwas Zeit dafür zu nehmen, zuerst einen Spielplan zu erstellen. Nutze diese Zeit dazu, zu bestimmen, was dein Projekt leisten soll und was das Ziel ist. Du solltest eventuell das Planungsdokument (auf den Seiten 16–18) als Hilfestellung nutzen, um deine Ideen zu erfassen und zu ordnen.

### 1. Wähle das betreffende Thema, Publikum und Ziel für deinen Chatbot.

Überlege dir, auf welches Thema du dich bei deinem Chatbot konzentrieren möchtest und was er erreichen sollte. Du kannst einen Chatbot erstellen, der informativ, humorvoll, gemeinschaftsbildend ist oder einen Quiz-Chatbot zu Fakten. Falls du dich festgefahren fühlst, findest du hier einige mögliche Projekt-Ideen:

- Ein Chatbot, der aufmunternde Worte für jene bereithält, die sich zu Hause gelangweilt und/oder isoliert fühlen
- Ein Chatbot, der Vorschläge gibt, wie man zu Hause mehr Spaß haben kann
- Ein Chatbot, der lustige Witze zum Besten gibt
- Ein Chatbot, der Tatsachen hinsichtlich deiner Lieblingsshow, deines Lieblingskünstlers bzw. -musikers oder deines Hobbys mitteilt

Wenn du die Wahl hinsichtlich des Publikums für deinen Chatbot triffst, überlege dir, welches Zielpublikum gerne dein Produkt benutzen würde. Berücksichtige dabei, was es gerne wissen möchte und wie du seine Aufmerksamkeit erregen kannst.



## Schritt 2: Plane dein Projekt (Fortsetzung)

2. Wähle drei „Ja-oder-Nein-Fragen“, die dein Chatbot stellen sollte und entwerfe Antworten für jede mögliche Frage.

Überlege dir, welche Fragen du stellen möchtest und inwiefern diese Fragen zu deinem Thema und dem Ziel des Projekts passen. Was geschieht, falls die Antwort des Benutzers auf deine Frage „ja“ lautet? Und was, falls die Antwort „nein“ ist? Was geschieht, wenn der Benutzer weder „ja“ noch „nein“ eingibt? Wie sollte dein Chatbot auf diesen Benutzerfehler reagieren?

## Schritt 3: Beginne, mit Trinket zu arbeiten (10 Min.)

**Python** ist eine textbasierte Programmiersprache, was bedeutet, dass sämtliche Befehle geschrieben werden müssen! In Python können die Programmierer\*innen zahlreiche Variablen, Datenstrukturen und Funktionen zur Informationsspeicherung und Befehlsausführung verwenden! Da Python textbasiert ist, ist es etwas schwieriger als andere Sprachen wie beispielsweise Scratch, aber es ist alles andere als unmöglich damit klarzukommen!

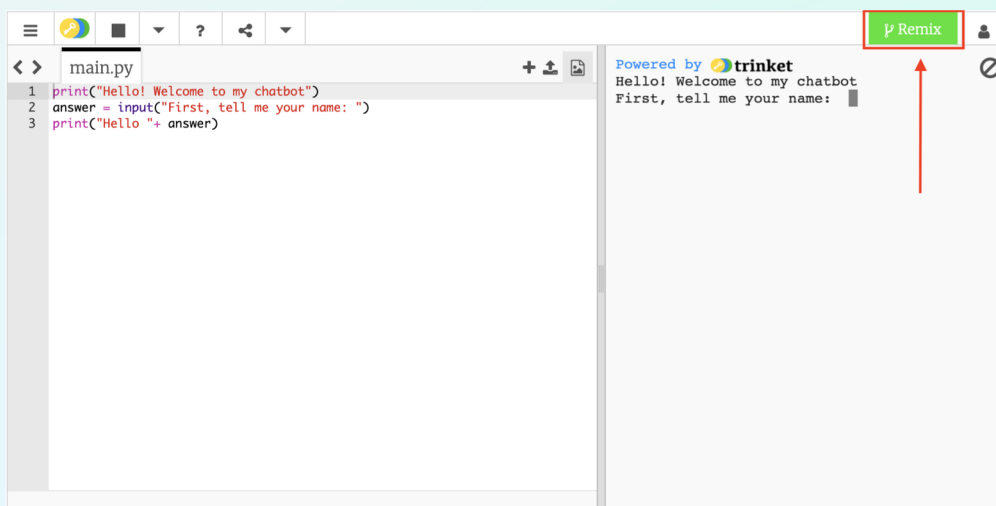
Um unseren Python-Code zu schreiben und auszuführen, werden wir die Programmierplattform [Trinket](https://trinket.io) benutzen. Es gibt viele unterschiedliche Programme, die zum Schreiben und Ausführen von Python-Code genutzt werden können. Wir haben dieses gewählt, da man es direkt im Browser nutzen kann!

1. **Registriere dich oder melde dich dort an:** [Trinket.io](https://trinket.io)

Um deine Arbeit bei Trinket 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.

2. „Remix“ diesen [Beginnercode](#) für deinen Chatbot.

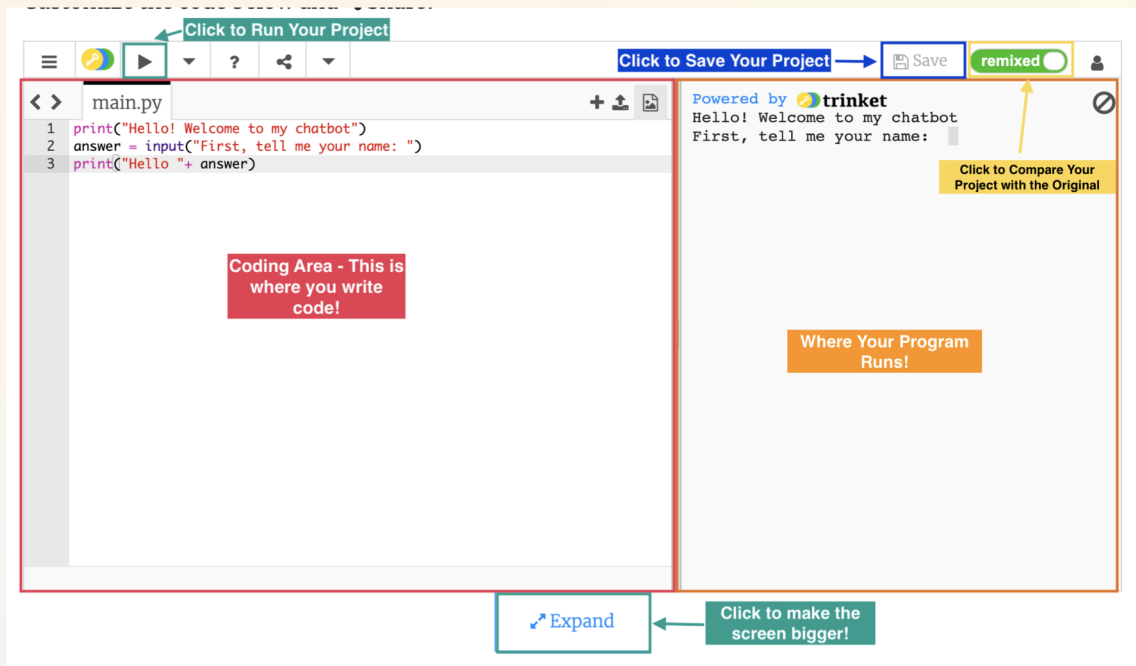
Nachdem du zum Beginnercode navigiert bist, solltest du zuerst auf **Remix** klicken oder eine Kopie des Projekts speichern. Klicke auf die **Remix**-Schaltfläche auf der linken Seite des Trinket-Fensters.



## Schritt 3: Beginne, mit Trinket zu arbeiten (Fortsetzung)

### 3. Erkunde die Oberfläche von Trinket.

Wenn du neu bei Trinket bist, nimm dir einige Minuten Zeit, um dir die Trinket-Oberfläche anzuschauen. Mache dich damit vertraut, wie du deinen Code speichern (Save) und ausführen lassen (Run) kannst.



## Schritt 4: Erkunde das Beginnerprojekt (2 Min.)

Lass uns die [Beginnerdatei](#) betrachten. Klicke auf die Schaltfläche für Run \_\_, um zu sehen, was dieser Code macht.

Du wirst feststellen, dass es eine kurze Vorstellung gibt, indem das Programm sagt:

Hallo! Willkommen bei meinem Chatbot.

Anschließend folgt eine Frage:

Freut es dich, mit diesem Programm zu interagieren?

Versuche „ja“ einzutragen und die Antwort zu sehen; dann schreibe „nein“ und dann schreibe „Ja“.

Du wirst feststellen, dass das Programm auf alle drei dieser Antworten eine unterschiedliche Antwort gibt. Da Python eine textbasierte Sprache ist, sind sowohl die Rechtschreibung als auch die Groß- und Kleinschreibung von Bedeutung! Die Antworten „ja“ und „Ja“ werden vom Computer als unterschiedlich gelesen.

Die wichtigsten Aspekte eines Chatbot sind dessen Fähigkeit, (1) den Benutzer um die Beantwortung einer Frage zu bitten und (2) mit dem Benutzer zu sprechen, indem eine Antwort ausgegeben wird. Als Erstes werden wir lernen, wie wir mittels des Codes mit dem Benutzer sprechen können, dann werden wir erörtern, wie Fragen gestellt und Antworten gegeben werden.

## Schritt 5: Füge eine Vorstellung hinzu (4 Min.)

Damit der Computer zum Benutzer „spricht“, schreiben wir `print`-Anweisungen in Code und die Anweisung wird dann auf der linken Seite des Trinket-Fensters angezeigt.

Lass uns einen Blick auf die erste Codezeile in der [Beginner-Datei](#) werfen. Klicke auf die Schaltfläche für Run \_\_, um zu sehen, was dieser Code macht.

Du kannst sehen, dass die erste Codezeile `print("Hallo! Willkommen bei meinem Chatbot")` lautet und die erste Zeile auf der **Ergebnis**-Seite `Hallo! Willkommen bei meinem Chatbot`.

Schauen wir uns mal die Schlüsselsymbole und Bedingungen für den Gebrauch einer `print`-Anweisung an:

`print("Beispieltext")`

- `print`: Dieses Schlüsselwort lässt den Computer wissen, dass er etwas auf der Ergebnis-Seite ausgeben soll
- `()`: Die Klammern lassen den Computer wissen, dass sich das, was er ausgeben soll, innerhalb der Klammern befindet.
- `" "`: Die Markierung durch doppelte Anführungszeichen (englische, nicht deutsche) lassen den Computer wissen, dass es sich bei dem, was sich zwischen ihnen befindet, um Worte handelt.
- `Beispieltext`: Zwischen die Anführungszeichen können wir jegliche Wörter schreiben, die identisch auf der **Ergebnis**-Seite ausgegeben werden.

### Versuche es mal selbst!

Füge eine Codezeile hinzu, um per `print` eine kurze Vorstellung deines Chatbots ausgeben zu lassen. In diese Beschreibung solltest du eventuell das Thema und das Ziel einfließen lassen.

Klicke auf die Schaltfläche Run \_\_, um zu sehen, ob der Computer deine Vorstellung ausgibt.

### Debugging-Tipps:

- Syntaxfehler: Überprüfe, ob deine Frage sich zwischen doppelten Anführungszeichen befindet, eines vor und eines nach der Frage.
- Syntaxfehler: Überprüfe, ob die Frage sich innerhalb von Klammern befindet und die Codezeile mit einer schließenden Klammer **endet**.

## Schritt 6: Stelle deine erste Frage (5 Min.)

Damit du deine erste Frage stellen kannst, lass uns zunächst zwei wichtige Informatik-Begriffe erörtern: **Input** und **Output**.

**Input** (Informationseingabe) findet statt, wenn dem Computer jegliche Art von Information gegeben wird. Manchmal kann das durch das Drücken einer Taste geschehen, durch das Einstöpseln eines Geräts wie beispielsweise einer Maus oder in unserem Fall durch das Schreiben einer Antwort.

**Output** (Ausgabe) findet statt, wenn der Computer Informationen an einen anderen Prozess übergibt. Die Antwort seitens des Benutzers (entweder „ja“ oder „nein“) ist der **Input**, während die Antwort des Computers der **Output** des Prozesses einer Fragestellung ist.



Um in Python einen Input für eine Frage zu erhalten, verwenden wir den Befehl `input()`.

Wie du sehen kannst, lautet die zweite Codezeile `answer = input("Freut es dich, mit diesem Programm zu interagieren?")` und die erste Zeile auf der **Ergebnis**-Seite „Freut es dich, mit diesem Programm zu interagieren?“

Schauen wir uns mal die Schlüsselsymbole und Bedingungen für den Gebrauch einer **Input**-Anweisung an:

```
answer = input("Beispielfrage")
```

- **answer**: Dies ist eine **Variable**, welche die Antwort des Benutzers speichert. In unserem Fall wurde diese Variable „answer“ genannt.
- **=**: Das Gleichheitszeichen zeigt die Zuweisung oder Wiederzuweisung eines Wertes an die **Variable** answer.
- **input**: Dieses Schlüsselwort lässt den Computer wissen, dass er dem Benutzer eine Frage stellen soll.
- **()**: Die Klammern lassen den Computer wissen, dass er den innerhalb der Klammern stehenden Text ausgeben soll.
- **" "**: Die Markierung durch doppelte Anführungszeichen lässt den Computer wissen, dass es sich bei dem, was sich zwischen ihnen befindet, um Worte und einen Teil der Input-Mitteilung handelt.
- **Beispielfrage**: Zwischen die Anführungszeichen können wir jegliche Wörter schreiben, und diese werden identisch auf der **Ergebnis**-Seite ausgegeben.

Eine **Variable** ist ein Informatik-Begriff, und Variablen werden dazu verwendet, Informationen (Daten) in einem Computerprogramm zu speichern. Variablen werden mit einem Namen bezeichnet, damit man in einem Programm leicht auf sie Bezug nehmen und sie ändern kann.


## Schritt 6: Stelle deine erste Frage (Fortsetzung)

### Versuche es mal selbst!

Schau auf deinem Arbeitsblatt für die Planung nach und stelle dem Benutzer deine erste Frage. Füge eine neue Codezeile hinzu, die dem gleichen Format wie dem unten angegebenen entspricht.

```
answer = input("Frage Nr. 1")
```

Ersetze **Frage Nr. 1** durch deine tatsächliche Frage.

Klicke auf die Schaltfläche Run , um zu sehen, ob der Computer deine Frage stellt.

### Debugging-Tipps:

- Syntaxfehler: Überprüfe, ob deine Frage sich zwischen doppelten Anführungszeichen befindet, eines vor und eines nach der Frage.
- Syntaxfehler: Überprüfe, ob die Frage sich innerhalb von Klammern befindet und die Codezeile mit einer schließenden Klammer **endet**.
- Syntaxfehler: Überprüfe, ob du nach dem Namen der Variable „answer“ auch ein Gleichheitszeichen geschrieben hast.

## Schritt 7: Auf eine Antwort reagieren (10 Min.)

Du hast vielleicht bemerkt, dass der Computer eine Frage stellt, doch ansonsten nichts tut, um zu reagieren, nachdem du eine Antwort eingetragen hast. Das liegt daran, dass wir dem Computer nicht mitgeteilt haben, dass er irgendetwas mit diesen Daten anfangen soll!

Wenn wir den Code schreiben, `answer = input("question#1")` wird die Antwort des Benutzers **gespeichert**, und zwar in der Variable `answer`. Auf diese Weise werden wir feststellen, ob der Benutzer mit „ja“ oder „nein“ geantwortet hat.

Zunächst müssen wir **bedingte** Anweisungen verwenden, um die dem Benutzer gewährten Auswahlmöglichkeiten zu vergleichen („ja“ oder „nein“). Eine **bedingte** Anweisung prüft, ob eine Gruppe von Regeln (oder Aussagen) erfüllt wird und entscheidet dann, welche Aktionen ausgeführt werden, falls die Regelgruppe oder die Aussage wahr oder falsch ist. Hinsichtlich unserer Frage gibt es drei Antwortmöglichkeiten, entweder „ja“ oder „nein“, oder aber jegliche andere Antwort.

Zur Anwendung von Bedingungen in Python müssen wir die Schlüsselworte `if`, `elif` und/oder `else` benutzen. Lass uns einen Blick darauf werfen, wie in einem Chatbot Bedingungen verwendet werden, um die geeignete Antwort auf jede mögliche Antwort zu bestimmen.

## Schritt 7: Auf eine Antwort reagieren (Fortsetzung)

```
answer = input("Freut es dich, mit diesem Programm zu interagieren?")
```

```
if (answer == "ja"):
    print ("Ich bin auch so froh!")
elif (answer == "nein"):
    print ("Aua, naja, ich hoffe, ich kann deine Meinung ändern!")
else:
    print ("Hmm... Anscheinend verstehe ich deine Antwort nicht.")
```

- **if**: Schlüsselwort, das eine if-Anweisung (falls) festlegt
- **elif**: Schlüsselwort, das eine else-if-Anweisung (sonst-falls) festlegt. Dies ist eine optionale Anweisung, muss aber **nach** einer if-Anweisung erfolgen.
- **else**: Schlüsselwort, das eine else-Anweisung (sonst) festlegt. Dies ist eine optionale Anweisung, muss aber **nach** einer if-Anweisung erfolgen.
- **()**: Klammern sind optional. Sie werden um die Bedingung herum hinzugefügt, um dazu beizutragen, dass unser Code geordnet und einfacher lesbar bleibt.
- **answer**: Dies ist eine **Variable**, welche die Antwort des Benutzers speichert. In unserem Fall wurde diese Variable „answer“ genannt.
- **==**: Das doppelte Gleichheitszeichen ist ein Vergleichs-Operator. Es wird dazu genutzt, die als Variable gespeicherte Antwort mit einem anderen Wert zu vergleichen, in unserem Fall entweder „ja“ oder „nein“.
- **„yes“/„no“**: Die doppelten Anführungszeichen werden dazu genutzt, den Computer anzuweisen, den zwischen diesen befindlichen Wert als Text zu lesen. Da wir die Antworten mit diesen Antworten in Textform vergleichen wollen, verwenden wir doppelte Anführungszeichen vor und nach dem „ja“ und dem „nein“.
- **:** Der Doppelpunkt lässt das Programm wissen, dass das Ende der bedingten Anweisung erreicht wurde.
- **Einrückung**: Alle Codezeilen, die ausgeführt werden sollen, falls eine Bedingung zutrifft, müssen nach einer if-Anweisung eingerückt werden. Dies lässt den Computer wissen, welche Codezeilen ausgeführt werden sollen. Siehe oben das Beispiel, wo der print-Befehl eingerückt ist.

Erinnere dich daran, dass wir die Variable **answer** dazu verwenden, die Antwort des Benutzers auf unsere Frage zu *speichern*. Die erste **if**-Anweisung vergleicht zunächst, ob die Antwort „**yes**“ lautet. Falls die Antwort „**ja**“ ist, wird die unter der **if**-Anweisung eingerückte Codezeile ausgegeben. Falls **answer** nicht „**ja**“ ist, gehen wir zur nächsten bedingten Anweisung, dem **elif**. Diese vergleicht dann die Variable **answer** mit „**nein**“. Ähnlich wie zuvor wird, falls die Antwort „**nein**“ ist, die unter der **elif**-Anweisung eingerückte Codezeile ausgegeben. Die letzte **else**-Anweisung ist ein Element, das sämtliche andere Arten von Antworten, die **answer** zugewiesen worden sein könnten, aufgreift. Da wir keinerlei andere Antworten als „ja“ oder „nein“ erwarten, gibt der Computer dann die eingerückte Anweisung aus, um dem Benutzer mitzuteilen, dass es sich um eine ungültige Antwort handelt.



## Schritt 7: Fortsetzung

### Versuche es mal selbst!

Schau auf deinem Arbeitsblatt für die Planung nach und lies deine vorgesehenen Antworten dafür, falls der Benutzer mit „ja“, „nein“ oder einer ungültigen Antwort reagiert.

Füge diese, das gleiche Format einhaltenden Codezeilen **nach** dem Code ein, der deine erste Frage stellt. Du musst unbedingt die **Einrückung** der print-Anweisung nach den Anweisungen **if**, **elif** und **else** berücksichtigen.

```
answer = input("Frage Nr. 1")
if (answer == "ja"):
    print ("Antwort auf ja")
elif (answer == "nein"):
    print ("Antwort auf nein")
else:
    print ("andere Antwort")
```

Aktualisiere für jede der Optionen die Antworten in den print-Anweisungen mit den Antworten, die du in deinem Planungsdokument notiert hast

Klicke auf die Schaltfläche Run \_\_, um zu sehen, ob der Computer richtig auf deine Antworten reagiert.

## Schritt 8: Mehr Fragen stellen, um Daten zu sammeln (10–15 Min.)

Da du nun eine Frage geschrieben hast, befolge die Anleitungen in den Schritten 5 und 6, um deine restlichen Fragen zu stellen. Dein Programm sollte das gleiche Schema befolgen wie hier unten dargestellt:

```
answer = input("Frage Nr. 1")
if (answer == "ja"):
    print ("Antwort auf ja")
elif (answer == "nein"):
    print ("Antwort auf nein")
else:
    print ("andere Antwort")
```

```
answer = input("Frage Nr. 2")
if (answer == "ja"):
    print ("Antwort auf ja")
elif (answer == "nein"):
    print ("Antwort auf nein")
else:
    print ("andere Antwort")
```

```
answer = input("Frage Nr. 3")
if (answer == "ja"):
    print ("Antwort auf ja")
elif (answer == "nein"):
    print ("Antwort auf nein")
else:
    print ("andere Antwort")
```

Nach dem Schreiben jeder Frage solltest du nicht vergessen, dein Programm zu **testen**, um sicherzustellen, dass es korrekt abläuft.

### Debugging-Tipps:

- Syntaxfehler: Überprüfe, ob deine Frage sich zwischen doppelten Anführungszeichen befindet, eines vor und eines nach der Frage.
- Syntaxfehler: Überprüfe, ob die Frage sich innerhalb von Klammern befindet und die Codezeile mit einer schließenden Klammer **endet**.
- Syntaxfehler: Überprüfe, ob du nach dem Namen der Variable „answer“ auch ein Gleichheitszeichen geschrieben hast.
- Syntaxfehler: Prüfe, ob die print-Anweisungen nach den Anweisungen if, elif und else **engerückt sind**.



## Schritt 9: Erweiterungen für deinen Chatbot (15–30 Min.)

Es gibt unzählige Möglichkeiten, dein Chatbot-Projekt auf eine höhere Ebene zu bringen!

- **Füge eine Frage mit anderen Antworten als „ja“ oder „nein“ hinzu. (5–8 Min.)**  
Wir können bedingte Anweisungen schreiben, um einfach alles zu prüfen! Bisher verglichen unsere bedingten Anweisungen die Antworten des Benutzers nur mit „ja“ oder „nein“, aber das lässt sich leicht ändern. Falls wir unsere möglichen Antworten auf „Wahr“ oder „Falsch“ hin ändern möchten, aktualisieren wir die Bedingung innerhalb der Klammern einfach so, dass sie `answer == "Wahr"` lautet

Früherer Code	Aktualisierter Code
<pre>answer = input("Frage") if (answer == "ja"):     print ("Antwort auf ja") elif (answer == "nein"):     print ("Antwort auf nein") else:     print ("andere Antwort")</pre>	<pre>answer = input("Frage") if (answer == "Wahr"):     print ("Antwort auf Wahr") elif (answer == "Falsch"):     print ("Antwort auf Falsch") else:     print ("andere Antwort")</pre>

Plane eine andere Frage, auf die es andere Antworten als „ja“ oder „nein“ gibt, und füge sie hinzu.

- **Füge eine Frage mit mehr als zwei möglichen Antworten hinzu. (8–10 Min.)**  
Wir haben ausschließlich Fragen mit zwei Antworten (ja oder nein) berücksichtigt, aber was wäre, wenn wir eine Frage stellen möchten, die zahlreiche Antworten haben könnte? Wenn wir eine andere Antwort erfassen möchten, müssen wir eine weitere **bedingte** Anweisung hinzufügen. Erwinnere dich daran, dass die Reihenfolge der bedingten Anweisungen `if`, `elif` und dann `else` ist. Falls wir eine andere mögliche Antwort hinzufügen wollen, fügen wir vor der `else`-Anweisung eine zusätzliche `elif`-Anweisung ein.

Da `else` sämtliche anderen möglichen Antworten abfängt, sollten wir sicherstellen, dass unsere dritte Option erfasst wird, bevor die `else`-Anweisung erreicht wird. Wenn wir beispielsweise die Antwort „vielleicht“ hinzufügen möchten, aktualisieren wir den Code entsprechend der folgenden Abbildung.

Früherer Code	Aktualisierter Code
<pre>answer = input("Frage") if (answer == "ja"):     print ("Antwort auf ja") elif (answer == "nein"):     print ("Antwort auf nein")  else:     print ("andere Antwort")</pre>	<pre>answer = input("Frage") if (answer == "ja"):     print ("Antwort auf ja") elif (answer == "nein"):     print ("Antwort auf nein") elif (answer == "vielleicht"):     print ("Antwort auf vielleicht") else:     print ("andere Antwort")</pre>

## Schritt 9: Erweiterungen (Fortsetzung)

- **Beachtung der Groß- und Kleinschreibung für Antworten entfernen. (5–8 Min.)**

Hast du es satt, immer darauf zu achten, dass du die Antworten nur in Kleinbuchstaben schreibst? Es gibt eine einfache Möglichkeit, dies zu beheben! Dafür werden wir den Befehl `lower()` verwenden, der alle Worte in Kleinbuchstaben umwandelt. Falls das Wort bereits in Kleinbuchstaben geschrieben ist, wird dieser Befehl nichts tun und das Wort bleibt, wie es ist. Falls das Wort sowohl Klein- als auch Großbuchstaben enthält, wird alles in Kleinbuchstaben umgewandelt (z. B. GWC → gwc).

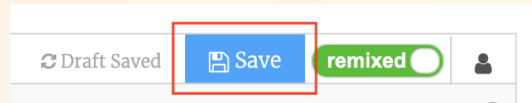
Um den Befehl `lower()` zu benutzen, können wir eine neue Codezeile hinzufügen, die der Variable `answer` die Version in Kleinbuchstaben neu zuweist, und zwar folgendermaßen:



Früherer Code	Aktualisierter Code
<pre>answer = input("Frage")  if (answer == "ja"):     print ("Antwort auf ja") elif (answer == "nein"):     print ("Antwort auf nein") else:     print ("andere Antwort")</pre>	<pre>answer = input("Frage") answer = answer.lower() if (answer == "ja"):     print ("Antwort auf ja") elif (answer == "nein"):     print ("Antwort auf nein") else:     print ("andere Antwort")</pre>

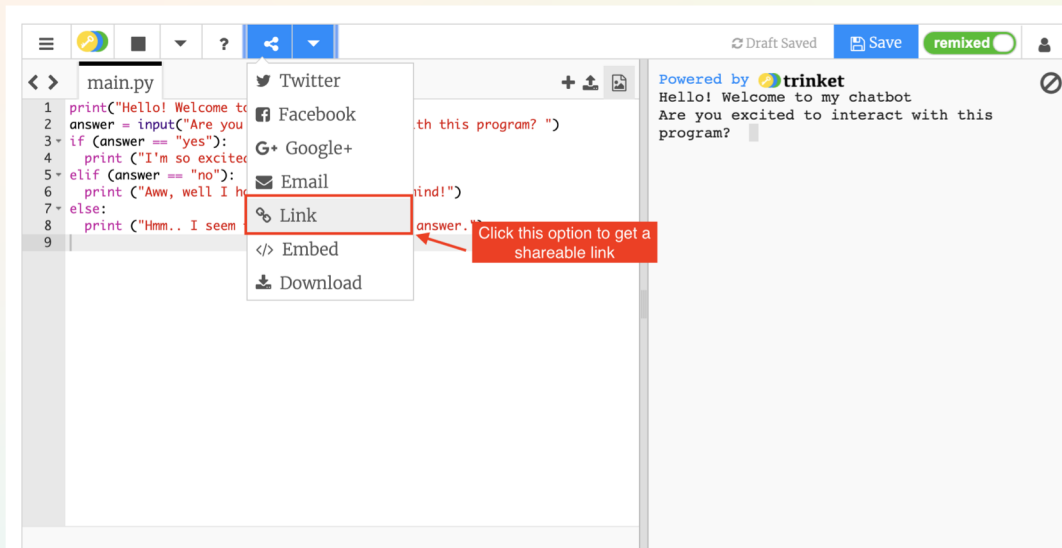
Diese neue Zeile müssen wir nach **jeder** gestellten Frage hinzufügen.

## Schritt 10: Teilt euer Projekt von Girls Who Code at Home! (5 Min.)

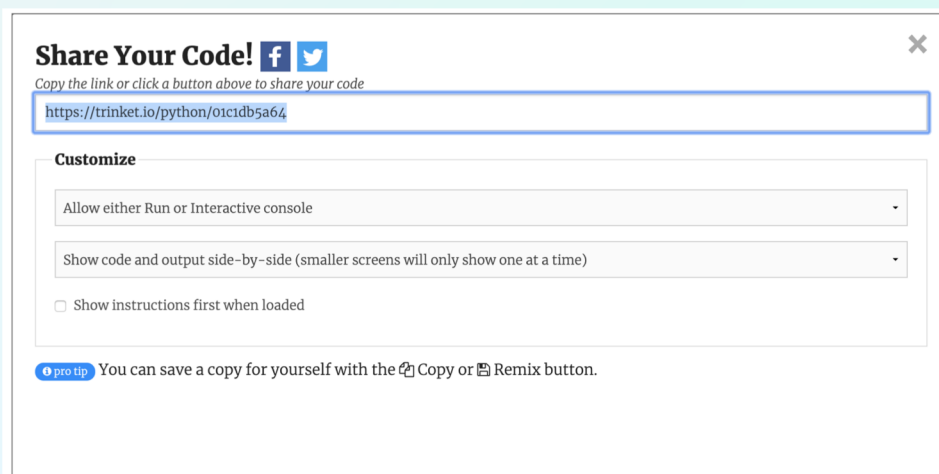
- Vergiss nicht, deine Arbeit zu speichern, indem du auf die Schaltfläche **Save** auf der rechten Seite des Trinket-Fensters klickst.



- Um dein Projekt im Freundeskreis und mit der Familie zu teilen, klicke auf das Symbol  zum **Teilen** auf der linken Seite und wähle die Option  **Link** im Ausklappmenü.



- Teile ein Foto oder Video deines funktionierenden Chatbots oder kopiere den Link und teile deinen Chatbot in sozialen Medien! Vergiss nicht, @girlswhocode zu taggen und verwende den Hashtag #codefromhome. Wir könnten dich vielleicht sogar in unserem Account vorstellen.



# Kann ich dir helfen? Arbeitsblatt für die Planung des Projekts

## Planungsübersicht für das Projekt

**Thema:** Welches Thema wird in deinem Chatbot behandelt?

**Publikum:** Wem wird dieser Chatbot nützen? Welche Personengruppe wird an deinem Chatbot interessiert sein?

**Ziel:** Was möchtest du mit deinem Chatbot erreichen? Warum ist das für dein Publikum von Interesse?

## Planung der Fragen

Wähle drei „Ja-oder-Nein-Fragen“, die du deinen Benutzern stellen wirst. Schreibe die Frage in die erste Zeile und die Antworten deines Chatbots jeweils in die Zeilen für „ja“ und „nein“. Python ist recht wählerisch bei der Annahme von Fragen, und die letzte Option „Andere Antwort“ ist als Reaktion auf Antworten gedacht, die weder „ja“ noch „nein“ sind. Du solltest eine Mitteilung hinzufügen, die den Benutzer wissen lässt, dass seine Antwort ungültig war.

Frage Nr. 1:	
Ja	
Nein	
Andere Antwort	

## Planung der Fragen (Fortsetzung)

Frage Nr. 2:	
Ja	
Nein	
Andere Antwort	

Frage Nr. 3:	
Ja	
Nein	
Andere Antwort	