



Girls Who Code en casa

¿Puedo ayudarte?
Chatbot en Python

Resumen de la actividad

En esta actividad, explorarás conceptos básicos de ciencias de la computación en Python mientras aprendes cómo ayudar a tu comunidad creando un chatbot o sistema de ayuda automatizado. Es probable que alguna vez hayas interactuado con un chatbot, tal vez sin darte cuenta. Algunos chatbots comunes son: Siri de Apple, Alexa de Amazon, Google Assistant, Lyft y, recientemente, la Organización Mundial de la Salud ha creado uno para [información sobre el COVID-19](#). Antes de que comiences a diseñar y programar tu chatbot, lee nuestro artículo destacado de “Women in Tech” sobre Erica Kochi. Cofundadora de la Unidad de Innovación del UNICEF, Erica creó un chatbot para ayudar a brindar atención y ayuda a personas en Nigeria y Ruanda.

Materiales

- [Editor Trinket](#)
- [Código inicial de Trinket](#)
- [Ejemplo de proyecto de chatbot](#)
- Opcional: Guía de planificación
- Opcional: Lápiz, lapicera o marcadores

Artículo destacado de “Women in Tech”: Erica Kochi



Erica Kochi es cofundadora y codirectora de la Unidad de Innovación de [UNICEF](#), que tiene como objetivo mejorar la salud mundial mediante innovaciones tecnológicas. Su proyecto, llamado [RapidSMS](#), utilizó teléfonos móviles y mensajes de texto (SMS) para facilitar la adquisición de datos para servicios de salud y educación.

El trabajo de Kochi junto con otros socios ayudó a desarrollar tecnologías de código abierto que registraron más de 7 millones de nacimientos en Nigeria y brindaron asistencia médica a miles de mujeres embarazadas en Ruanda. En 2013, fue nombrada una de las “100 personas más influyentes del mundo” por la revista TIME.

Ve este [video](#) para conocer más sobre Erica Kochi y el trabajo que lleva a cabo en [UNICEF](#).

Reflexión

Ser una experta informática es más que sencillamente ser buena programando. Toma unos minutos para reflexionar sobre cómo Erica y su trabajo reflejan las características que todos los verdaderos expertos informáticos deben desarrollar: valentía, resistencia, creatividad y propósito.



CORAJE

Un a gran parte de lo que hace Erica Kochi consiste en tratar de aliviar el sufrimiento de otros. ¿Cómo podría ser esto un reto en una carrera profesional?

Comparte tus respuestas con un familiar o amigo. Anímales a otros que lean sobre Erica y se unan la charla.

Primer Paso: Explora (5 minutos)

En este tutorial, crearás un chatbot personalizado en Python. Un **chatbot** es un programa para computadora que simula conversaciones con una persona real. Es una forma muy simple de inteligencia artificial. Algunos chatbots que quizá ya hayas usado son: [Starbucks Barista](#), [Siri de Apple](#), [Alexa de Amazon](#) y [Google Assistant](#).

Toma 5 minutos para explorar algunas de las características de [este chatbot](#) que creamos sobre mujeres en la tecnología. Para nuestro ejemplo, seleccionamos el siguiente tema, audiencia y objetivo:

- **Tema del proyecto:** Mujeres y tecnología
- **Audiencia:** Personas que quieren aprender sobre las mujeres que trabajan en el área de la tecnología
- **Objetivo:** Compartir chistes basados en la tecnología e información sobre las mujeres en la tecnología

Mientras exploras el ejemplo de proyecto, piensa en lo siguiente:

- ¿Cómo se presenta el chatbot?
- ¿Qué tipos de preguntas se hacen? ¿Se logra la meta con esto?
- ¿Qué características del ejemplo anterior incorporarías en tu proyecto? ¿Qué harías diferente en tu proyecto?

Segundo Paso: Genera ideas sobre las características y planifica el proyecto (10 minutos)

Ya que has tenido la oportunidad de explorar un ejemplo de proyecto, es buena idea tomarte un poco de tiempo para elaborar un plan de acción. Aprovecha este tiempo para determinar qué quieres que haga tu proyecto y cuál será el objetivo. Tal vez quieras usar el documento de planificación al final (páginas 16 a 18) para captar y organizar tus ideas.

1. Selecciona un tema, una audiencia y una meta para tu chatbot.

Piensa en qué tema quieres centrarte con el chatbot y lo que debe lograr. Puedes crear un chatbot informativo, gracioso, para fortalecer a la comunidad, o de datos y preguntas. Si te sientes sin inspiración, ve aquí algunas posibles ideas para tu proyecto:

- Un chatbot que anima a personas si se sienten aburridas y/o aisladas en casa
- Un chatbot que ofrece sugerencias para incorporar más diversión en casa
- Un chatbot que cuenta chistes
- Un chatbot que comparte datos sobre tu programa, artista, músico o pasatiempo favorito

Al seleccionar a la audiencia del chatbot, piensa en las personas a las que les emocionaría usar tu producto. Piensa en lo que les interesaría conocer y cómo captarás su atención.

Segundo Paso: Planifica tu proyecto (continuación)

2. **Selecciona tres preguntas “sí o no” que responderá tu chatbot y elabora opciones para cada respuesta posible.**

Piensa en cuales preguntas te gustaría formular y cómo se relacionan esas preguntas con el tema y el objetivo del proyecto. ¿Qué sucederá si la respuesta del usuario a tu pregunta es “sí”? ¿Qué pasará si la respuesta es “no”? ¿Qué pasará si el usuario no escribe ni “sí” ni “no”? ¿Cómo debe responder el chatbot a estos errores del usuario?

Tercer Paso: Comienza en Trinket (10 minutos)

Python es un lenguaje de programación basado en texto, lo que significa que es necesario escribir todos los comandos. En Python, los programadores utilizan muchas variables, estructuras de datos y funciones para almacenar información y ejecutar comandos. Dado que Python está basado en texto, es un poco más difícil que otros lenguajes, como Scratch, pero no imposible.

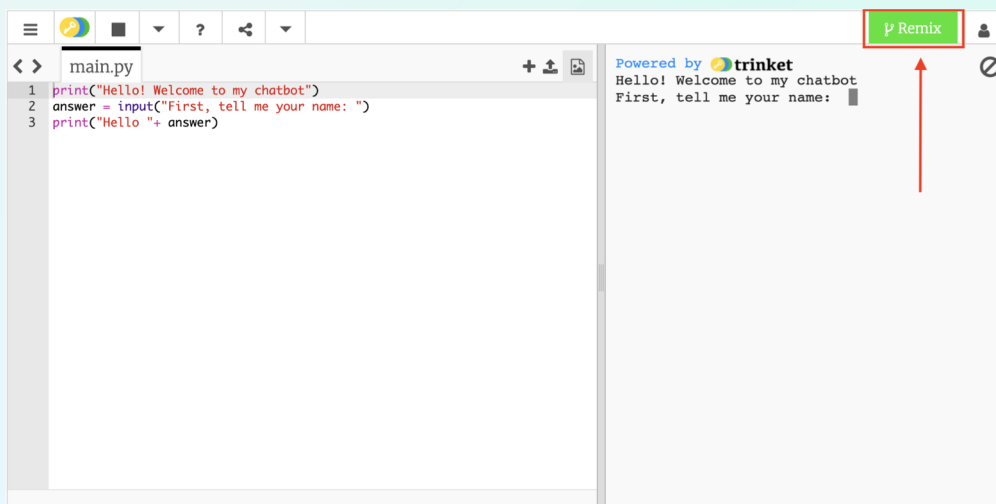
Usaremos la plataforma de codificación [Trinket](https://trinket.io) para escribir y ejecutar nuestro código de Python. Existen muchos programas que pueden usarse para escribir y ejecutar código de Python. Elegimos este porque puedes usarlo directamente en tu navegador.

1. **Regístrate o inicia sesión en [Trinket.io](https://trinket.io)**

Para guardar tu trabajo en Trinket, tendrás que crear una cuenta, si es que aún no tienes una. Sigue las instrucciones que aparecerán en el formulario de inscripción para crear una cuenta. Si eres menor de 13 años, necesitarás la dirección de correo electrónico de uno de tus padres para inscribirte.

2. **Haz un remix de este [código inicial](#) para tu chatbot.**

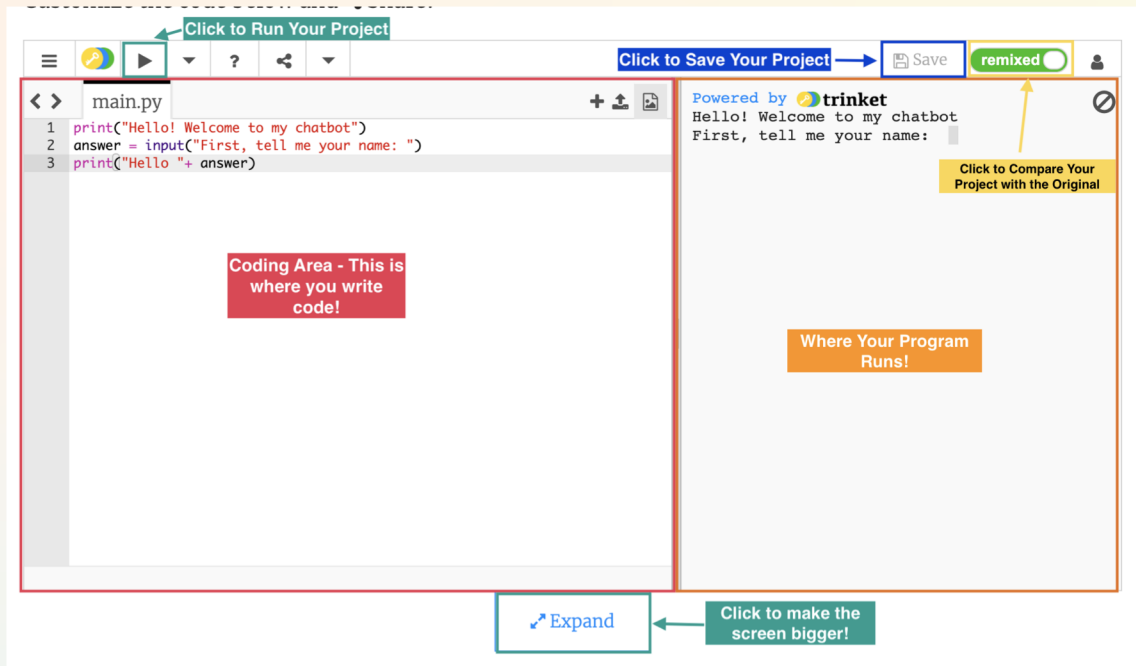
Después de navegar al código inicial, lo primero que debes hacer es un **remix**, o copia, del proyecto. Haz clic en el botón **Remix** en el lado izquierdo de la ventana de Trinket.



Tercer Paso: Comienza en Trinket (continuación)

3. Explora la interfaz de Trinket.

Si eres nueva en Trinket, tómate unos minutos para explorar su interfaz. Familiarízate con cómo guardar y ejecutar tu código.



Cuarto Paso: Explora el proyecto inicial (2 minutos)

Veamos el [archivo inicial](#). Presiona el botón Run  (Ejecutar) para ver qué hace el código.

Notarás una breve introducción en la que el programa dice:

Hello! (¡Hola!) Welcome to my chatbot. (Bienvenida a mi chatbot.)

Seguido por una pregunta:

Are you excited to interact with this program? (¿Estás emocionada de interactuar con este programa?)

Intenta escribir "yes" (sí) y ver la respuesta; luego, escribe "no" y después "Yes" (Sí).

Notarás que las tres respuestas generan una reacción distinta en el programa. Dado que Python es un lenguaje basado en texto, la ortografía y el uso de minúsculas y mayúsculas cuentan. La computadora interpreta de manera diferente las respuestas "yes" y "Yes".

Los aspectos más importantes de un chatbot son la capacidad de (1) solicitarle al usuario una respuesta a una pregunta y (2) hablar con el usuario mediante la función print para las respuestas. Primero, aprenderemos a hablar con el usuario a través de código, y luego veremos cómo hacer preguntas y responder.

Quinto Paso: Añade una introducción (4 minutos)

Para que la computadora “hable” con el usuario, escribimos declaraciones `print` (mostrar) en el código, y la declaración se muestra en el lado izquierdo de la ventana de Trinket.

Veamos la primera línea de código del [archivo inicial](#). Presiona el botón Run `__` (Ejecutar) para ver qué hace el código.

Notarás que la primera línea de código contiene `print("Hello! Welcome to my chatbot")` (`¡Hola! Bienvenida a mi chatbot`) y que la primera línea en la pantalla de **resultados** es `Hello! (¡Hola!). Welcome to my chatbot. (Bienvenida a mi chatbot.)`

Veamos los símbolos y términos clave para usar una declaración `print`:

```
print("ejemplo de texto")
```

- `print`: Esta palabra clave le indica a la computadora que debe mostrar algo en la pantalla de resultados.
- `()`: Los paréntesis le indican a la computadora que debe mostrar lo que está entre los paréntesis.
- `" "`: Las comillas dobles le indican a la computadora que todo lo que está entre ellas es palabras.
- `ejemplo de texto`: Podemos incluir cualquier palabras que deseemos entre las comillas y se mostrarán exactamente igual en la pantalla de **resultados**.

¡Inténtalo!

Añade una línea de código para `mostrar` una breve introducción de tu chatbot. Tal vez quieras incluir el tema y el objetivo en esta descripción.

Presiona el botón Run `__` (Ejecutar) para ver si la computadora muestra tu introducción.

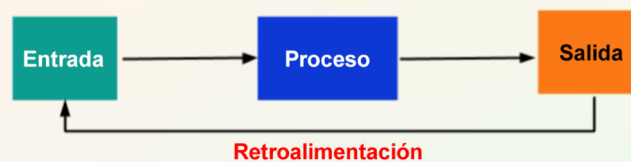
Consejos de depuración:

- Error de sintaxis: Comprueba que haya comillas dobles tanto al principio como al final de la pregunta.
- Error de sintaxis: Comprueba que la pregunta esté entre paréntesis y que la línea de código **se acaba** con un paréntesis de cierre.

Sexto Paso: Haz tu primera pregunta (5 minutos)

Para hacer tu primera pregunta, hablaremos de dos términos importantes en las ciencias de la computación: **entrada** y **salida**.

Una **entrada** es información que se le proporciona a la computadora. Podría ser presionar una tecla, conectar un dispositivo (por ejemplo, un mouse) o, en nuestro caso, escribir una respuesta. Una **salida** es información enviada desde la computadora a otro proceso. La respuesta del usuario ("sí" o "no") en la **entrada**, mientras que la respuesta de la computadora es la **salida** del proceso de formular una pregunta.



Para obtener una entrada para una pregunta en Python, usamos el comando `input()`.

Notarás que la segunda línea de código dice, `answer = input("Are you excited to interact with this program? ")` (¿Estás emocionada de interactuar con este programa?) y que la primera línea en la pantalla de **resultados** es `Are you excited to interact with this program?` (¿Estás emocionada de interactuar con este programa?).

Veamos los símbolos y términos clave para usar un enunciado `input`:

```
answer = input("ejemplo de pregunta ")
```

- **answer**: Esta es una **variable** que guarda la respuesta del usuario. En particular, a esta variable se le ha asignado el nombre "answer" (respuesta).
- **=** : el signo de igualdad muestra la asignación o reasignación de un valor a la **variable** "answer".
- **input**: Esta palabra clave le indica a la computadora que debe hacerle una pregunta al usuario.
- **()**: Los paréntesis le indican a la computadora que debe mostrar el texto entre los paréntesis.
- **" "**: Las comillas dobles le indican a la computadora que todo lo que está entre ellas es palabras y parte del mensaje de entrada.
- **ejemplo de pregunta**: Podemos incluir cualquier palabras que deseemos entre las comillas y esta pregunta se mostrará exactamente igual en la pantalla de **resultados**.

Una **variable** es un término de las ciencias de computación utilizado para guardar información (datos) en un programa de computación. Se asignan nombres a las variables, para que sea fácil hacer referencia a ellas y cambiarlas en un programa.

Sexto Paso: Haz tu primera pregunta (continuación)

¡Inténtalo!

Consulta tu hoja de trabajo de planificación y haz tu primera pregunta al usuario. Añade una nueva línea de código con el siguiente formato.

```
answer = input("pregunta1")
```

Reemplaza `pregunta1` por tu pregunta real.

Presiona el botón Run  (Ejecutar) para ver si la computadora hace la pregunta.

Consejos de depuración:

- Error de sintaxis: Comprueba que haya comillas dobles tanto al principio como al final de la pregunta.
- Error de sintaxis: Comprueba que la pregunta esté entre paréntesis y que la línea de código **se acaba** con un paréntesis de cierre.
- Error de sintaxis: Comprueba que hayas escrito un signo de igualdad después del nombre de variable `answer`.

Séptimo Paso: Responde a una pregunta (10 minutos)

Tal vez hayas notado que la computadora hace una pregunta, pero no hace nada más para responder después de que escribes una respuesta. Esto es porque no le indicamos a la computadora que hiciera algo con los datos!

Al escribir el código: `answer = input("pregunta1")`, la respuesta del usuario se **guarda** en la variable `respuesta`. Es así como determinaremos si el usuario respondió “yes” (sí) o “no”.

Primero, debemos usar declaraciones **condicionales** para comparar las opciones posibles (“yes” o “no”). Una **declaración condicional** revisa si se cumple un conjunto de reglas (o un enunciado) y decide qué acciones realizar si las reglas o el enunciado son verdaderos o falsos. Hay tres opciones posibles para nuestra pregunta: “yes” (sí), “no” o cualquier otra respuesta.

Al usar condiciones en Python, debemos utilizar las palabras clave `if`, `elif` o `else`. Veamos cómo se usan las declaraciones condicionales en el chatbot para determinar la manera correcta de responder a cada contestación posible.

Séptimo Paso: Responde a una pregunta (continuación)

```
answer = input("Are you excited to interact with this program? ")
[¿Estás emocionada de interactuar con este programa?]
if (answer == "yes"):
    print ("I'm so excited too!") [Yo también estoy muy emocionada]
elif (answer == "no"):
    print ("Aww, well I hope I can change your mind!") [¡Qué lástima!
Espero hacerte cambiar de opinión.]
else:
    print ("Hmm.. I seem to not comprehend your answer.") [No puedo entender
tu respuesta]
```

- **if**: Palabra clave para indicar una declaración "if" (si).
- **elif**: Palabra clave para indicar una declaración "else-if" (de lo contrario si). Este es una declaración opcional, pero debe aparecer **después** de una declaración "if".
- **else**: Palabra clave para indicar una declaración "else" (de lo contrario). Este es una declaración opcional, pero debe aparecer **después** de una declaración "if" y "elif".
- **()**: Los paréntesis son opcionales. Se añaden alrededor de la condición para mantener el código organizado y más fácil de leer.
- **answer**: Esta es una **variable** que guarda la respuesta del usuario. En particular, a esta variable se le ha asignado el nombre "answer" (respuesta).
- **==**: Los signos de igualdad son un comparador. Se usan para comparar la variable "answer" con otro valor, en nuestro caso, "yes" o "no".
- **"yes" / "no"**: Las comillas dobles se usan para indicarle a la computadora que debe leer el valor dentro de ellas como texto. Como queremos comparar la respuesta con estas palabras, usamos comillas dobles para encerrar "yes" y "no".
- **:** Los dos puntos le indican al programa dónde termina la declaración condicional.
- **Sangría**: Todas las líneas de código que deben ejecutarse si se cumple una condición deben tener sangría después la declaración "if". Esto le indica a la computadora cuales líneas de código debe ejecutar. Ve el ejemplo anterior, donde el comando `print` tiene sangría.

Recuerda que usamos la variable **answer** para *guardar* la respuesta del usuario a nuestra pregunta. La primera declaración **if** compara si la respuesta en la variable es "yes" (sí). Si la respuesta es "yes", se muestra la línea de código con sangría debajo la declaración **if**. Si **answer** no es "yes", avanzamos la siguiente declaración condicional, **elif**. Esta declaración compara la respuesta con "no". De manera similar, si la **respuesta** es "no", se muestra la línea de código con sangría debajo la declaración **elif**. La última declaración **else** captura todas las otras respuestas que pudiera contener **answer**. Como no esperamos respuestas distintas de "yes" y "no", la computadora muestra la declaración con sangría para informarle que su respuesta no fue válida.

Septimo Paso: Continuación

¡Inténtalo!

Revisa tu hoja de trabajo de planificación y tus respuestas para cuando el usuario conteste “yes” (sí), “no” o con una respuesta no válida.

Agrega estas líneas de código con el mismo formato **después** del código donde formulas la primera pregunta. Asegúrate de **sangrar** las declaración “print” en las declaraciones **if**, **elif** y **else**.

```
answer = input("pregunta1 ")
if (answer == "yes"):
    print ("yes response") [respuesta sí]
elif (answer == "no"):
    print ("no response") [respuesta no]
else:
    print ("other response") [otra respuesta]
```

Actualiza las respuestas de las declaraciones “print” de cada opción con las respuestas que escribiste en tu documento de planificación.

Presiona el botón Run  (Ejecutar) para ver si la computadora responde según tus contestaciones.

Octavo Paso: Hacer más preguntas para recopilar datos (10 a 15 minutos)

Ya que has escrito una pregunta, sigue las instrucciones de los pasos 5 y 6 para formular las demás. Tu programa debe seguir el siguiente formato:

```
answer = input("pregunta1")
if (answer == "yes"):
    print ("yes response") [respuesta sí]
elif (answer == "no"):
    print ("no response") [respuesta no]
else:
    print ("other response") [otra respuesta]

answer = input("pregunta2")
if (answer == "yes"):
    print ("yes response") [respuesta sí]
elif (answer == "no"):
    print ("no response") [respuesta no]
else:
    print ("other response") [otra respuesta]

answer = input("pregunta3")
if (answer == "yes"):
    print ("yes response") [respuesta sí]
elif (answer == "no"):
    print ("no response") [respuesta no]
else:
    print ("other response") [otra respuesta]
```

Cuando termines de escribir una pregunta, no olvides **probar** el programa para comprobar que funciona correctamente.

Consejos de depuración:

- Error de sintaxis: Comprueba que haya comillas dobles tanto al principio como al final de la pregunta.
- Error de sintaxis: Comprueba que la pregunta esté entre paréntesis y que la línea de código **se acaba** con un paréntesis de cierre.
- Error de sintaxis: Comprueba que hayas escrito un signo de igualdad después del nombre de variable answer.
- Error de sintaxis: Comprueba que las declaraciones "print" después de las declaraciones "if", "elif" y "else" tengan **sangría**.

Noveno Paso: Extensiones para tu chatbot (15 a 30 minutos)

Hay muchas maneras de llevar tu proyecto de chatbot a otro nivel.

- **Añade una pregunta con respuestas distintas de “yes” (sí) y “no”. (5 a 8 minutos)**

Podemos escribir declaraciones condicionales para verificar lo que sea. En este momento, nuestras declaraciones condicionales solo comparan la respuesta del usuario con “yes” y “no”, pero esto es fácil de cambiar. Si quisiéramos cambiar nuestras posibles respuestas a “True” (Verdadero) o “False” (Falso), bastaría actualizar la condición entre paréntesis para que diga `answer == “True”`.

Código anterior	Código actualizado
<pre>answer = input("question") [pregunta] if (answer == "yes"): print ("yes response") [respuesta sí] elif (answer == "no"): print ("no response") [respuesta no] else: print ("other response") [otra respuesta]</pre>	<pre>answer = input("question") [pregunta] if (answer == "True"): [Verdadero] print ("true response") [respuesta verdadero] elif (answer == "False"): [Falso] print ("false response") [respuesta falso] else: print ("other response") [otra respuesta]</pre>

Planifica y añade otra pregunta que tenga respuestas distintas de “yes” y “no”.

- **Añade una pregunta con más de dos posibles respuestas. (8 a 10 minutos)**

Solamente hemos considerado preguntas con dos respuestas (sí y no), pero ¿qué pasaría si quisiéramos hacer una pregunta con múltiples respuestas? Si queremos capturar otra respuesta, debemos añadir otra declaración **condicional**. Recuerda que el orden de las declaraciones condicionales es “if”, “elif” y luego “else”. Si queremos añadir otra respuesta posible, agregamos un enunciado “elif” antes la declaración “else”.

Como “else” captura todas las demás respuestas posibles, queremos asegurarnos de capturar nuestra tercera opción antes de llegar la declaración “else”. Por ejemplo, si quisiéramos añadir la respuesta “Maybe” (Tal vez), actualizaríamos el código como sigue.

Código anterior	Código actualizado
<pre>answer = input("question") [pregunta] if (answer == "yes"): print ("yes response") [respuesta sí] elif (answer == "no"): print ("no response") [respuesta no] else: print ("other response") [otra respuesta]</pre>	<pre>answer = input("question") [pregunta] if (answer == "yes"): print ("yes response") [respuesta sí] elif (answer == "no"): print ("no response") [respuesta no] elif (answer == "maybe"): [tal vez] print ("maybe response") [respuesta tal vez] else: print ("other response") [otra respuesta]</pre>

Noveno Paso: Extensiones (continuación)

- **Eliminar la diferencia entre mayúsculas y minúsculas en las respuestas. (5 a 8 minutos)**

¿Estás cansada de asegurarte de escribir las respuestas solo en minúsculas? Hay una manera fácil de resolverlo. Para hacerlo, usaremos el comando `lower()`, que convierte una palabra a minúsculas. Si la palabra ya está en minúsculas, este comando no hace nada y la palabra permanece sin cambio. Si la palabra tiene una combinación de minúsculas y mayúsculas, todas las letras se convertirán a minúsculas (por ejemplo, GWC → gwc).

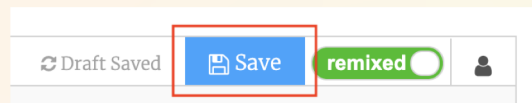
Para usar el comando `lower()`, añadimos una nueva línea de código que reasigna la respuesta variable a la versión en minúsculas, como sigue:

Código anterior	Código actualizado
<pre>answer = input("question") [pregunta] if (answer == "yes"): print ("yes response") [respuesta sí] elif (answer == "no"): print ("no response") [respuesta no] else: print ("other response") [otra respuesta]</pre>	<pre>answer = input("question") [pregunta] answer = answer.lower() if (answer == "yes"): print ("yes response") [respuesta sí] elif (answer == "no"): print ("no response") [respuesta no] else: print ("other response") [otra respuesta]</pre>

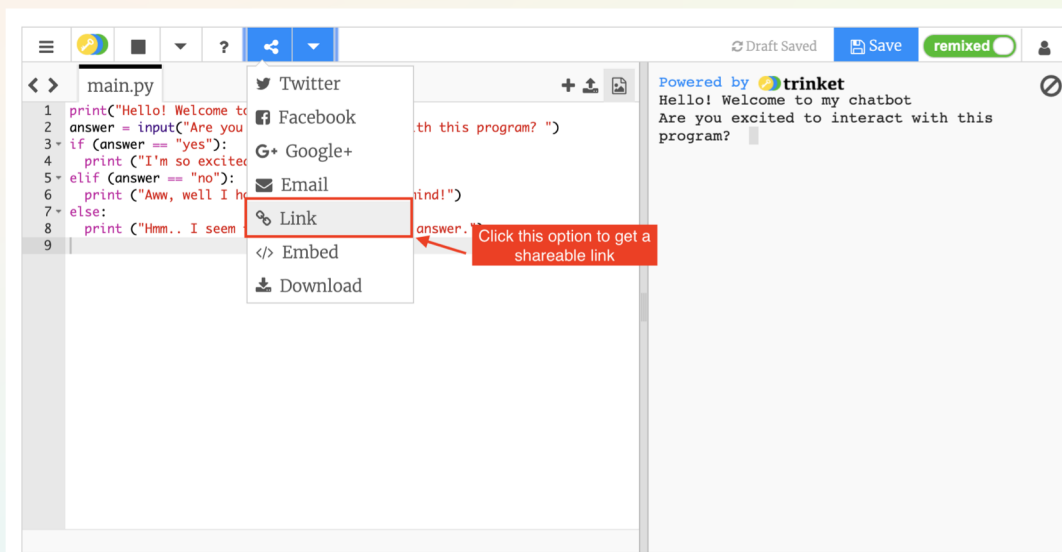
Debemos añadir esta nueva línea después de formular **cada** pregunta.

Décimo Paso: Comparte tu proyecto de Girls Who Code en casa (5 minutos)

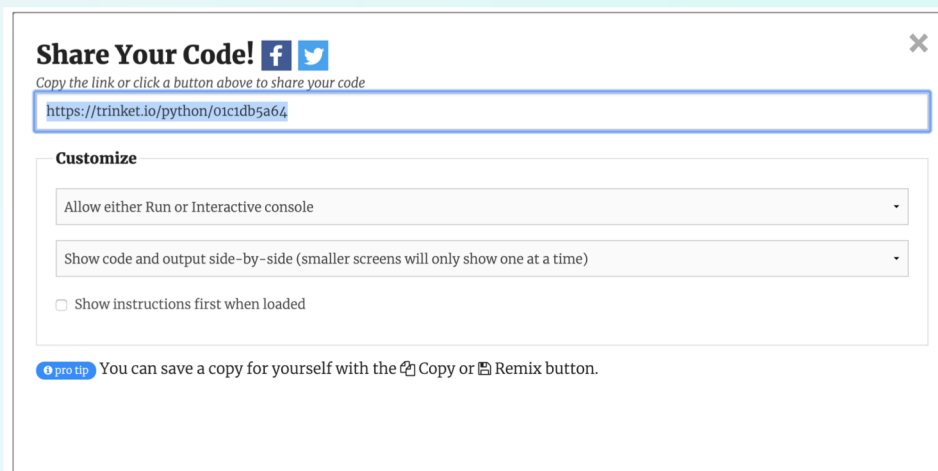
- No olvides guardar tu trabajo haciendo clic en el botón **Save** (Guardar) en el lado derecho de la ventana de Trinket.



- Para compartir tu proyecto con amigos y familiares, haz clic en el icono **Share** (Compartir) de la izquierda y selecciona la opción **Link** (Enlace) en el menú desplegable.



- Comparte una foto o un video de tu chatbot en funcionamiento o copia el enlace y comparte tu chatbot en las redes sociales. No olvides etiquetar @girlswhocode y usa el hashtag #codefromhome, y puede que te destaquemos en nuestra cuenta.



¿Puedo ayudarte? Hoja de trabajo de planificación de proyecto

Planificación general del proyecto

Tema: ¿Qué tema abarcará tu chatbot?

Audiencia: ¿A quiénes ayudará este chatbot? ¿Qué grupo de personas estarían interesadas en tu producto?

Objetivo: ¿Qué quieres que logre tu chatbot? ¿Por qué le interesa esto a tu audiencia?

Planificación de preguntas

Selecciona tres preguntas con respuesta “sí” o “no” que les harás a los usuarios. Escribe la pregunta en el primer renglón y la respuesta del chatbot en cada renglón para “yes” o “no”. Python es muy quisquilloso al aceptar respuestas, así que la última opción, “other responses” (otras respuestas) es para considerar respuestas distintas de “yes” y “no”. Si quieres, podrías añadir un mensaje para informarle al usuario que su respuesta no fue válida.

Pregunta 1:	
Sí	
No	
Otra respuesta	

Planificación de preguntas (continuación)

Pregunta 2:	
Sí	
No	
Otra respuesta	

Pregunta 3:	
Sí	
No	
Otra respuesta	