



Girls Who Code At Home

Can I Help You?
Chatbot in Python

Activity Overview

In this activity you will be exploring basic computer science concepts in Python while learning how to help your community by creating a chatbot, or an automated help system! You have probably interacted with a chatbot before, maybe without even realizing it. Some common chatbots include: Apple's Siri, Amazon's Alexa, Google Assistant, Lyft, and most recently the World Health Organization has built one for [COVID-19 facts](#). Before you start designing and coding your chatbot, check out our featured Women in Tech Spotlight, Erica Kochi. As the Co-founder of UNICEF's Innovation Unit, Erica built a chatbot to help provide care and assistance for people in Nigeria and Rwanda.

Materials

- [Trinket Editor](#)
- [Starter Trinket Code](#)
- [Example Chatbot Project](#)
- Optional: Planning Guide
- Optional: Pen/Pencil/Markers

Women in Tech Spotlight: Erica Kochi



Erica Kochi co-founded and co-leads [UNICEF's](#) Innovation Unit, which aims to improve global health using technological innovations. Her project called [RapidSMS](#) utilized mobile phones and SMS messages to help facilitate data collection for health and education services.

Kochi's work with partners helped develop open source technologies that registered over 7 million births in Nigeria and provided care to thousands of pregnant women across Rwanda. In 2013, she was named in TIME 100's "World's Most Influential People"!

Watch this [video](#) to learn more about Erica Kochi and some of the work she does at [UNICEF](#).

Reflect

Being a computer scientist is more than just being great at coding. Take some time to reflect on how Erica and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



BRAVERY

A big part of what Erica Kochi does is try to help the suffering of others. How might that be challenging in a career?

Share your responses with a family member or friend. Encourage others to read more about Erica to join in the discussion!

Step 1: Explore (5 min)

In this tutorial you will be making a custom chatbot in Python! A **chatbot** is a computer program that simulates conversations with a real person. It is a very simple form of Artificial Intelligence, or AI. Some chatbots that you may already use include: [Starbucks Barista](#), [Apple's Siri](#), [Amazon's Alexa](#), and [Google Assistant](#).

Take 5 minutes to explore some of the features of [this chatbot](#) we created about women in technology. For our example we selected the following theme, audience, and goal:

- **Project Theme:** Women/Technology
- **Audience:** People who want to learn about women working in technology
- **Goal:** Share technology based jokes and provide information about Women in Technology

As you explore the example project think about:

- How is the chatbot introduced?
- What types of questions are being asked? Does this accomplish the goal?
- What features from the example would you want to incorporate into your own project?
What would you want to do differently in your own project?

Step 2: Brainstorm Features and Plan Your Project (10 min)

Now that you have gotten the chance to explore a sample project, it's a good idea to next take some time to make a game plan first. Use this time to figure out what you want your project to do and what the goal is. You may want to use the planning document (on pages 16-18) to help you capture and organize your ideas.

1. Choose a Theme, Audience, and Goal for your Chatbot.

Think about what topic you want to focus on with your chatbot and what it should accomplish. You can make an informative, humourous, community-building, or a fact/quiz chatbot. If you are feeling stuck, here are some possible project ideas:

- A chatbot that provides words of encouragement to others if they are feeling bored and/or isolated at home
- A chatbot that gives suggestions on how to incorporate more fun at home
- A chatbot that tells funny jokes
- A chatbot that shares facts about your favorite show, artist, musician, or hobby

When choosing the audience of your chatbot think who is the target audience that would be excited to use your product. Consider what they are interested in knowing and how you will capture their attention.

Step 2: Plan Your Project Continued

2. Choose three “yes or no” questions for your chatbot to ask and draft out responses for each possible answer.

Think about which questions you want to ask and how those questions connect to your theme and goal of the project. What happens if the user’s response is “yes” to your question? What if the response is “no”? What if the user doesn’t type either yes or no? How should your chatbot respond to these user errors?

Step 3: Get Started on Trinket (10 min)

Python is a text-based programming language, which means that all commands will need to be typed! In Python programmers use many variables, data structures, and functions to help store information and do commands! Since Python is text-based this makes it a little more difficult than other languages, like Scratch, but not impossible!

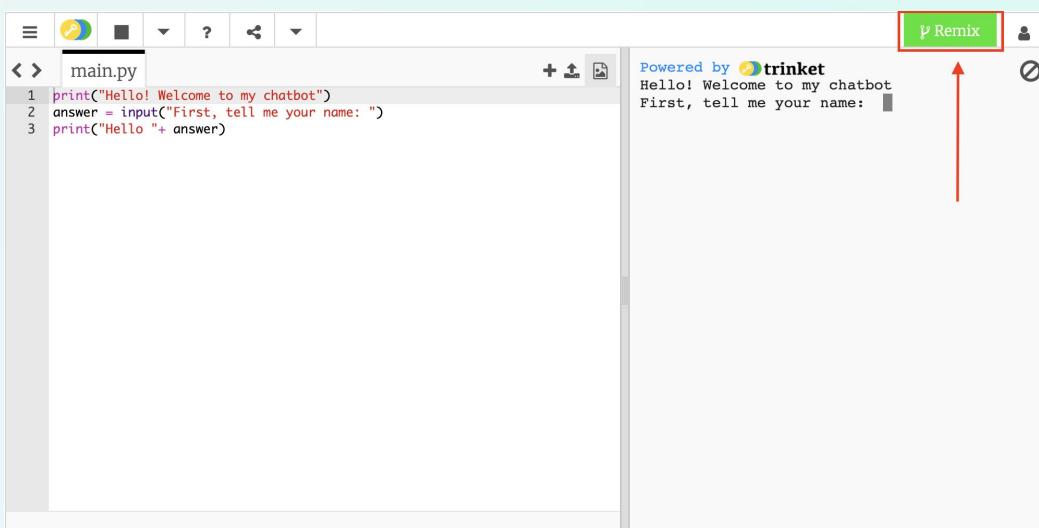
We'll be using the [Trinket](#) coding platform to write and run our Python code. There are many different programs that can be used for writing and running Python code. We chose this one because you can use it directly in your browser!

1. **Sign Up or Log in into [Trinket.io](#)**

In order to save your work on Trinket you will need to create an account if you don't have one already. Follow the instructions on the sign up form to create an account. If you are under 13 you'll need your parent's email address to sign up.

2. **Remix this [starter code](#) for your chatbot.**

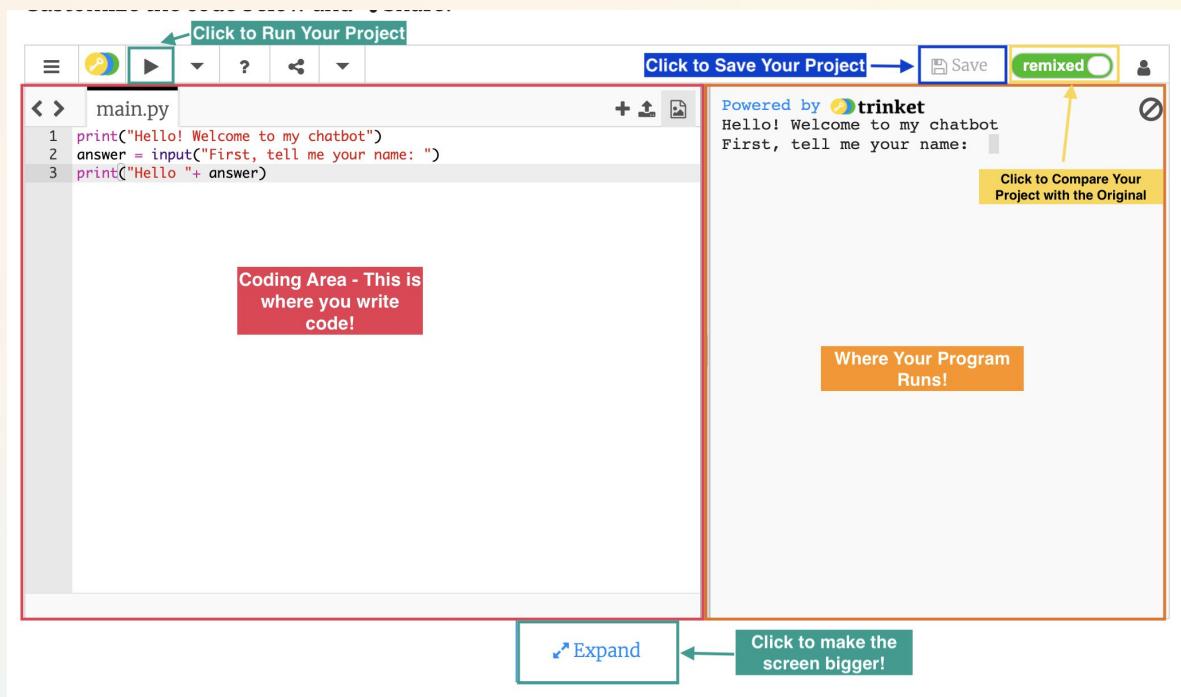
After navigating to the starter code, you should first **remix**, or save a copy, of the project. Click on the **Remix** Button on the left side of the Trinket Window.



Step 3: Get Started on Trinket Continued

3. Explore the Trinket interface.

If you are new to Trinket, take a few minutes to explore the Trinket interface. Get familiar with how to save and run your code.



Step 4: Explore the Starter Project (2 min)

Let's take a look at the [starter file](#). Press the Run Button to see what this code does.

You'll notice that there is a short introduction where the program says:

Hello! Welcome to my chatbot.

It is then followed by a question:

Are you excited to interact with this program?

Try typing in “**yes**” and seeing the response, then type “**no**” and then type “**Yes**”.

You will notice that all three of these answers have a different response from the program. Since Python is a text-based language both spelling and case of the letters count! The answer “**yes**” and “**Yes**” are read differently to the computer.

The most important aspects of a chatbot is the ability to (1) ask the user for an answer to a question and (2) talk to the user by printing out responses back. We will first learn how to talk to the user through code then discuss how to ask questions and respond.

Step 5: Add an Introduction (4 mins)

In order for the computer to “talk” to the user we write `print` statements in code and it then shows the statement on the left side of the Trinket window.

Let’s take a look at the first line of code in the [starter file](#). Press the Run  Button to see what this code does.

Notice that the first line of code reads `print("Hello! Welcome to my chatbot")` and the first line in the **results** screen is `Hello! Welcome to my chatbot`.

Let’s look at the key symbols and terms for using a `print` statement:

```
print("sample text")
```

- `print`: This keyword lets the computer know to print something to the results screen
- `()`: Parentheses lets the computer know to print whatever is inside the parentheses.
- `" "`: The double-quotation marks lets the computer know that everything inside are words.
- `sample text`: We can include any words inside of the quotation marks and this will be printed exactly as is to the **results** screen.

Try It Yourself!

Add a line of code to `print` out a short introduction of your chatbot. You might want to include the theme and goal in this description.

Press the Run  Button to see if the computer prints out your introduction.

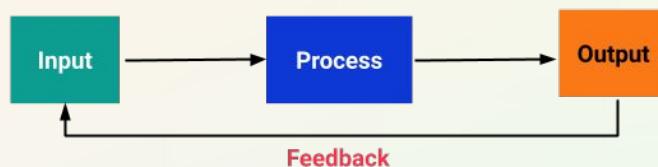
Debugging Tips:

- Syntax Error: Check that you have double-quotation marks around your question both in the beginning and end of the question.
- Syntax Error: Check that you have parentheses around the question and that the line of code **ends** with a closing parenthesis.

Step 6: Ask Your First Question (5 mins)

In order to ask your first question, let's discuss two important computer science terms: **input** and **output**.

Input is when some sort of information is given to the computer. Sometimes this could be the pressing down of a key, plugging in a device like a mouse, or in our case, the typing of a response. **Output** is when information is given from the computer to another process. The response from the user (either "yes" or "no") is the **input** while the computer's response is the **output** of the process of asking a question.



In order to get an input to a question in Python we use the command `input()`.

Notice that the second line of code reads `answer = input("Are you excited to interact with this program? ")` and the first line in the **Results** screen is `Are you excited to interact with this program?`

Let's look at the key symbols and terms for using a `input` statement:

```
answer = input("sample question ")
```

- **answer**: This is a **variable** that stores the user's response. In particular, this variable is named "answer".
- **=**: The equals sign shows assignment or reassignment of a value to the **variable** answer.
- **input**: This keyword lets the computer know to ask a question to the user
- **()**: Parentheses let the computer know to print the text inside the parentheses.
- **" "**: The double-quotation marks lets the computer know that everything inside are words and part of the input message.
- **sample question**: We can include any words inside of the quotation marks and this question will be printed exactly as is to the **results** screen.

A **variable** is a computer science term used to store information (data) in a computer program. Variables are given a name so that they can be easily referenced and changed in a program.

Step 6: Ask Your First Question Continued

Try It Yourself!

Look back at your planning worksheet and ask the user your first question.

Add a new line of code that follows the same format as below.

```
answer = input("question#1")
```

Replace **question#1** with your actual question.

Press the Run  Button to see if the computer asks your question.

Debugging Tips:

- Syntax Error: Check that you have double-quotation marks around your question both in the beginning and end of the question.
- Syntax Error: Check that you have parentheses around the question and that the line of code **ends** with a closing parenthesis.
- Syntax Error: Check that you have an equal sign after writing the variable name answer.

Step 7: Respond to an Answer (10 mins)

You may have noticed that the computer asks a question but does not do anything else to respond once you have typed in an answer. That is because we didn't tell the computer to do anything with that data!

When typing in the code: `answer = input("question#1")` the user's response is **stored** in the variable **answer**. This is how we will determine if the user answered "yes" or "no".

First, we must use **conditional** statements to compare the possible choices ("yes" or "no"). A **conditional** statement checks if a set of rules (or statement) are met and then decides which actions are performed if the rules or statement is true or false. There are three possible choices for our question either "yes", "no", or any other response.

When we use conditions in Python we must use the keywords **if**, **elif**, and/or **else**. Let's take a look at how conditionals are used in the chatbot to determine the appropriate response to each possible answer.

Step 7: Respond to an Answer Continued

```
answer = input("Are you excited to interact with this program? ")
if (answer == "yes"):
    print ("I'm so excited too!")
elif (answer == "no"):
    print ("Aww, well I hope I can change your mind!")
else:
    print ("Hmm.. I seem to not comprehend your answer.")
```

- **if**: Keyword to indicate an if statement
- **elif**: Keyword to indicate an else-if statement. This is an optional statement but must come **after** an if statement.
- **else**: Keyword to indicate an else statement. This is an optional statement but must come **after** an if and elif statement.
- (): Parentheses are optional. They are added around the condition to help keep our code organized and easier to read.
- **answer**: This is a **variable** that stores the user's response. In particular, this variable is named answer.
- == : The equals sign is a comparator. This is used to compare the variable answer to another value, in our case either "yes" or "no"
- "yes" / "no": The double-quotation marks are used to tell the computer to read the value inside as text. Since we want to compare the answer to these word responses we use double quotation marks around yes and no.
- : The colon lets the program know the end of the condition statement.
- **Indent**: All lines of code that should be executed if a condition is met should be indented after the if statement. This lets the computer know which lines of code should be run. See example above where the print command is indented.

Recall that we use the **answer** variable to store the user's answer to our question. The first **if** statement first compares if the variable answer is "yes". If the answer is "yes", then the indented line of code below the **if** statement is printed. If **answer** is not "yes", then we go to the next conditional statement, the **elif**. This then compares the answer to "no". Similarly, if the **answer** is "no", then the indented line of code below the **elif** statement is printed. The last **else** statement is a catch all for all other types of responses that **answer** could be. Since we are not expecting any other response other than yes or no, the computer then prints the indented statement to let the user know that it was an invalid answer.

Step 7: Continued

Try it Yourself!

Look back at your planning worksheet and check your responses for if the user answers "yes", "no", or an invalid response.

Add these lines of code that follow the same format **after** the code that asks your first question. Be sure to **indent** the print statements after the **if**, **elif**, and **else** statements.

```
answer = input("question#1 ")
if (answer == "yes"):
    print ("yes response")
elif (answer == "no"):
    print ("no response")
else:
    print ("other response")
```

Update the responses in the print statements for each option with the responses you wrote in your planning document

Press the Run ▶ Button to see if the computer responds accordingly to your answers.

Step 8: Asking More Questions to Gather Data (10-15 mins)

Now that you have written one question, follow the instructions in steps 5 and 6 to ask your remaining questions. Your program should follow the same format as below:

```
answer = input("question#1")
if (answer == "yes"):
    print ("yes response")
elif (answer == "no"):
    print ("no response")
else:
    print ("other response")

answer = input("question#2")
if (answer == "yes"):
    print ("yes response")
elif (answer == "no"):
    print ("no response")
else:
    print ("other response")

answer = input("question#3")
if (answer == "yes"):
    print ("yes response")
elif (answer == "no"):
    print ("no response")
else:
    print ("other response")
```

After you finish writing one question don't forget to **Test** your program to make sure it runs correctly.

Debugging Tips:

- Syntax Error: Check that you have double-quotation marks around your question both in the beginning and end of the question.
- Syntax Error: Check that you have parentheses around the question and that the line of code **ends** with a closing parenthesis.
- Syntax Error: Check that you have an equal sign after writing the variable name answer.
- Syntax Error: Check that print statements after the if, elif, and else statements are **indented**.

Step 9: Extensions for Your Chatbot (15-30 mins)

There are so many ways to take your chatbot project to the next level!

- **Add a question with responses other than "yes" or "no". (5-8 mins)**

We can write conditional statements to check for anything! Right now our conditional statements only compare the user's answer to "yes" or "no" but this can be easily changed. If we wanted to change our possible answers to "True" or "False" we simply update the condition inside the parentheses to read answer == "True"

Previous Code	Updated Code
<pre>answer = input("question") if (answer == "yes"): print ("yes response") elif (answer == "no"): print ("no response") else: print ("other response")</pre>	<pre>answer = input("question") if (answer == "True"): print ("true response") elif (answer == "False"): print ("false response") else: print ("other response")</pre>

Plan and add another question that has different answers other than "yes" or "no".

- **Add a question with more than two possible responses. (8-10 mins)**

We have only considered questions with only two responses (yes or no) but what if we wanted to ask a question that might have multiple responses? If we want to capture another response we need to add another **conditional** statement. Recall that the order of the conditional statements was if, elif, then else. If we want to add another possible response we add an additional elif statement before the else statement.

Since else is a catch all for all other possible answers, we want to make sure to capture our third option before reaching the else statement. For example, if we wanted to add the response "Maybe" we will update the code as illustrated below.

Previous Code	Updated Code
<pre>answer = input("question") if (answer == "yes"): print ("yes response") elif (answer == "no"): print ("no response") else: print ("other response")</pre>	<pre>answer = input("question") if (answer == "yes"): print ("yes response") elif (answer == "no"): print ("no response") elif (answer == "maybe"): print ("maybe response") else: print ("other response")</pre>

Step 9: Extensions Continued

- Remove case sensitivity for responses. (5-8 mins)

Tired of having to make sure you type answers in all lowercase? There is an easy way to fix this! For this we will use the command `lower()` which converts a word to all lowercase. If the word is already lowercase then this command does nothing and keeps the word as is. If the word has mixed cases then all letters will be converted to lowercase (ex. GWC → gwc)

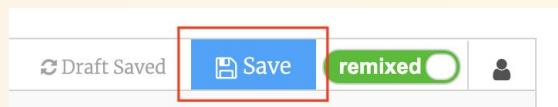
To use the command `lower()` we can add a new line of code that reassigns the variable `answer` to the lowercase version like so:

Previous Code	Updated Code
<pre>answer = input("question")\n\nif (answer == "yes"):\n print ("yes response")\nelif (answer == "no"):\n print ("no response")\nelse:\n print ("other response")</pre>	<pre>answer = input("question")\nanswer = answer.lower()\n\nif (answer == "yes"):\n print ("yes response")\nelif (answer == "no"):\n print ("no response")\nelse:\n print ("other response")</pre>

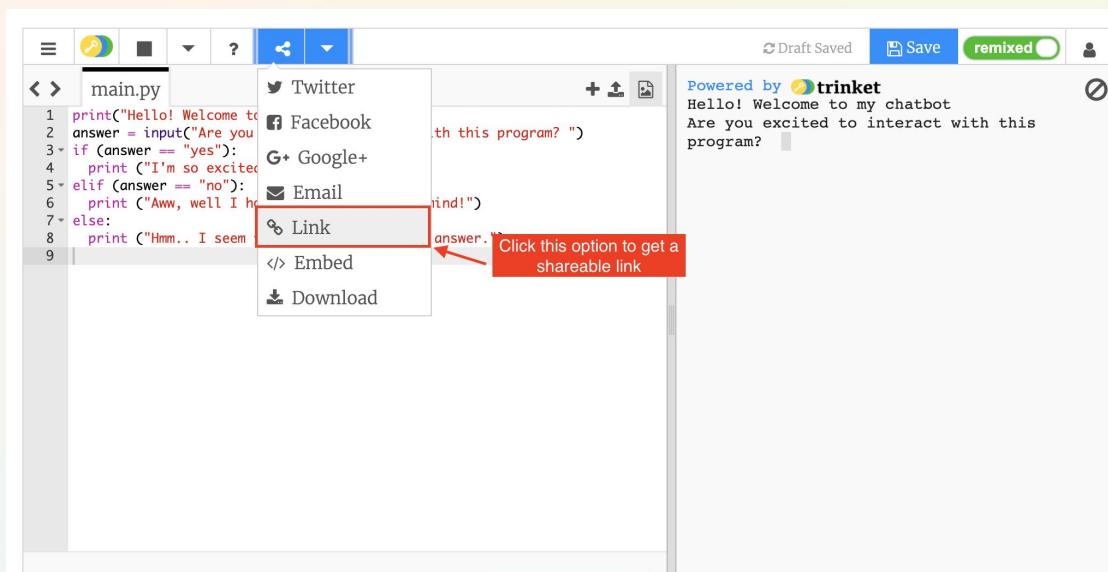
We must add this new line after **every** question is asked.

Step 10: Share Your Girls Who Code at Home Project! (5 mins)

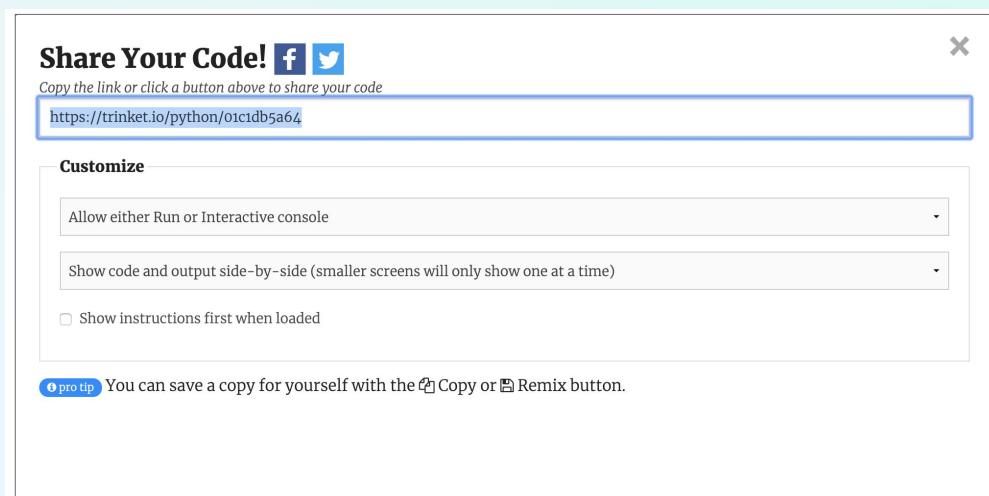
- Don't forget to save your work by clicking the **Save** button on the right side of the Trinket window



- To get a share your project with friends and family click the Share icon on the left and choose the Link option in the drop down menu.



- Share a photo or video of your working chatbot or copy the link and share your chatbot on social media! Don't forget to tag @girlswhocode and use the hashtag #codefromhome and we might even feature you on our account!



Can I Help You? Project Planning Worksheet

Project Overview Planning

Theme: What is the topic that will be covered with your chatbot?

Audience: Who is this chatbot going to serve? What group of people will be interested in your product?

Goal: What do you want your chatbot to accomplish? Why is this of interest to your audience?

Question Planning

Choose three yes or no questions to ask your users. Write the question in the first row and your chatbot's response in each row for "yes" or "no". Python is very picky in accepting answers, the last option "other responses" is to respond to answers other than yes or no. You may want to add a message letting the user know that their response was invalid.

Question #1:	
Yes	
No	
Other Response	

Question Planning Continued

Question #2:	
Yes	
No	
Other Response	

Question #3:	
Yes	
No	
Other Response	