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.

Step 1: Explore

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

You may want to use the planning document (can be found at the end of this document) 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, humorous, 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.

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 with Python

Copy the starter Python code, ChatBotStarter.py, into your development environment.

Step 4: Explore the Starter Project

Let’s take a look at the starter file.

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

In order for the computer to “talk” to the user we write print statements in code and it then shows the statement in the command window.

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.

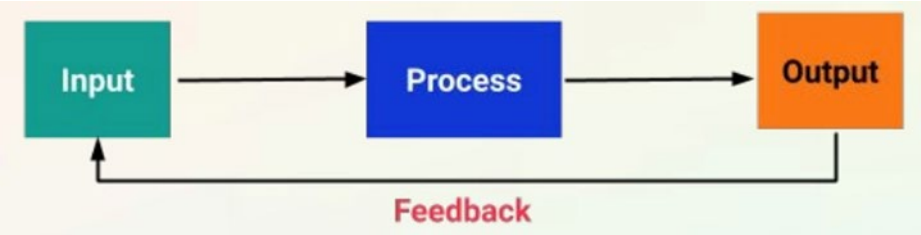
See if the computer prints out your introduction.

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

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.

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.

Step 7: Respond to an Answer

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.

```
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.

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

Step 8: Asking More Questions to Gather Data

Now that you have written one question, follow the instructions in steps 5 and 6 to ask your remaining questions. After you finish writing one question don't forget to Test your program to make sure it runs correctly. USE FUNCTIONS. An example has been provided in the Python file, ChatBotExpanded.py.

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

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

- Add a question with responses other than "yes" or "no".
- Add a question with more than two possible responses.
- Remove case sensitivity for responses.
- Use loops to ask a question repeatedly that may require different answers from the user each time.
- Use functions to reduce repeated instructions.
- Use lists to display multiple options to the user or to store multiple values from the user.

Project Planning Worksheet

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?