Sanny Ye

"Unlock the power of conversation and calculation with a revolutionary chatbot that effortlessly solves mathematical problems while engaging in interactive dialogue, making complex equations a breeze for users."

Passion Project

Math Problem Solving Chatbot

https://github.com/SY7ARIES/Math-problem-solving-chatbot

Contents

Introduction

This project is aiming to solve mathematical problems for people who needs or struggles on math problems. My project is going to be created by using the Python language and it is going to solve math problems in the form of chatting with the users like a chatbot. Python was chosen to complete the project was because it is widely accepted as beneficial as it is a powerful, versatile, and easy-to-learn programming language with a large community of developers, libraries, and tools, making it ideal for a wide range of applications. The target audience for this project would be people in any age as long as they like to solve mathematical problems, this chatbot can help them to do simple calculations and verifications more efficiently.

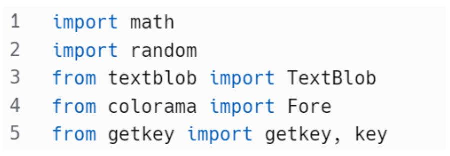
Background and Related Work

This mathematical problem solving chatbot is going to be created and tested on the coding website, Replit, and is going to mainly work by if statements, while loops and mathematical calculations with variables. Before the I started to create this project, she has made a simple chatbot about calculating the constellations of the users. This passion for this project was that the I wanted to involve more complex math calculations in the project. The math problem solving chatbot might sound similar to a scientific calculator. However, there are three main differences:

1. Better communication: This chatbot is able to solve the user's mathematical problems in the form of keep asking the user questions (chatting), while the scientific calculators contain lots of complex mathematical terms.
2. Humanisation: This chatbot is able to solve the type of test condition questions by collecting all the conditions given in the problem while chatting with the users. As there are enough rational conditions, the answer would be calculated automatically. While scientific calculations require insert the whole equation in order to solve for pronumerals.
3. Solution: Other than just giving the user the answer to the mathematical question, this chatbot also provides the equation of how the result came out, this provides the user a better understanding on not just the result but also the process, this also increases the reliability of the result.

Project Implementation

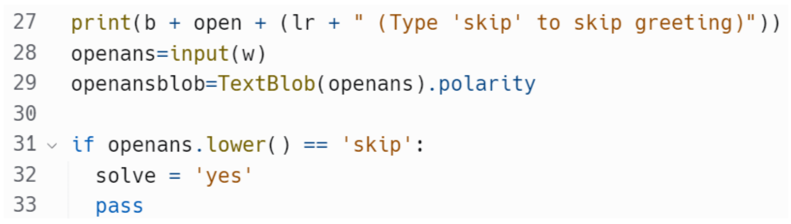
The project firstly started with importing math, random, textblob, colorama and getkey modules for a better performance in the outcome.



```python
import math
import random
from textblob import TextBlob
from colorama import Fore
from getkey import getkey, key
```

Then some lists and variables were set for later use. Mainly five colours for fonts were used in the project: white for user input, blue for chatbot's conversation, red for error messages, light green for question type selections and light magenta for all the answers.

A set of if…else statements were created for greeting the users, mainly asking for user's mood and explanation on the chatbot's name: Cici. Then after I have tried to communicate with my chatbot, I realised some old users might not want to do the unnecessary greeting with the chatbot again, they might just want to ask the chatbot a mathematical question. Therefore, I added a special keyword in the first output line from the robot for the old users: "(Type 'skip' to skip greeting)", so that if the users wish to, they are able to skip the greeting dialogues with the chatbot and go straight to solve the mathematical problems.



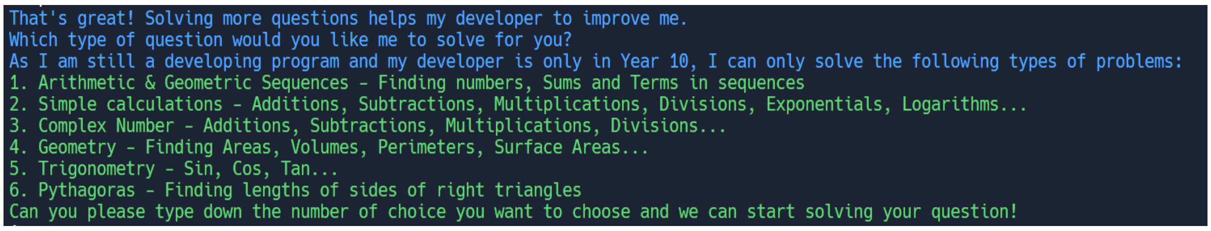
```python
print(b + open + (lr + " (Type 'skip' to skip greeting)"))
openans=input(w)
openansblob=TextBlob(openans).polarity

if openans.lower() == 'skip':
  solve = 'yes'
  pass
```

The whole greeting dialogues made the else statement for the if statement showed in the image above. It is mainly made by if..else statements, the textblob module and the random module.

Basically, the chatbot gives three different respond to three different types of respond which depends on the polarity index produced by the textblob module. However, whatever input the user inserts, the output respond is always going to lead to the same question of letting the user guess why the chatbot was called Cici. Then another couple of if..else statements were made for different types of user's respond.

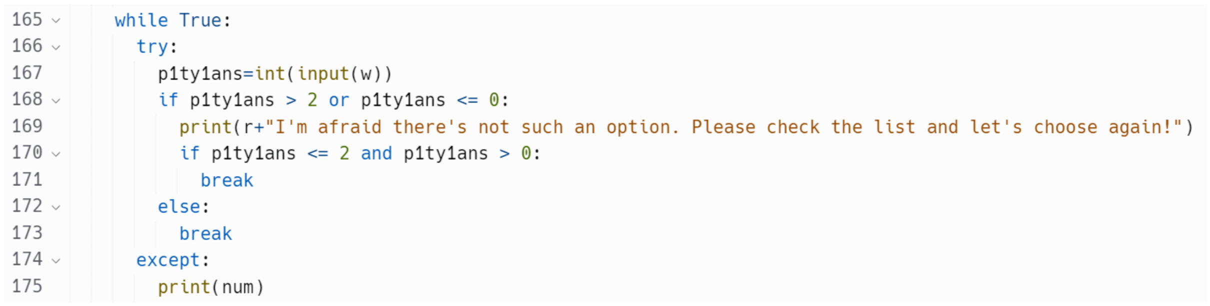
Towards end of the greeting or after the user inserts ‘skip’ for the first input, the whole math calculation begins with a big menu of asking which type of math question the user wants to solve. At the meanwhile, the whole math calculation has been put in a while loop and the calculation menu starts the loop. So, when the user finished one question, the chatbot always go back to the calculation menu.

The calculation menu always contains the same types of calculations and output question:



```
That's great! Solving more questions helps my developer to improve me.
Which type of question would you like me to solve for you?
As I am still a developing program and my developer is only in Year 10, I can only solve the following types of problems:
1. Arithmetic & Geometric Sequences - Finding numbers, Sums and Terms in sequences
2. Simple calculations - Additions, Subtractions, Multiplications, Divisions, Exponentials, Logarithms...
3. Complex Number - Additions, Subtractions, Multiplications, Divisions...
4. Geometry - Finding Areas, Volumes, Perimeters, Surface Areas...
5. Trigonometry - Sin, Cos, Tan...
6. Pythagoras - Finding lengths of sides of right triangles
Can you please type down the number of choice you want to choose and we can start solving your question!
```

However, an improvement can be made here as the chatbot was ‘commanding’ the user to insert the number of choice they want to choose, because I didn’t use an if..else statement but a try…except statement, therefore the user has to insert a float number, otherwise the chatbot would keep repeating the question as a ‘non-float’ answer would return False in the system. (num is a variable set earlier for input errors, used in most of except statements)



```
while True:
  try:
    p1ty1ans=int(input(w))
    if p1ty1ans > 2 or p1ty1ans <= 0:
      print(r+"I'm afraid there's not such an option. Please check the list and let's choose again!")
      if p1ty1ans <= 2 and p1ty1ans > 0:
        break
    else:
      break
  except:
    print(num)
```

After the calculation menu, the rest of (about) 1500 lines of the project are about solving mathematical questions. Six if…else statements were used to define the type of question the user chose.

1. Arithmetic & Geometric sequences: the user is asked to pick one sequence. While true, the input answer from the user has to be an integer (as the only option in the output question). Lots of while loops, try…except statements and calculations are used in this part. The inputs from the users are set to variables straight away and was used in the calculation and the final equation sample. Basic arithmetic and geometric sequences were used in the calculations. A special function used was pow(x,y) (for example), this returns the value of x to the power of y. And in the sum to infinity calculation, abs(x) were used (example), this returns the absolute values of x.

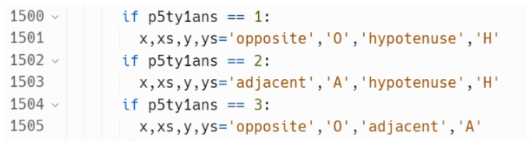
2. Simple Calculations: Just simple calculations. ($+$, $-$, $\times$, $\div$)

> 2.1. Additions: as addition is just add all the numbers together, so in this part, the chatbot allows the user to keep inserting all the addends by insert numbers and press ‘enter’. The sum will be calculated once the user pressed ‘enter’ three times. Therefore, the getkey module was used to read the keys the user pressed.
>
> 2.6. Logarithms: a problem I met in this part is when I tried to use log(), but as I tested the chatbot, I found that the output returns 2.999… as I inputted base 10 to 1000. So, I went to search online for ways to solve this problem, then I found and solved with the log10() function.
>
> 2.7. Square Root: the function math.sqrt(x) was used to return the value of square root of x.
>
> 2.8. Root of x: After I have done square root I started to think what if the user wants to know a different number of root? So, I searched online and found the math.cbrt(x) function, but it still didn’t solve my problem exactly. Then after I thought about this again, I found I can just use the pow(x) function as pow(y,1/x), so the user is able to insert the number of root they want.

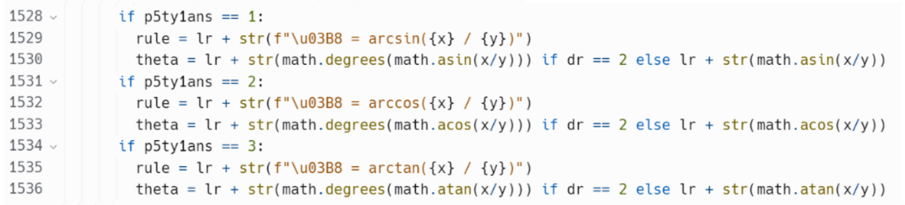
3. Complex Numbers: This part used mainly used complex() number type instead of float in the try…except statements. Others are just the same as the simple calculations.

4. Geometry: Calculating areas, surface areas, perimeters and Volumes for different shapes and prisms. math.sqrt(x) function was used again to solve problems. For calculations for circles, math.pi was used for state the rational number pi.

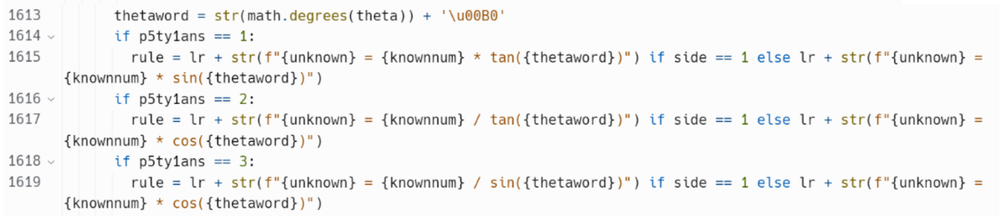
5. Trigonometry: Calculating angles and lengths of right triangles when the user gives valid conditions. The functions math.sin/cos/tan/asin/acos/atan(x) were used to solve problems in this part. As they ways of calculating the angles/lengths are the same, I didn't do the same thing as I did for the previous calculations, indeed, I used lots of if…else statements and input variables. e.g.:



```
if p5ty1ans == 1:
  x,xs,y,ys='opposite','O','hypotenuse','H'
if p5ty1ans == 2:
  x,xs,y,ys='adjacent','A','hypotenuse','H'
if p5ty1ans == 3:
  x,xs,y,ys='opposite','O','adjacent','A'
```



```
if p5ty1ans == 1:
  rule = lr + str(f"\u03B8 = arcsin({x} / {y})")
  theta = lr + str(math.degrees(math.asin(x/y))) if dr == 2 else lr + str(math.asin(x/y))
if p5ty1ans == 2:
  rule = lr + str(f"\u03B8 = arccos({x} / {y})")
  theta = lr + str(math.degrees(math.acos(x/y))) if dr == 2 else lr + str(math.acos(x/y))
if p5ty1ans == 3:
  rule = lr + str(f"\u03B8 = arctan({x} / {y})")
  theta = lr + str(math.degrees(math.atan(x/y))) if dr == 2 else lr + str(math.atan(x/y))
```



```
thetaword = str(math.degrees(theta)) + '\u00B0'
if p5ty1ans == 1:
  rule = lr + str(f"{unknown} = {knownnum} * tan({thetaword})") if side == 1 else lr + str(f"{unknown} = {knownnum} * sin({thetaword})")
if p5ty1ans == 2:
  rule = lr + str(f"{unknown} = {knownnum} / tan({thetaword})") if side == 1 else lr + str(f"{unknown} = {knownnum} * cos({thetaword})")
if p5ty1ans == 3:
  rule = lr + str(f"{unknown} = {knownnum} / sin({thetaword})") if side == 1 else lr + str(f"{unknown} = {knownnum} * cos({thetaword})")
```

Etc.

The functions math.degrees/radians(x) were also used in this part for users to chose which form of answer they want.

6. Pythagoras: This part mainly used the math.sqrt(x) function and ** operators.

There are certainly some limitations of my project as there was a time limitation, so I am going to improve and work on this chatbot furthermore in depth in the future.

Results and Evaluation

My project can be viewed on: https://github.com/SY7ARIES/Math-problem-solving-chatbot

The project is successful in the simple calculations, complex numbers, Geometry and Pythagoras parts. A serious problem I met was in the trigonometry part, as I test ran the chatbot, the result for sin(30)*50 came out as 24.999… while the result should be exactly 25. Then I did some more test runs on length that should come out as an integer, but they all came out as similar results like 24.999… As the time was limited for this passion project, so I didn't get time to fix this problem.

As I was brand new to Python and just learnt about Python in term 1 this year, I didn't know much syntax about Python, therefore, nearly all the codes in my project were something I learned. I had some confusion with the while loops and try…except statements, however, once I understood the rule of the indentations, I was able to fluently code what I wanted the output to be, just like how I type texts. I also had a huge confusion and took a long time to figure out the difference between break, pass and continue.

Conclusion

In conclusion, the key aspects of my chatbot are that it's able to solve simple and complex mathematical problems by getting information from user's inputs, it can also show the equation for solving the question to the user for a better understanding on the process of getting the answer. The significance of your project lies in its dual function as a calculator and a conversational agent. By incorporating conversational capabilities, my chatbot goes beyond the traditional calculator experience, giving users a more interactive and engaging way to solve math problems. This can enhance the learning and problem-solving experience for users, especially those who may have difficulty using traditional math notation.

Overall, I have successfully combined mathematical problem-solving and conversational abilities in my chatbot, providing users with a unique and user-friendly experience. It is important to evaluate user feedback and iterate on my project to further enhance its functionality and address any potential limitations or areas for improvement.

References

References

Canva. (2023, May 15). *Canva cover page*. Retrieved from Canva: https://edit.org/edit#

Chatgpt. (2023, May 15). *Math Chatbot Integration*. Retrieved from Chatgpt: https://chat.openai.com/c/e4107836-e957-4b23-814e-35bd429cad39