Girls' Programming Network

Secret Diary Chatbot Extension!

Keep your secrets safe from nosy siblings or friends with the Secret Diary Chatbot!

This project was created by GPN Australia for GPN sites all around Australia!

This workbook and related materials were created by tutors at:

Sydney, Canberra and Perth

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Courtney Ross
Alex McCulloch

Testers

Ash Liu
Aadeeba Mou
Manou Rosenberg

Extension 1: What's your style?

Task 7.1: Give it a fun theme!

In the instructions, we've based it on keeping a secret diary safe from our little siblings! **But you can add your own theme!!**

Think about a theme you want to add to your chatbot! Some ideas are below!!

A spy theme!!

The chatbot protects the secret identities and speaks like a secret agent.

A secret club for a school project!!

Keep copycats at bay and teach them a lesson for trying to steal your ideas! Teach cheaters a lesson!

Pokemon Go!!

Use your chatbot to store the secret locations of rare pokemon!

A fight between Good and Evil!

Every team has to store their secret info somewhere whether they are in Dumbledore's Army, The Rebel Alliance or the Powerpuff Girls!

Task 7.2: S-S-S-Style!

Use your theme!

1. Change, or add in more print statements to reflect your chosen theme!
2. Change your questions and answers so they match the theme as well!

Extension 2: Intelligent Checks!

Task 8.1: You're so smart!

You know lots of things and you can prove it! Let's add another level of security that checks if you are smart enough to read the diary.

1. Find the part of the code before we ask the user to enter the password.
2. Use `input` to ask the user the answer to an intelligence question. For example: `"What is 2 + 2? "` and store it in a variable called **`answer`**

Task 8.2: Or are you?

Now we need to check whether or not the answer is right!

1. Write an `if` statement that checks if the **`answer`** that they gave does NOT equal the right answer.
2. If the answer is wrong, `print` something to let them know that they aren't smart enough to get into your diary and use **`exit()`** to make the program stop right there.

Task 8.3: I know lots of things!

You can add as many of these questions as you want by copying your code from 8.1 and 8.2 and changing the question and answer! See if the person next to you can get past your intelligence checks!

Extension 3: Random Maths questions!

Note: only do this if you have already done Extension 2!

Task 9.1: That's so random!

Let's generate some random maths questions for security, so they change every time! We'll use the random module!

If you haven't already, at the top of your code, `import random`..

Task 9.2: Number Generator!

Now we need to generate the numbers for the maths question! Let's get the answer at the same time.

1. Randomly choose a number between `1` and `10`, and store it in a variable called `num1`.
2. Randomly choose another number between `1` and `10`, and store it in a variable called `num2`.
3. Add the two numbers together, and store it in a variable called `answer`.

Hint: Randomly Choosing a Number

You can select randomly from a range of numbers using `random.randint()`
Note: randint is short for Random Integer!

This code gets a random age between 0 and 100

```
age = random.randint(0, 100)
```

Task 9.3: Question Creation

Now we need to create the question, so we can ask it.

1. Make a String that asks what happens when you add **`num1`** and **`num2`** together. Store it in a variable called **`question`**.
2. Ask the user for input, using the question variable you just made as the prompt. Store the answer in a variable called **`user_answer`**

Hint: Converting numbers to strings

You can convert integers into strings, so we can print them:

```
age = 7
print("My age is " + str(age))
```

Task 9.4: Am I right?

Now we need to check their response.

1. Add an **`if`** statement that checks if the **`user_answer`** is NOT the same as the actual **`answer`**

2. If the answer is incorrect, change the **`access_type`** to either **`"denied"`** or **`"decoy"`**

Extensions 4: Make it Funnier!

Task 10.1: More security, more!

Think about jokes and tricks to add to your chatbot when the user guesses the decoy password. Some ideas are below!

A Magic 8 Ball!

Trick imposters into thinking that your secret is a magic 8 ball. Distract the imposter with you magic 8 ball!

```
Ask the magic 8 ball for advice: Will it rain today?
>>> Yes
Ask another question: Should I go shopping?
>>> Maybe
Ask another question: Will it rain today?
>>> Ask again later
```

ASCII Art

Add some ASCII art for fun!!!

```
Here is the super secret info
...
                _=,_
              o_/6 /#\\
              \__ |##/
               ='|--\\
                 /   #'-.
                 \#|_   _'-. /
                  |/ \\_( # |"
                 C/ ,--___/
Woof...
```

Task 10.2: Magic 8 Ball

Let's create a Magic 8 Ball!

1. At the top of your code, `import random`. This will let us choose things randomly!
2. Create a `list` called `answers`. Add in some Magic 8 answers to this list, like `"Yes"`, `"No"`, or `"Maybe"`. Add as many answers to this list as you like!
3. Use `input` to ask the user what question they'd like the answer to.
4. Use `random.choice` to select an answer from the answers list. Tell the user!
5. Add a `while` loop, so your code keeps asking questions and giving random answers.

Hint: Lists

You can create a list of your favourite foods like this:

```
myFave = ["pizza", "chocolate", "nutella", "lemon"]
```

Hint: Random choice

You can use `random.choice` to select a random number between 1 and 3 like this:

```
mynum = random.choice([1, 2, 3])
```

Task 10.3: ASCII art

Let's show some ASCII art to our secret agents!

1. Pick some ASCII art! You can choose from http://www.ascii-art.de/ascii/, or make your own!

2. Copy and paste it into your program. You will need to `print` your ASCII art.

Note: If your picture doesn't print out exactly as you expect you might have quotes inside it that are breaking your print. Edit the picture, or choose another!

Hint:

You can use triple quotes to `print` things on multiple lines using one `print` statement.

```
print("""This is
A really really
Long answer""")
```

Extension 5: Em🍊jifacti🍊n Functi🍊n

Right now our diary is protected from snoops when they use the chatbot, because they need the password. **But what if someone just goes and opens the text file and reads it!**

We'll make our words unreadable by encoding them with emoji!

Task 11.1: Emoji-fy function

Let's write a function that will convert messages to emoji-fied messages

1. Define a function called emojify that takes 1 parameter, the message we want to put emoji's into.

2. We'll start by replacing all of the letter "a" with the 🍎 emoji using this line of code:
   `message = message.replace("a", "🍎")`

   To get the emoji you can go to https://emojityper.com/, search for apple and copy it.

3. Return the message (which is now apple-ified!)

Task 11.2: Emoji-fy function

Time to use our emojify function before we store our secret diary entry

1. Go to the place in your code after you ask the user for their diary entry. (before you write it to the diary file). Use the emojify function on the diary_entry and then write the emojified entry to the file.

2. Emoji are different to normal letters so we need to tell Python to do things a little differently when we write to the file. Add `encoding = "utf-8"` at the end of your file open command so it looks like this: `open("secret_diary.txt", "a", encoding="utf-8")`

3. Test your code!
   - Run your code, and add a new message (make sure it includes the letter "a")
   - Look in your secret diary file by hand, does it have an apple?
   - Try and read your diary with your program.
     …. Whoops there's still apples there. We haven't written the decoder yet!

Task 11.3: De-apple-ify your diary!

We want to read the diary again so we need to put the letters instead of emoji back in!

1. Start a new function called demojify, which takes one parameter, the content you want to put letters back into.

2. We'll do the opposite of what we did before. Use replace again. But this time replace `"🍎"` with `"a"`. *(switch the order from before)*

Task 11.4: De-apple-ify your diary!

Now to use the demojify function to make our diary readable again!

1. Go to the place in your code after you read the content of the diary. Use the demojify function to put the letters back in the text. Then print out the content.

2. Add the **`encoding = "utf-8"`** code when you open the file for reading here too! It should look like `open("secret_diary.txt", "r", encoding="utf-8")`

3. Test your code!
    a. Run your code, and add a new message (enter a message with an "a" in it)
    b. Try and read your diary with your program. Are the apples gone?

Task 11.5: More emoji is more secret!

The more letters you replace with emoji the harder it will be to understand any words.

1. Repeat the encoding and decoding process above, but for different letters and emoji. You can use *emojityper* to find more emoji to copy.