Girls' Programming Network

Sassy Security Chatbot!

Keep your secrets safe from nosy siblings or friends with the Sassy Security 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

Sarah MacLeod
Renee Noble
Vivian Dang
Courtney Ross
Lauren Northcote
Monique Dyson
Malar Sreedaran
Annie Liu

Testers

Pauline Kelly
Meike Moeckel

Part 0: Setting up

Task 0.1: Set Up the File

Open the start menu, and type 'IDLE'. Select the IDLE 3.X version.

1. Go to the file menu and select 'new file'. This opens a new window.
2. Go to the file menu, select 'Save as'.
3. Go to the Desktop and save the file as 'chatbot.py'

Task 0.2: You've got a blank space, so write your name!

At the top of the file use a comment to write your name! Any line starting with # is a comment.

```
# This is a comment
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 1:

- ☐ You should have a file called chatbot.py
- ☐ Your file has your name at the top in a comment
- ☐ Run your file with F5 key and it does nothing!!

Part 1: Welcome to the secret club

Task 1.1: Welcome

Let's say hi to our user!

1. `print` out a message to welcome the user

For example:

```
Welcome to the secret club!
Let's find out if you're cool enough to be in the secret club...
```

Hint

Use `print` to say whatever you like!

```
print("Hello, Everybody!")
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 2:

- ☐ Print a welcome
- ☐ Try running your code!

★ BONUS 1.2: Wait a second!

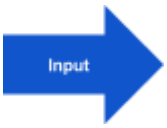
Let's make our chatbot a bit more human. We can make it wait a bit between responses, so it looks like it's thinking.

1. At the top of your file write:
   ```
   import time
   ```

2. Before any print statement, add:
   ```
   time.sleep(3)
   ```

This will make your program 'sleep' for 3 seconds!

Change the number to change how long the chatbot 'thinks'. Don't make it too long though!



Part 2: Who's there?

Our chatbot needs to know who it's talking to. Let's get it to ask the user for their name!

Task 2.1: Ask the user for their name and store it in a variable!

Our chatbot needs to know who it's talking to. Let's get it to ask the user for their name!

1. Using `input`, get the chatbot to ask for a name. Store it in a variable (maybe use **`user_name`**)
2. Get the chatbot to welcome the player using the **`user_name`**.

For example:

```
Please enter your name: Jasmine
Welcome, Jasmine
```

Hint

We can use a **+** to add a variable to some text, like:

```
print("Hello, " + variable_name)
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 3:

- ☐ You have the user's name stored in a variable.
- ☐ You have welcomed the user using their name.
- ☐ You have run your code to make sure it works.

★ BONUS 2.2: A Sassier Welcome

Add some extra sass to your welcome to the user. It could be:

```
Please enter your name: Jasmine
Well well well, it's Jasmine and you are 8. We'll see if you
are worthy...
```

Part 3: Password Please!

Task 3.1: Make up some passwords!

To keep our site secure we need a **secret password** to give to approved members. To trick snoops we'll also have a **decoy password** to give them trick information. *(A decoy is used to lead someone into a trap).*

1. Create a variable called **`real_password`** and store a password in it.
2. Create a variable called **`decoy_password`** and store a different password in it.

Some password ideas are: `"opensesame"` or `"PlEaSeOpEn"`

Hint

Using a variable to store a string is like:

```
variable_name = "this is a variable"
```

Task 3.2: Password Please...

Now we need to check if the user knows the password!

1. Ask the user for the password using `input`. Store the answer in a variable called **`user_password`**.

For example:

```
What's the secret password? opensesame
```

Task 3.3: Is the Password Correct?

Check if they got the correct password!

1. Use an `if` statement to check if **`user_password`** matches the **`real_password`**.
2. If it does match, let's show the user some *real* secret information inside the `if` statement.

Hint

You can check to see `if` it's raining like this:

```
if raining == "true":
    print("It's raining!")
```

Task 3.4: Decoy Password Trick!

1. Add an `elif` statement to your `if` statement to check if **`user_password`** matches the **`decoy_password`**.

2. If it does, `print` out some fake secret information to trick the snoops with a decoy password!

It should look something like this, if the decoy password is `"mybirthday"`:

```
What's the secret password: mybirthday
Well here's the link to the secret club information:
https://sites.google.com/site/girlsprogrammingnetwork/
It would have just been easier to google it :P
```

Hint

You can add an `elif` to an `if` statement like this:

```
if raining == "true":
    print("It's raining!")
elif sunny == "true":
    print("Wear a hat!")
```

Task 3.5: Imposter Alert

If the password is anything other than the real password or decoy password, tell them they're an imposter!

1. Add an `else` statement to your `if-elif` statement.

2. Inside your `else` statement, `print` out that the user is an imposter!

For example:

```
What's the secret password: password1234
Well Well Well, I see we have an imposter here!
```

Hint

You can add an `else` to an `if-elif` statement like this:

```
if raining == "true":
    print("It's raining!")
elif sunny == "true":
    print("Wear a hat!")
else:
    print("I don't know what the weather is today")
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:

☐ You have created a real password and a decoy password.

☐ You have asked the user for the password.

☐ You print out a different message to the user, depending on which password they used.

★ BONUS 3.6: Add a Personal Touch

Make the text you print out more personal! Mention their name, maybe give them a cool nickname based on the first letter of their name!

It could look something like

```
Please enter your username: Jasmine
Well well well, it's Jasmine
What's the secret password: mybirthday
Well J-Swizzle here's the link to the secret club information:
https://sites.google.com/site/girlsprogrammingnetwork/
It would have just been easier to google it :P
```

Or even

```
Please enter your username: Maddie
Well well well, it's Maddie
What's the secret password: opensesame
Welcome Agent M, glad you have joined the fight against evil!
Secret club documents in here!
```

Part 4: Wrong Password! Try again!

Task 4.1: Try again!

So the users put in a password! But it's wrong! Let's give the user another chance.

1. Find the spot in your code that is after you ask the user for their password, but before you check to see if the password is correct.

2. Add a `while` loop that keeps running while **`user_password`** does not equal the **`real_password`**.

3. Inside the `while` loop, `print` a message telling the user that the password was wrong.

4. Inside the `while` loop, ask the user to `input` the password again, storing it in **`user_password`**.

This would look something like this, if the secret password was "`opensesame`"

```
What's the secret password: open
Well it looks like someone's forgetful :P
Try again, what's the secret password: pLea5e0P3N
Well it looks like someone's forgetful :P
Try again, what's the secret password: opensesame
Looks like you pass, welcome to the secret club!
Secret club documents in here!
```

Hint: While Loops

You can keep asking questions in a `while` loop like this:

```
while answer != "4":
     answer = input("What number am I thinking of? ")
```

Task 4.2: Don't forget the decoy!

But we want the loop to stop if we put in the decoy password too, not only the real password!

1. Add the condition to your `while` loop so it keeps going while **`user_password`** doesn't match the **`decoy_password`** as well.

Hint

You can use `and` to check multiple conditions:

```
if today == "Monday" and holidays == False:
    print("It's time to go to school!")
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 5:

☐ You keep asking for the password while the user hasn't given the real password or the decoy password

☐ You have run your code to make sure it works.

★ BONUS 4.3: Too many tries!

Right now, we keep asking the user what the **`user_password`** is until they enter either the **`decoy_password`** or the **`real_password`**! This means that the user can keep guessing until they get it right.

1. Create a **`variable`** called **`attempts`** just before the `while` loop. Set it to `0`.

2. Add another condition to your `while` loop, so it only runs while **`attempts`** is less than `3`.

3. Inside your `while` loop, add `1` to **`attempts`** each time!

Now the user will only get 3 attempts to get the password right, before being locked out!

Part 5: Intelligence Test Time

Task 5.1: Ask the question

Let's add some extra security in case someone guesses our secret password!

1. Find the spot after your `while` loop that asks for the password, but before the `if` statement that checks the password.
2. Ask the user a maths question, such as `"What is 5 + 7? "`. Store the user's response in a variable called **`answer`**.
3. Convert the **`answer`** to an `int`eger, storing it again in **`answer`**.
4. Store the correct answer to the maths question in a variable called **`correct_answer`**.

Hint

You can convert a string to an integer like this:

```
myNum = "4"
myNum = int(myNum)
```

Task 5.2: Check the answer!

We need to check the answer, then store whether they passed or not!

1. Use an `if` statement to see if **`answer`** is equal to **`correct_answer`**.
2. If it is, create a variable called **`test_result`** and store the word `"passed"` in it.
3. Add an `else` to your `if` statement. Inside it, create a variable called **`test_result`** and store `"failed"` in it.
4. Add some `print` statements to sassily tell your user if they passed or failed!

Task 5.3: Secure the secret information

Use the test result to help decide what secret info they get to see!

1. Go to the `if-elif-else` statement at the end of your file that displays the different secret information.

2. Update the `if` part of your `if-elif-else` statement, so it only gives out the real secret information if the user entered the **`real_password`** and the **`test_result`** is **`"passed"`**.

3. Update the `elif` part of your `if-elif-else` statement so it displays the fake secret information if the user entered the **`decoy_password`** or the **`test_result`** is **`"failed"`**.

4. The `else` part can stay the same!

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 6:

☐ You ask the user a maths question, and check to see if they passed or not.

☐ The user only gets the real secret information if they got the password and the maths question correct.

☐ The user only gets the fake secret information if they got the decoy password or the maths question incorrect.

Part 6: Harder maths!!

Task 6.1: This is random!

So right now it asks the same maths question every time! This means sneaky little kids could just keep guessing different numbers until they get the right answer!

Let's change the code so that the maths question is randomly generated!

1. At the top of your code, import the **`random`** package using `import random`

Task 6.2: Random Numbers

Now we want to randomly choose 2 numbers between 1 and 10!

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. `print` **`num1`** and **`num2`** to make sure it worked! You can `comment (#)` this `print` statement out once you're sure it works.

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 6.2: A Better Question and Answer

Let's change our maths question so it uses our random numbers!

1. Update the `question` to use `num1` and `num2` instead of the two numbers you hard-coded in before.

2. Update the `real_answer` to match our random numbers! Calculate the `real_answer` using `num1` and `num2`, like you would in maths!

Hint: Changing integers to strings

We can't just add integers (numbers) to strings. To change an integer to a string, we can use:

```
mystring = str(6)
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Extensions:

☐ Your chatbot randomly selects two numbers.

☐ Your security maths question uses the two numbers for the question and answer.

★ BONUS 6.3: Bigger Numbers!

Ten numbers just isn't enough!! We want more numbers for bigger harder questions, but don't want to spend a lot of time writing out lots of numbers that's boring!

1. Change your line of code that randomly selects a number between `1` and `10` to use `range` instead. Make it choose between `1` and `100`!

Hint: Range

You can use `range` to select a generate a list of numbers between 5 and 20 like this:

```
mynum = range(5, 21)
```

Note: `range` doesn't include the last number!

Extension 7: 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 8: Make it More Secure!

So we now have a password and a random maths question to protect our secrets. But is that enough security?

Task 8.1: More security, more!

This is where you can get creative and come up with your own security questions. **Put on those thinking hats!!**

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

A question only you know!

Just like Dumbledore asked Harry when Lord Voldemort returned.

```
What's my favourite flavour of jam?
Strawberry
```

A joke question

If you want to add in some fun and humour, why not add a joke?!

```
What do you call a bear with no teeth?
A gummy bear!
```

A knowledge question

If you're super smart, maybe you can ask a smart question. Just like the Ravenclaw dormitory password is the answer to a difficult question!

```
What's the fastest bird in the world?
Peregrine falcon at 389km/h
```

Complete the song lyric!

If you like Taylor Swift, you could have something like:

```
Cause the players gonna play, play, play, play, play
And the haters gonna:
Hate, hate, hate, hate
```

Task 8.2: Let's add that extra layer of security!

Have an idea now? Great! Here are the steps you need to take to add in your extra layer of security. These questions are tailored for a security joke question, but if you're using a different idea, make sure you change the variable name!

1. Let's make a new variable and ask the question. Use the variable `joke_question`.

2. Store the answer to the joke_question in a variable called `joke_answer`.

3. Just like in Task 5.2, we need to put in a checking mechanism to see if they got the correct answer! Don't forget to print a message saying if the user got it right or wrong!

★ BONUS 8.3: Pick another security layer!

There's no limit to how much security you can have. Feel free to be creative and add in as many as you can think of.

1. Follow the steps in **`Task 8.2`** again and again! Make sure to give each security question and answer different variable names.

Extensions 9: Make it Funnier!

Task 9.1: More security, more!

Think about jokes and tricks to add to your chatbot. 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 9.2: Magic 8 Ball

Let's create a Magic 8 Ball!

1. 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!
2. Use `input` to ask the user what question they'd like the answer to.
3. Use `random` to select an answer from the answers list. Tell the user!
4. 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"]
```

Task 9.3: 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 10: Make Maths Harder!!

Task 10.1: Asking the question

Only having addition is too easy especially as your snoopy younger siblings get older and better at maths!

1. Create a `list` called `operators`. In this list, store `"plus"`, `"minus"` and `"times"`.

2. Randomly choose an item from the `operators` list. Store it in a variable.

3. Update how you ask your question, so it uses the operator you selected!

For example, if we randomly chose `"times"`, we want the question to say:

```
What is 4 times 7?
```

Task 10.2: Calculating the answer

Now we need to make sure we calculate the answer correctly based on the operator we selected!

1. Create an `if-elif` statement that calculates the correct answer, based on whether the operator is `"plus"`, `"minus"` or `"times"`. Store the answer in a variable.

2. Test your code to make sure it works!

Extension 11: The More Maths the Merrier!

Task 11.1: Asking lots of questions!

One maths question is not enough! We want to keep asking them questions, even if they'd never get in anyway.

1. Add a `for` loop so we can ask them `5` maths questions!

2. Move the code that asks the maths questions inside the `for` loop.

Hint: For loops

You can loop a set number of times with a `for` loop and `range`:

```
for i in range(10):
    print(i)
```

Task 11.2: A better scoring system!

Instead of storing `"passed"` or `"failed"` for our test result, let's keep count of how many they get right by storing a number instead!

1. Create a variable called `test_result` and set it to `0`.

2. Change your `if` statement that checks to see if they got the answer right, so `1` is added to `test_result` when they do.

3. Change the `if` statement that checks to see if `test_result` is `"passed"` or `"failed"` to check if they got enough questions right. Make it so they have to get at least `3` out of `5` right!.