Girls' Programming Network

2017

Sassy Security Chatbots!

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

**This workbook and related materials were created by tutors at:
GPN Sydney and GPN Canberra**

Girls' Programming Network - Sydney

Girls' Programming Network - Canberra

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

Writers

Courtney Ross
Renee Noble
Catherine Murdoch
Christie Boyd
Meike Moeckel
Kelly Stewart
Kassandra Di Bona
Amanda Hogan

Testers

Wendy Dourado

Sassy Security Chatbot

Keeping your secrets safe is just plain hard! Especially if you have nosy people in your family who won't stop snooping around!

We're taking our diary digital! We just need to give it a great security system! Our security chatbot that will keep our secrets safe and trick any snoops lurking around! Only people with the real password will be able to access the secrets! **We're going to run it on a server so you can access it on the go!**

Part 0: Files and Themes!

Goal: Let's get our file and our ideas set up before we start!

Task 0.1: Set Up the File

First we need to create a file where we are going to write the code for our game.

1) In your Python 3 IDLE click ***File*** and create a ***New File***.
2) Start by saving your file to the desktop and calling it ***secretserver.py***

Now when you want to run your code, just click ***Run*** (or press ***F5*** on the keyboard)!

We also need to make sure that we have two extra Python files on the desktop:

1) portal.py

2) helper.py

These two programs are going to help us make and run our secret server!
Great! Now we're ready to code!

Task 0.2: Fun Themes

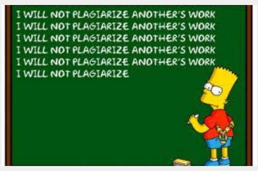
In the instructions, we're going to base it on keeping a secret diary safe from hackers and most importantly little siblings! **But you can add your own theme!!**

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!!


I WILL NOT PLAGIARIZE ANOTHER'S WORK
I WILL NOT PLAGIARIZE ANOTHER'S WORK
I WILL NOT PLAGIARIZE ANOTHER'S WORK
I WILL NOT PLAGIARIZE ANOTHER'S WORK
I WILL NOT PLAGIARIZE ANOTHER'S WORK
I WILL NOT PLAGIARIZE

Keep your ideas safe from copy cats! Trick any sneaks and teach them a lesson!

Pokemon Go!!

Use your chatbot to store the secret locations of rare pokemon! Only share it with people on your team!

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!

Part 1: Using the helper!

Today, to create our secret club we're going to need a little bit of help! Luckily, we've got a great little helper, a special library that helps us send messages!! We just have to import it!

Goal: Import the helper and make sure it's working!

Task 1.1: Importing your helper!

In your file ***secretserver.py***, `import helper`

Task 1.2: Test It!

The helper has a function called test, it checks that the helper is working correctly! The test function takes two numbers, adds them together and gives them back!

Step 1) Use `helper.test(.......)` Inside the brackets give it two numbers to add!
Hint: *Don't forget to store it in a variable!*

Step 2) Print out the result!

Step 3) Run your code!
Once you've run the test you can get rid of this test code.

Part 2: Connecting our Portal

We want to be able to send messages from computer to computer which means we need to know where to send them! Just like when your send a letter to a house, when you send a message online you need to know the address. To send a message to the server we need to know its address!

1. **Our IP Address is 127.0.0.1**
2. **Our Port Number is 8888**

Goal: Set up the address for our server and connect to it!

Task 2.1: Give our socket an address!

Step 1: Store the IP Address in a variable called server_ip
__Hint:__ This one needs to be a string

Step 2: Store the port number in a variable server_port
__Hint:__ This one needs to be an integer

Step 3: Give our socket an address!
Use `helper.bind(server_ip, server_port)`

Task 2.2: Wait for a user!

Now our server has an address, we need to wait until a user connects!

Step 1: We need to listen for anyone who is joining our server, `helper.listen(1)` is how we tell our program to listen!
Tip: The 1 in the brackets just means we are listening for one user to join

Step 2: Print a message that says we are waiting for a user to connect

Task 2.3: Accept the user's connection!

When a user tries to connect we need to accept their connection!

Step 1: We need to accept the connection using this code:

```
connection = helper.accept()
```

When we accept the connection we get back the connection details!
We'll use the connection later to send a message to the right place!

Step 2: Add a print statement saying that we have connected a user!

Step 3: Run **secretserver.py**

Step 4: Run **portal.py**

Task 2.4 : Send a message to the user

We can use the connection we just accepted to send a message by using this code:

```
helper.send(connection, 'message')
```

Step 1) Send a welcome message to the user! Make it any message you like!

Part 3: Who's there??

Now we are connected we can give the user a personalised welcome! But we need to know who they are!

Goal: Ask the user who they are and send them a personalised welcome message

Task 3.1: Ask a question

We want to ask a question, so we need a different kind of message sender that waits for an answer. **You can ask questions like this:**

```
age = helper.ask_question(connection, "How old are you?")
food = helper.ask_question(connection, "What's your fave food?")
```

Step 1) Change the code above! Ask the user what their name is

Hint: Don't forget to end your question with a **question mark**, this tells the helper to wait for an answer!

Step 2) You can test that you got their name by printing it out

Task 3.2: Send a personalised welcome

Step 1) Using the helper's send function, send the user a personalised welcome message **that contains their name**!

Hint: Remember how we can add strings together!

Task 3.3: Slow Down!

Woah, our server sends messages fast! We want our chatbot to be more human like. Humans need time to think before responding, so let's slow down! Put a delay before sending each message!

We need a special library that can freeze time!

Step 1) At the top of you file add `import time`

If we want our program to pause for 3 seconds we can write: `time.sleep(3)`

Step 2) Add this line of code between getting the user's name and sending a welcome Message.

Step 3) Change the number in the brackets to make the message send really fast or really slow!

Step 4) Add this line of code before each of the messages or questions you have already sent!

Part 4: Password Please!

We know who our user says they are, but now we need them to prove it!
To keep our server secure we need a **secret password** to give to authorised users.
To trick snoops we'll also have a **decoy password** to give them trick information.

Goal: Check if the user has the <u>real password</u>, the <u>decoy password</u> or a <u>password that is just wrong</u>. Give them the information they are allowed to access.

Task 4.1: Make up some passwords!

Step 1) Create a variable called `real_password` and store the real password in there

Step 2) Create another variable called `decoy_password` and store the decoy password

Hint: Some password ideas are…. "`opensesame`" or "`PlEaSeOpEn`"

Task 4.2: Password Please...

Step 1) Ask the user for the password. Store it in the variable user_password.
Hint: *Use helper, and don't forget the question mark!*

Step 2) Don't forget to **add in the delay** before you ask the question!

Task 4.3: Password Correct

Check if they got the correct password!

Step 1) Add an *`if`* statement that checks If the password matches the <u>real password</u>,

Step 2) If it does match, let's show them some <u>real secret information</u> inside the if statement

It should look something like if the real password is "`opensesame`":

```
What's the secret password?
opensesame
Looks like you pass, welcome to the secret club!
Top Secret sleepover ideas:
   - Eat nutella pizza
   - Play truth or dare
   - Watch all the Harry Potter movies in order!
```

Task 4.4: Imposter Alert

If the password is anything other than the real password, tell them they're an imposter!
Hint: *Add an* `else` *statement to your* `if` *statement*

It could look something like:

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

Task 4.3: Decoy Password Trick!

But what if an authorised user has been compromised, and has given the imposter a decoy password to the secret server?
If the password matches the decoy password, let's show them some trick information!

It should look something like 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: *Add an* `elif` *statement to your* `if` *statement , before your* `else` *statement*

★Bonus 4.4: 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 info:
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 haved joined the fight for evil!
```

Part 5: Wrong Password! Try again!

Our secret server is pretty nice, and knows that occasionally our user won't spell their password right. Let's give the user multiple tries at getting their password right!

Goal: Allow the user to try and enter the password again if they get it wrong!

Task 5.1: Try again!

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

Step 2) Add a while loop that keeps running while the user's password doesn't equal the real secret password

Step 3) Inside the while loop send a message telling the user telling them they got the wrong password

Step 4) Next Inside the while loop we need to ask them again for their password
Hint: *Don't forget to sleep!*

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?
opensesame
Looks like you pass, welcome to the secret club!
Secret club documents in here!
```

Task 5.2: Don't forget the decoy!

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

Step 1) Add to the condition of you while loop, keep going while….

The user password doesn't equal the secret password
and
The user password doesn't equal the decoy password

Hint: *use and to join the two conditions together*

★BONUS 5.3: Too many tries!★

We can’t let the user try and guess the password forever, give them 3 tries!

Step 1) Before the loop, create a variable that counts how many guesses the user has had. Set it to 1.

Step 2) In the loop, update the counter after the user has another guess.

Step 3) Update the while loop condition, keep asking for the password while
The Secret password `and` *the decoy password are wrong*
`and` ***the counter is less than or equal to 3***