Girls' Programming Network

Secret Diary Chatbot!

In this workbook, you will make a chatbot that lets you write and read secret contents of your diary and add a password to keep it safe from nosy siblings or friends!

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
Renee Noble

Testers

Ash Liu
Aadeeba Mou

Intro to Python

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!!

Today's Project Plan:

Secret Diary Chatbot

We'll make a computer program that lets us read and write to a secret diary file. We don't want others to read our diary so we'll protect it with a password and secret questions.

1. Welcome the user
2. Deal with if they want to read or write to the diary
3. Write secret messages to the diary
4. Read the contents of the diary
5. Add a secret password to the diary

+ more

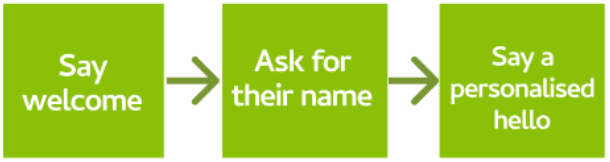
Once your base chatbot works add cool extensions!

There's extensions for encrypting your data with emoji, for adding tricky maths questions to protect from little siblings, and ways to make your chatbot sassier!

Part 1: Who are you?

Personal welcomes are the best!

Welcome the user, ask for their name and say hello!



Input & Variables

Task 1.1: Welcome User

It's time to welcome our user to the Secret Diary Chatbot!

`print` a welcome message.

For example:

```
Welcome to the Secret Diary Chatbot!
```

Task 1.2: Identify yourself!

The Secret Diary Chatbot needs to know who they're talking to.

Use `input` to ask the user `"What's your name? "`. Store the answer in a **`variable`** called **`name`**.

Task 1.3: Personalised Welcome

Now we can say hello using their name!

`print` a personalised welcome message, using the user's **`name`**.

For example:

```
Hello Courtney!
```

☑ CHECKPOINT ☑

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

- ☐ You have printed a welcome message.
- ☐ You have asked for the user's name.
- ☐ Your code prints a personalised welcome.

★ BONUS 1.4: 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.

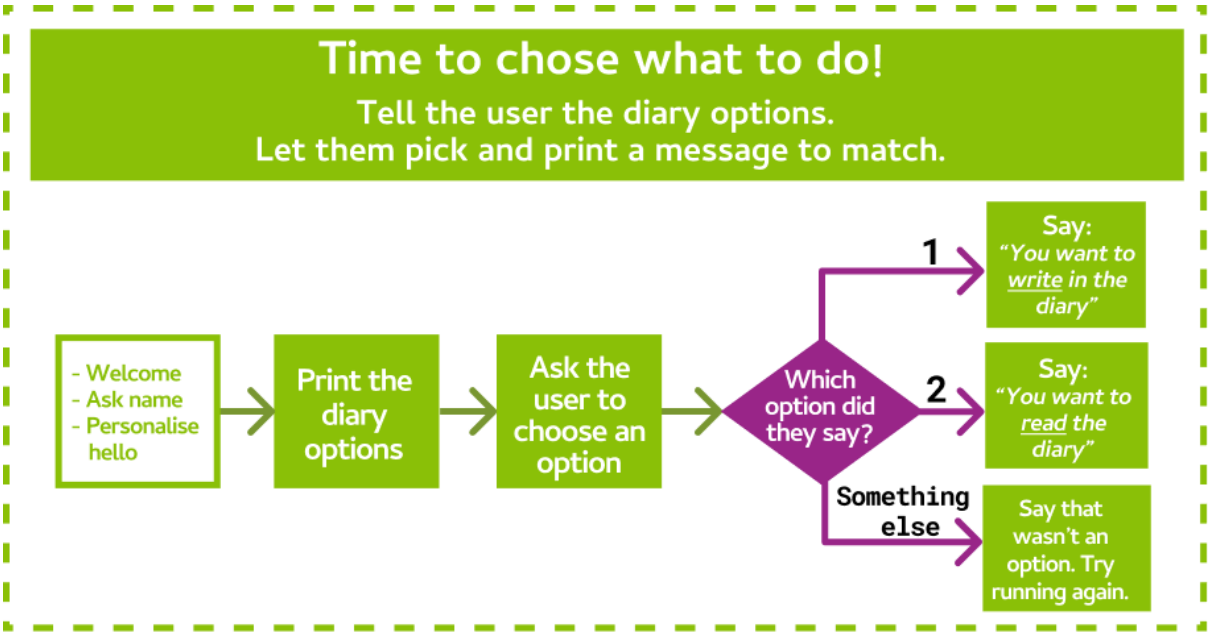
1. At the top of your code write:
   ```
   import time
   ```

2. Before any print statements you want, add:
   ```
   time.sleep(0.5)
   ```

This will make your program 'sleep' for half a second!

Change the number to change how long the chatbot 'thinks'. Don't make it too long though or it'll be really boring!

Part 2: What do you want to do?



Task 2.1: Welcome to the Secret Diary

Our Secret Diary Chatbot protects our secret diary! We can either read what's in it or add something new.

> Use `input` to ask the user what they would like to do to the secret diary. They can either type a 1 to write in the diary, or a 2 to read the diary. Store the answer in a variable called **`action`**.

For example, your code might print:

```
What would you like to do?
(1) Write in the diary
(2) Read the diary
```

Hint: Multi-line Print statements

You can print multiple lines of text in one print statement by using triple quotes like this:

```
print(""" Here is a line of text
Here is some more text on a new line
Here is a third line of text
""")
```

Task 2.2: Do you want to write?

First let's check to see if the user wanted to write in the diary.

1. Create an `if` statement that checks to see if the user entered `"1"` for the `action` to write in the secret diary.

2. If they did, `print` something to let them know. For example:

```
You want to write in the diary!
```

Hint: If Statements

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

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

Task 2.3: Do you want to read?

Now let's check to see if the user wanted to write in the diary.

1. Add an `elif` statement that checks to see if the user entered `"2"` for the `action` to read the secret diary.

2. If they did, `print` something to let them know. For example:

```
You want to read the diary!
```

Task 2.4: I don't understand!

If they put in something else, then we don't know what they want!

Add an `else` statement to tell the user something like:

```
I don't know what you want to do! Run the program again and
say either 1 to write or 2 to read
```

☑ CHECKPOINT ☑

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

- ☐ You have asked the user what they would like to do with the diary.
- ☐ If the user selects option 1, the program prints out that they can write a new line of text to the diary.

- [ ] If the user selects option 2, the program prints out that they can read the diary.
- [ ] If the user says anything else, we tell them that we don’t understand.

Part 3: Writing in the Diary

Files

Task 3.1: Tell me your secret

Time to find out what the user wants to write in the diary!

1. Find your `if` statement that checks if you want to write in the diary
2. Inside the if statement, use `input` to ask the user `"What do you want to write? "`. Store it in a variable called **`diary_entry`**.

Task 3.2: Open the diary!

We need to `open` a file to store our diary in! *To keep adding to the end of the file we'll open it in a mode called* ***`append`***.

1. Create a new line after you ask the user for their **`diary_entry`**.
2. Add a line of code that opens the diary in append mode and stores it in a variable called **`diary`**. The diary is in the file called `"secret_diary.txt"`

The file will be created in the same location as your **`chatbot.py`** file on your computer.

Hint: Opening a File

For example you can open a file in append mode like this:

```
groceries = open("groceries.txt", "a")
```

Task 3.3: Writing our secrets!

Now that we have the diary open, we can write our secrets into the diary!

1. We want each diary entry to be on its own line, so first we need to add a special symbol called the "newline character" to the end of **`diary_entry`** like this:
   `diary_entry = diary_entry + "\n"`

2. Write the line of code that will **`write`** the **`diary_entry`** into the open **`diary`** file.

Hint: Writing to a File

For example we can write a line into a file like this:

```
groceries.write("Eggs")
```

Task 3.4: Close the diary!

It's really important that we close the text file when we're finished with it!

`close` the diary.

Hint: Closing a File

For example you can close a file like this:

```
groceries.close()
```

☑ CHECKPOINT ☑

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

- ☐ When the user answers "1" they can tell the program what they want to write in the diary
- ☐ After you write something in the diary, if you open the file on your computer, it will have the new thing you wrote
- ☐ If you run the program again and write another thing and open the file on your computer, it will have the old thing you wrote AND the new thing

Part 4: Reading the Diary

Task 4.1: Open the diary!

So we can read what's in the diary, we need to open it in a mode called **read**.

1. Find your `elif` statement that checks if you want to *read* the diary
2. Add a line of code that will open the diary in read mode.

This is a lot like when we opened the diary to write in it, but we're using a different mode.

Hint: Opening a File

For example you can open a file in *read mode* like this:

```
readmeFile = open("readme.txt", "r")
```

Task 4.2: Read the diary!

It's time to show the user what's inside the diary!

`print` out the contents of the diary to the user.

Hint: Reading a File

For example you can read the contents of a file like this:

```
contents = readmeFile.read()
```

Task 4.3: Close the diary!

Just like before, it's really important that we close the text file when we're finished with it!

`close` the diary.

☑ CHECKPOINT ☑

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

☐ When you say "2" it will print out everything that's written in the diary!

☐ Run it again to write something new in the diary. Then run and read the diary to see the updated contents.

Part 5: It's a SECRET Diary!

While

Task 5.1: Let's make a password

It's way too easy to get to the secret diary! We have no security! Let's add some in with a password.

1. Add a variable called **`password`** at the start of your code.
2. Make it equal to a fake password like "I love GPN"

Task 5.2: What's the password?

Let's ask the user if they know what the secret password is!

1. Find the point after the user gives their name, but before we ask what they want to do with the diary.
2. Use **`input`** to ask the user "what is the secret password? ". Store their answer in a variable called **`guessed_password`**.

Task 5.3: Guessing Game

The user might not get the password right on the first try. They might have made a typo! Let's give them some chances to get the password right.

1. Create a `while` loop that keeps going while the **`guessed_password`** does NOT (use !=) match the **`password`**

2. Inside the `while` loop, ask the user to guess again for the password! Store the user's answer in the same **`variable`** as before (**`guessed_password`**)..

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? ")
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to the extensions:

- ☐ You have a password.
- ☐ You keep asking the user for the password until they get it right.
- ☐ Once they get the password right they can use the diary!

★ BONUS 5.4: Preventing too many guesses

Right now, we keep asking the user what the **`password`** is until they get it right! This means that the user can keep guessing forever.

1. Create a variable called **`wrong`** just above the `while` loop and set it to `0`.

2. In the `while` loop, add 1 to **`wrong`** before they get to guess again so that we can count how many times they get the password wrong.

3. Add an `if` statement at the top of your `while` loop, that checks if **`wrong`** is equal to 3.

4. Inside that `if` statement, `print` something to let the user know that they have guessed wrong too many times and then make the program stop using **`exit()`**

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

Extension 6: It's a trap!

Sometimes siblings like to get in your stuff, so we're going to set a trap for anyone trying to sneak into our secret diary. Let's make a decoy password that shows them a fake diary if they enter it

Task 6.1: Decoy Password

In the same place that you set **`password`**, make a new variable called **`decoy_password`** and set it to something else that you think your friends or siblings might guess

Task 6.2: Let them through!

If they guess the **`decoy_password`** while they are trying to get into the secret diary, we want to let them past the while loop.

Add an `and` condition to the while loop so that it keeps looping if the **`guessed_password`** isn't the real password and the **`guessed_password`** isn't the decoy

Task 6.3: Fake news!

We need to make sure that if they guessed the **`decoy_password`**, they don't get into our real diary and get a fake diary instead!

1. Create an `if` statement after the `while` loop that checks if the **`guessed_password`** is the same as the **`decoy_password`**

2. Inside the `if` statement, print out whatever you want the fake diary to be! For example:

```
Here's the secret diary (I hope you can read gibberish!)
aghioahgiagipsjgij sagagieja iahgoaihg aoireioaghoiag io
wiohga aowihgaiog aohireg hai ogaa oiarghag iroagoi aoihgri
oaehr iohaihagh oiaegroha oaihgehaeo arheioargh oihrg haeo
```

Task 6.4: Could the real Secret Diary please step forward?

We still want whoever puts in the real password to have access to the diary

Add an `else` after the `if` that we just wrote and put all of the code below inside of it so that it only runs when the password isn't the decoy.

Hint: Indenting multiple lines

It can be a little annoying when you have to add a tab to the front of a lot of lines. In IDLE you can select all of the lines that you want to indent (that means, add a space in front of) and press the TAB key once and it will move all of the lines you had selected! Aren't computers cool?

☑ CHECKPOINT ☑

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

- ☐ You have a decoy password
- ☐ When the user puts in the decoy password they see the fake diary
- ☐ When the user puts in the real password they can still get the real diary

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: 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 9: Make it Funnier!

Task 9.1: More security, more!

Think about jokes and tricks to add to your chatbot when the user guesses the decoy password. Some ideas are below! If you haven't done extension 6 yet, do that first and come back.

A Magic 8 Ball!

Trick imposters into thinking that your secret is a magic 8 ball. Distract the imposter with your 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. 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 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 2 special tricks to print your art:

- triple quotes to `print` things on multiple lines using one `print` statement.
- Put the letter r just before your multiline string. This makes it a **raw** string. *Normal strings do special things with characters like `"\"`. We don't want that.*

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

Extension 10: Enc🍊ding in Em🍊ji

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 10.1: Apple-ify your message!

We'll start by replacing any letter "a" with an apple emoji.

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.)

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

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

3. 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")`

4. 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 10.2: De-apple-ify your diary!

When we want to read the diary again we'll need to put the letters back in, and remove the emoji!

1. Go to the place in your code after you read the content of the diary.

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)*

3. 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")`

4. 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 10.3: More emoji is more secret!

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

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