Girls' Programming Network

Password Cracker!

Workbook 2

In this workbook, you will learn about rainbow tables and how they can be used to guess other people's passwords!

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

This workbook and related materials were created by tutors at:

Sydney, Perth and Canberra

Girls' Programming Network

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

Writers

Alex McCulloch
Renee Noble
Caitlin Shaw
Taylah Griffiths

Testers

Manou Rosenberg

☑ CHECKPOINT ☑

You should only start working on this booklet if:

☐ You have completed Workbook 1

Part 0: Setting up

How can hackers find out your passwords if they are **hashed**? With **rainbow tables**!

Rainbow tables are used to find **commonly used passwords**.

For this workbook, you will build a rainbow table by hashing common passwords, and use it to try to hack into someone else's account!

Task 0.1: Making room for new code

Let's get started by making space for our new code!

Create a new .py file and call it **"rainbow.py"**

Task 0.2: Getting the files

We need to add some data files to our project

1. Download the files called **"accounts.txt"** and **"common-passwords.txt"** to your computer. You will find them on the website for today's project.

2. In your **online-python.com** project, click on the "open file from disk" 📂icon at the top of the page and upload the two **.txt** files.

☑ CHECKPOINT ☑

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

- ☐ You should have a file called rainbow.py
- ☐ You have 2 .txt files in your project
- ☐ Run your file with F5 key and it does nothing!!

For Loops and Files

Part 1: Reading in Files

Task 1.2: Loop through the file line by line

Let's use a `for` loop to iterate through each line in the **common-passwords.txt** file.

Hint

If we had a file called "cats.txt", we could open and loop through it like this:

```
for line in open("cats.txt"):
  print(line)
```

This opens the file and makes a for loop so that we can look through each line of the file!

Note: We can only open files that are located in the same folder as our code. Otherwise, the computer won't know where to look for the file!

Task 1.2: Strip that password!

We can tidy up our line variable by removing all the whitespace from the beginning and end of it. This will help us generate hashes properly.

While we're at it, let's store each tidied line in a new local variable called `password`.

In your for loop, create the `password` variable and use `.strip()` on `line`.

Hint

If I wanted to strip the beginning and ending of a variable `space` I would:

```
space =  "      GPN is great!      "
spaceStripped = space.strip()
```

Result (notice there aren't big spaces before and after the words):

```
space = "GPN is great!"
```

Task 1.3: Print that password!

Print out `password` inside the for loop to make sure that it's working correctly

★ Bonus 1.4: Extra Passwords? ★

Add additional passwords in the text file. Remember **not** to use your actual passwords, that wouldn't be good security, now would it!

☑ CHECKPOINT ☑

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

- ☐ You have created a for loop, to loop through each of the passwords in the txt file.
- ☐ You have stripped the passwords of all whitespace
- ☐ You have printed out all the passwords!
- ☐ Run your code!

Part 2: Hashing Passwords

Task 2.1: Preparing for hashing

Now what we want to do is encode each of the passwords.

Store the encoded password in a new variable called **`password_encoded.`** This code should be in the for loop that we made in the last part!

Hint

To encode a variable and save it to a new variable:

```
new = old.encode()
```

Task 2.2: Importing hashlib

We are going to use the **MD5 hash function** to hash our passwords. This function is included in the **`hashlib`** library.

Import **`hashlib`** so we have access to the **MD5 hash function**.

Hint

We can import this module by writing:

```
import hashlib
```

Remember to write import statements at the **top of the program**, so we can access the libraries later on!

Task 2.3: MD5 Hash that password!

Now hash the password and store it as a new variable called **`password_hash`.**

Hashing the password completely will take a few steps.
This is because the hash function only outputs a raw hash. If we're going to use it in our code, it needs to be digested and converted into a string format.

Use both **`str()`** and **`.digest()`** to process **`password_hash`**.

Hint

The **`md5()`** hash function can be tricky to use because it has a special syntax:

```
new = hashlib.md5(old)
```

This syntax tells the computer we are using the **`md5()`** function from within the **`hashlib`** library.

You can process a raw hash with:

```
new = str(new.digest())
```

Task 2.4: A rainbow?

We can finally build our rainbow table!

Go back to before the for loop and create an empty dictionary called **`rainbow`.**

It's important the dictionary is located **before** the for loop. This lets us access it from within the loop.

Hint

You can make an empty dictionary like this:

```
empty = {}
```

Task 2.5: Add to the rainbow!

Now add the plaintext original `password` as a **value** of `rainbow` with `password_hash` as the **key**.

Hint

To add a key-value pair to an existing dictionary, you can write:

```
dictionary_name["key"] = "value"
```

Task 2.6: Print the rainbow!

After the for loop we've just finished, **print** the dictionary called **`rainbow`.**

Delete the other **print** statement that's inside the for loop.

☑ CHECKPOINT ☑

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

- ☐ You have imported hashlib
- ☐ You have created a dictionary called "rainbow"
- ☐ You have encoded the passwords
- ☐ You have hashed and digested the encoded passwords
- ☐ You have added each hashed password to the dictionary
- ☐ You have printed the dictionary

Lists

Part 3: Do They Match?

In this section we are going to be reading in some accounts of other people to see if they match the password in our rainbow table!

Task 3.1: A new for loop

Create a new for loop to read through the other .txt file called **"accounts.txt"**.
This text file includes account names and their hashed passwords.

Task 3.2: Cleaning up the lines

Like in Task 1.3, we want to strip down the information we're reading in each line.
Strip down each line using the `.strip()` method, stored in a variable called `line`.

Task 3.3: Splitting out attributes

Next, use the `.split()` method to split the account name away from the account's hashed password at the comma.
Store this in a list called `account`.

Hint

If I wanted to split the variable `gpn = "gpn,is,fun"` at the comma I would write:

```
My_list = gpn.split(",")
```

This is what would be in `My_list`:

```
['gpn', 'is', 'fun']
```

Task 3.4: Assigning account name

In each line of the file, the first part is the account name and the second is the hashed password.

Make a new variable called **`name`** and store the first thing in the **`account`** list.

Hint

The first item in a list is always stored at index 0.

Task 3.5: Assigning account hash

Similar to above, we want to assign the account's hashed password to a variable called **`password_hash`.**
The account's hashed password is stored as the second index in our list called **`account`**.

Task 3.6: Print out the name

Now use an **if statement** to check if **`password_hash`** is in our **`rainbow`** dictionary.
If it is, print out the name of the account

Hint

To check if something is in a dictionary we can use code like this:

```
phone_numbers = {"Alex", "123", "Renee", "456"}
if "Renee" in phone_numbers:
  print("I can call Renee!")
```

Task 3.7: Print out the password

Inside the if statement also print out the password of the account.

Hint

To get something out of a dictionary we write it like this:

```
phone_numbers = {"Alex", "123", "Renee", "456"}
print(phone_numbers["Renee"])
```

☑ CHECKPOINT ☑

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

- ☐ You created a new for loop to read through **accounts.txt**
- ☐ You split each line and placed the attributes in a list
- ☐ You put account name and hashed password in new variables
- ☐ You printed out any matched passwords

Part 4: Finding Secrets

Task 4.1: Secrets!

Using the accounts and passwords you found before, go to the following link to find secrets on the website!
`https://girls-programming-network.github.io/meme-exchange/`

★ Bonus 1.4: Grow the rainbow! ★

How can we hack into more accounts? Let's come up with more passwords to add to the rainbow table!

Try adding modified versions of some of the common passwords. You can use common **substitutions**, like swapping "s" with 5 or "a" with @.

Hint

Go back to **Part 2** to refresh on hashing and adding things to dictionaries.

More examples of **common substitutions:**

- "E" with 3
- "L" with 1
- "O" with 0
- "!" on the end of the password
- Capital letter at the beginning

You can replace a letter in a word like this:

```
original = "password"
new = original.replace("s", "5")
print(new)

>> pa55word
```