



## 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!*

# TUTORS ONLY





**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!

1. Create a new folder called "**rainbow\_tables**"
2. Create a new .py file, call it "**rainbow.py**" and save it inside "**rainbow\_tables**".

## Task 0.2: Getting the files

Make sure you download the files called "**accounts.txt**" and "**common-passwords.txt**", and add them to the folder we created above.

## ✔ 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 a folder called rainbow\_tables
- You have 2 .txt files in your folder
- Run your file with F5 key and it does nothing!!



# Part 1: Reading in Files

## Task 1.1: 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 by using `.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:

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



### Task 1.3: Print that password!

Print out `password` in each iteration of the loop.

### ★ 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!

### TUTOR TIPS

The code should look like this (no bonuses):

```
for line in open("common-passwords.txt"):
    password = line.strip()
    print(password)
```



## 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.

In python, some libraries are stored outside the **IDE** (Integrated Development Environment), so we need to import them.

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 plain 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["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





## TUTOR TIPS

The code should look like this (no bonuses):

```
import hashlib

rainbow = {}

for line in open("common-passwords.txt"):
    password = line.strip()
    password_encoded = password.encode()
    password_hash = hashlib.md5(password_encoded)
    password_hash = str(password_hash.digest())
    rainbow[password_hash] = password

print(rainbow)
```



## 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: Splitting out attributes

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

Next, access the `account` list and assign the account name to a new variable called `name`. The account name is stored as the first index in our 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 password

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 and the matching password.

### 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



## TUTOR TIPS

The code should look like this (no bonuses):

```
import hashlib

rainbow = {}

for line in open("common-passwords.txt"):
    password = line.strip()
    password_encoded = password.encode()
    password_hash = hashlib.md5(password_encoded)
    password_hash = str(password_hash.digest())
    rainbow[password_hash] = password

print(rainbow)

for line in open("accounts.txt"):
    line = line.strip()
    account = line.split(",")
    name = account[0]
    password_hash = account[1]
    if password_hash in rainbow:
        print(name)
        print(rainbow[password_hash])
```

# 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 4.2: 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

## TUTOR TIPS

### List of some accounts and passwords (no bonuses):

Anna	Christina	Amy
Alicia	Sean	Victoria
Timothy	Holly	Jesse
Shannon	Erica	Melissa
Rebecca	Kristina	Eric
Cody	Kenneth	Zachary
Megan	Patrick	Erin
Kathryn	Kevin	Meghan
Ryan	David : 12345	Jessica : abc123
Antonio	Sarah	Alexander
Justin	Paul	Mary
Chad	Jeffrey	Gregory

Rachel  
Michelle  
Joseph  
Laura  
Benjamin  
Allison  
Katherine  
Christopher : soccer  
Robert  
Shawn  
Cynthia  
Nicole  
Jacqueline  
Michael : thunder  
Tyler  
Kristen  
Jose  
Jenna  
Jacob  
Marcus  
Juan  
April  
Danielle  
Cassandra  
Jennifer : qazwsx  
Crystal  
Kimberly  
Bryan  
Brett  
Ashley  
Jared  
Bradley  
Kyle  
Brian  
Keith  
Catherine  
John  
Kelly  
Jason  
Nicholas  
Edward  
Tiffany  
Richard  
Mark

Peter  
Ronald  
Stephen  
Brittany  
Sara  
Maria  
Scott  
Charles  
Monica  
Brandon  
Adam  
Angela  
Amanda : ginger  
Aaron  
Daniel : password  
Andrea  
Donald  
James  
Thomas  
George  
Joshua : hunter  
Anthony  
Heather  
Julie  
Christine  
Lauren  
Jeremy  
Elizabeth  
Patricia  
Katie  
Lindsey  
William  
Vanessa  
Shane  
Lindsay  
Lisa  
Joel

Steven  
Nathan  
Krystal  
Emily  
Samuel  
Andrew  
Corey  
Natalie  
Phillip  
Dustin  
Matthew : pass  
Derek  
Courtney  
Tara  
Stephanie  
Jamie  
Travis  
Jordan  
Amber  
Samantha  
Kristin  
Jonathan