



Girls' Programming Network

Password Cracker

*In this workbook, you will learn how to encode
plaintext using a hash function and compare it with
a stored passphrase for authentication!*

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 and Perth



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
Rama Kaorma

Testers

Sheree Pudney
Manou Rosenberg

Part 0: Setting up

Task 0.1: Making a python file

1. Got to <https://www.online-python.com/>
2. **Delete** all the example code that's there

Task 0.2: Saving your work

1. **Click** the **share button** next to the run button about halfway down the page
2. It will give you a share link that looks something like this
<https://www.online-python.com/>——
3. **Write** the last bit after the / **here** _____

TUTOR TIPS

Students can save the file to the computer if they want to but it's much easier to record the unique url. This way it autosaves their work regularly.


Task 0.3: You've got a blank space, so write your name!

a main.py file will have been created for you!

1. At the top of the file use a comment to write your name!

Any line starting with # is a comment.

`# This is a comment`

2. Run your code using the  **Run** button. It **won't** do anything yet!

✔ CHECKPOINT ✔

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

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

TUTOR TIPS

The code should look like this (no bonuses):

```
# <the student's name>
```

Part 1: Welcome to Passphrases

A **passphrase** is a sentence that has meaning for you and therefore easier to remember than a password.

One example of a passphrase is: "The ship sails at midnight"

We use **passphrases** rather than **passwords** as they are longer than passwords and therefore more secure.

Task 1.1: Welcome to Passphrases

1. Let's make a variable called `correct` that stores a passphrase. This can be any sentence you like!

Hint

To create variable called `favourite` and store a string in it:

```
favourite = "Chocolate"
```

Task 1.2: What is the passphrase?

Let's guess what the passphrase is!

1. Use `input` to ask the user for their guess. **Store** their answer in a variable called `guess` so we can use it in our code!

```
What is the passphrase?
```

Hint

To find out someone's favourite ice-cream and store it in a variable called `favourite`

```
favourite = input("What is your favourite ice-cream? ")
```

Task 1.3: Let's see!

Now that we know the user's guess, let's print it and the correct answer out:

1. `print` out the **correct** passphrase
2. `print` out the user's **guess**.

Your program might look like when you run it:

```
What is the passphrase? My guess passphrase
The ship sails at midnight
My guess passphrase
```

Hint - Example

Remember to `print` the `guess` variable that you made in Task 1.2!

if we had stored a name, we could print it like this:

```
name = "Renee"
print(name)
```

✓ CHECKPOINT ✓

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

- ☐ Create a variable storing the passphrase
- ☐ Ask for the passphrase
- ☐ Print the correct passphrase
- ☐ Print the guessed passphrase
- ☐ Try running your code!

TUTOR TIPS

The code should look like this (no bonuses):

```
# <the student's name>

correct = "The ship sails at midnight"
guess = input("What is the passphrase? ")

print(correct)
print(guess)
```



Part 2: Is the guess correct?

If
Statements

Task 2.1: Check if they have guessed correctly!

1. Use an **if** statement to tell the user whether they have made the right guess.
2. Welcome the user if they got it right:

Your program should look like this when you enter the correct password.

```
What is the passphrase? The ship sails at midnight
The ship sails at midnight
The ship sails at midnight
Welcome to the club!
```

(we'll get rid of the parts in your code where we print out the guess and the correct answer soon)

Hint - Example

In the **if** statement, compare the user's guess with the passphrase you chose. Don't forget to use `==` .

To check if someone guessed my favourite fruit

```
guess= "apple"
if guess == "banana":
    print("I love bananas!")
```

TUTOR TIPS

Some students may use a single equal sign to test for equality. Make sure they are using double equals!

Task 2.2: And if they got it wrong!

1. Under your **if** statement, add an **else** statement for when they guess incorrectly.
2. **Tell them to go away** if they have guessed wrong, like below:

```
What is the passphrase? At midnight the ship sails
The ship sails at midnight
At midnight the ship sails
Go away!
```

Hint - Example

This is what an **if** and **else** statement looks like!

```
guess= "apple"
if guess == "banana":
    print("I love bananas!")
else:
    print("I don't like that fruit")
```

Task 2.3: Stop printing

Now that we have our **if** and **else** statements, we don't need to print out the `correct` and `guess` variables anymore.

1. Delete those two `print` lines that display the correct answer and the guess. (or you can comment them out)

Hint - Example

To comment out a line of code you can add a `#` like this:

```
# print("something")
```

✓ CHECKPOINT ✓

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

- ☐ Welcome them if they got the passphrase correct
- ☐ Tell them to go away if they are wrong
- ☐ Run your code and test different guesses
- ☐ Your code doesn't print out the guess or correct passphrases

TUTOR TIPS

The code should look like this (no bonuses):

```
# <the student's name>

correct = "The ship sails at midnight"
guess = input("What is the passphrase? ")

if guess == correct:
    print("Welcome to the club!")
else:
    print("Go away!")
```

Part 3: What is Hashing?

Task 3.1: Hash a word by hand (no code for this part!)

First hash function

Replace each letter with its place in the alphabet:

G P N

Now add the numbers together:

Every time we follow this process for the acronym 'GPN', we will get the same number!

Now try hashing this word:

P N G

=

What number did you get? Is this a good thing? What happened here is called a collision!

Second hash function

Now try again but this time multiply the letter's place in the alphabet by its place in the word:

G P N

=

P N G

=

What do you notice?

Hint

You can use the table below to help find what number in the alphabet a letter is:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Task 3.2: Hash your name

Follow the same process as the second hash function and try to hash your name!

★ Bonus 1.4: Does Method 2 always work? ★

Can you find a word that collides with GPN using our second hash function?

abled	deface	lean	rug
acre	dime	leek	saiga
app	faked	mend	salad
base	foci	oct	sat
blam	foo	omaha	scala
bleed	heel	panda	tame
choca	hued	peach	teal
day	ion	psi	tho
deem	jaw	raged	toed

Hint

Collision is when 2 different words are hashed to the same number.

✔ CHECKPOINT ✔

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

- ☐ Found the hash of GPN and PNG for both methods
- ☐ Found the hash value of your name

TUTOR TIPS

First hash function

Replace each letter with its place in the alphabet:

$$\begin{array}{ccc} \mathbf{G} & \mathbf{P} & \mathbf{N} \\ \boxed{7} & \boxed{16} & \boxed{14} \\ & = & \\ & \boxed{37} & \end{array}$$

Now try hashing this word:

$$\begin{array}{ccc} \mathbf{P} & \mathbf{N} & \mathbf{G} \\ \boxed{16} & \boxed{14} & \boxed{7} \\ & = & \\ & \boxed{37} & \end{array}$$

What number did you get? Is this a good thing? What happened here is called a collision!
We got the same number for two different words. No it's not.

Second hash function

Now try again but this time multiply the letter's place in the alphabet by its place in the word:

$$\begin{array}{ccc} \mathbf{G} & \mathbf{P} & \mathbf{N} \\ \boxed{7 \times 1} & \boxed{16 \times 2} & \boxed{14 \times 3} \\ & = & \\ & \boxed{81} & \end{array}$$

$$\begin{array}{ccc} \mathbf{P} & \mathbf{N} & \mathbf{G} \\ \boxed{16 \times 1} & \boxed{14 \times 2} & \boxed{7 \times 3} \\ & = & \\ & \boxed{65} & \end{array}$$

What do you notice?

Different numbers for different words.

Part 4: Let's hash our code!

hashlib

Task 4.1: Import the hash library

First we need to import the python library that has pre-made hashing functions - this makes our life easier as we can use code that has been written by other people!

1. At the very top of your code add the following line:

```
import hashlib
```

This tells our code to look for and use the hashlib library.

Task 4.2: Encode our passphrase

1. After we set the `correct` variable, create a new variable called `correct_encoded` and set it to encode `correct` using the hashlib library.

Hint - Example

To encode a variable, you use the following code (replace `variable_name` with the variable you want to use):

```
name_encoded = name.encode()
```

TUTOR TIPS

Make sure they are hashing the `correct` variable and not the `guess`.

Task 4.3: Time to hash the passphrase!

1. Create a new variable called `correct_hashed`. Hash the `correct_encoded` variable and store it in `correct_hashed`.

Hint - Example

To hash a variable, you use the following code (replace `variable_name` with the variable you want to use):

```
name_hashed = hashlib.md5(name_encoded).digest()
```

Remember that hashlib is the library, md5 is the hashing algorithm and digest is what shows us what the hash is.



TUTOR TIPS

Make sure they are hashing the `correct_encoded` variable and not the `correct` variable or the `guess`.

Task 4.4: Print the hashed passphrase

Now that we have hashed the passphrase, let's `print` it so we can see what it looks like!

Once you run your code, copy the printed passphrase to a text file or add it as a comment in your code to save it for use in the next part.

Hint

The hash should look something like this:

```
b'\xcc\xd6R\x16\xb9\x1bP~lK\x01\x0e\x063\x10\xec'
```

Remember that you can save code as a comment like this:

```
#this is a comment
```

TUTOR TIPS

If the students get something like this as the hashed passphrase:
<md5 _hashlib.HASH object @ 0x0000017CEF74ABF0>
then they accidentally omitted `.digest()`

✓ CHECKPOINT ✓

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

- ☐ Encoded your passphrase
- ☐ Hashed your passphrase
- ☐ Printed the hashed passphrase
- ☐ Run your code!
- ☐ Copied the printed hash to a text file or comment to use later

TUTOR TIPS

The code should look like this (no bonuses):

```
# <the student's name>

import hashlib

correct = "The ship sails at midnight"
guess = input("What is the passphrase? ")

correct_encoded = correct.encode()
correct_hashed = hashlib.md5(correct_encoded).digest()
print(correct_hashed)

# b'\xcc\xd6R\x16\xb9\x1bP~lK\x01\x0e\x063\x10\xec'

if guess == correct:
    print("Welcome to the club!")
else:
    print("Go away!")
```

Note: the value of the hash will be different if they used a different passphrase

Part 5: Making our code secure.

If someone reads our code they can see the passphrase written there - that's very **insecure**!

To fix this **we will store the hash we calculated of our passphrase** only so that if someone sees our code they can't read the passphrase.

Task 5.1: Delete extra code

Let's **delete** or **comment** out the code we wrote in the last part.

1. Delete the line where we create the `correct_encoded` variable
2. Delete the line where we create the `correct_hashed` variable
3. Delete the line where we `print` the hashed value.

Task 5.2: Replace the string with a hash

Time to store the hash we calculated earlier as **our hashed passphrase**

1. **Delete** the variable `correct`
2. **Replace** it with a variable called `correct_hashed`.
3. **Store the hash you copied** in the previous part in `correct_hashed`.

Hint

Remember that the hash should look something like this:

```
correct_hashed = b'\xcc\xd6R\x16\xb9\x1bP~lK\x01\x0e\x063\x10\xec'
```

TUTOR TIPS

The code for Task 5.1 should change from

```
correct = "The ship sails at midnight"
```

to:

```
correct_hashed = b'\xcc\xd6R\x16\xb9\x1bP~lK\x01\x0e\x063\x10\xec'
```

Task 5.3: Encode and hash the guess

It's time to **encode** and **hash** our guess!

1. Create a new variable called `guess_encoded`. Store the encoded value of our guess! (**Put this** on the line just after you get the guess.)
2. On the next line create a new variable called `guess_hashed`. Store the hashed value of our `guess_encoded`.

Hint

If you've forgotten how to do this, have another look at Part 4!

Task 5.5: Compare the hashes

1. Change your if statement to compare the `guess_hashed` variable and `correct_hashed` variable instead of the `guess` and `hash` variables.

Make sure the if statement comes after all the hash code!

☑ CHECKPOINT ☑

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

- ☐ Remove the `correct` variable
- ☐ Encode the guess and store it in the variable `guess_encoded`
- ☐ Hash the guess and store it in the variable `guess_hashed`
- ☐ Change your `if` statement to compare hashes instead of strings
- ☐ Run your code!

TUTOR TIPS

The code should look like this (no bonuses):

```
# <the student's name>

import hashlib

correct_hash = b'\xcc\xd6R\x16\xb9\x1bP~lK\x01\x0e\x063\x10\xec'
guess = input("What is the passphrase? ")

guess_encoded = guess.encode()
guess_hash = hashlib.md5(guess_encoded).digest()

if guess_hash == correct_hash:
    print("Welcome to the club!")
else:
    print("Go away!")
```


Extension 6: Let's get Cracking!

Here is a list of the 10 most common passwords. However, we only have the hashes and forgot to write down what the plain password is! In this part, you will use your python program from parts 0 to 5 to figure what the plain text for each hash is.

Once you figure out a password you can try putting the username and password into the Meme Exchange site!

Plain text	Username	Hash
1234	James	b'\x81\xdc\x9b\xdbR\xd0M\xc2\x006\xdb\xd81>\xd0U'
ashley	Robert	b"\xad\xffD\xc5\x10/\xca'\x9f\xceY\xab\xfa6o\xee"
123456789	John	b'%\xf9\xe7\x942;E8\x85\xf5\x18\x1f\x1bbM\x0b'
freedom	Joseph	b'\xd5\xaa\x17)\xc8\xc2S\xe5\xd9\x17\xa5&HU\xea\xba8'
monkey	Andrew	b'\xd0v>\xda\xa9\xd9\xbd*\x95\x16(\x0e\x90D\xd8\x85'
michael	Ryan	b'\n\xcfE9\xa1K:\xa2}\xee\xba4\xcb\xdfn\x98\x9f'
11111111	Brandon	b'\x1b\xbd\x88d'\x82p\x15\xe5\xd6\x05\xedD%"Q'
Qazwsx	Jason	b'vA\x9cXs\r\x9f5\xdez\xc58\xc2\xfdg7'
starwars	Sarah	b'[\xad\xca\xfa7\x89\xd3\xd1\xd0\x97\x94\xd8\xfa0!\xfa4\x0f\x0e'
Password	Amber	b"_M\xcc;Z\xa7e\xd6\x1d\x83'\xde\xba8\x82\xcf\x99"

Possible passwords

Each of these hashes will match one of these plain text passwords:

monkey	11111111	qazwsx	ashley
password	freedom	michael	starwars
	123456789	1234	

Task 6.1: What is the password?

1. Go back to the website for today's workshop. On the website for your room, you should be able to find a text file called `account_info.txt` with the list of the hashes provided above for you to copy and paste into your python program for convenience.
2. Pick a hash from the list and change your `hashed_correct` variable to it. Then run the program and guess different options from the **possible passwords** above to find the correct plaintext password.
3. Once you figure out a username and password pair, put it into the Meme Exchange website.