

NAME: Kelvin Kimotho

LinkedIn: [kelvin kimotho](#)

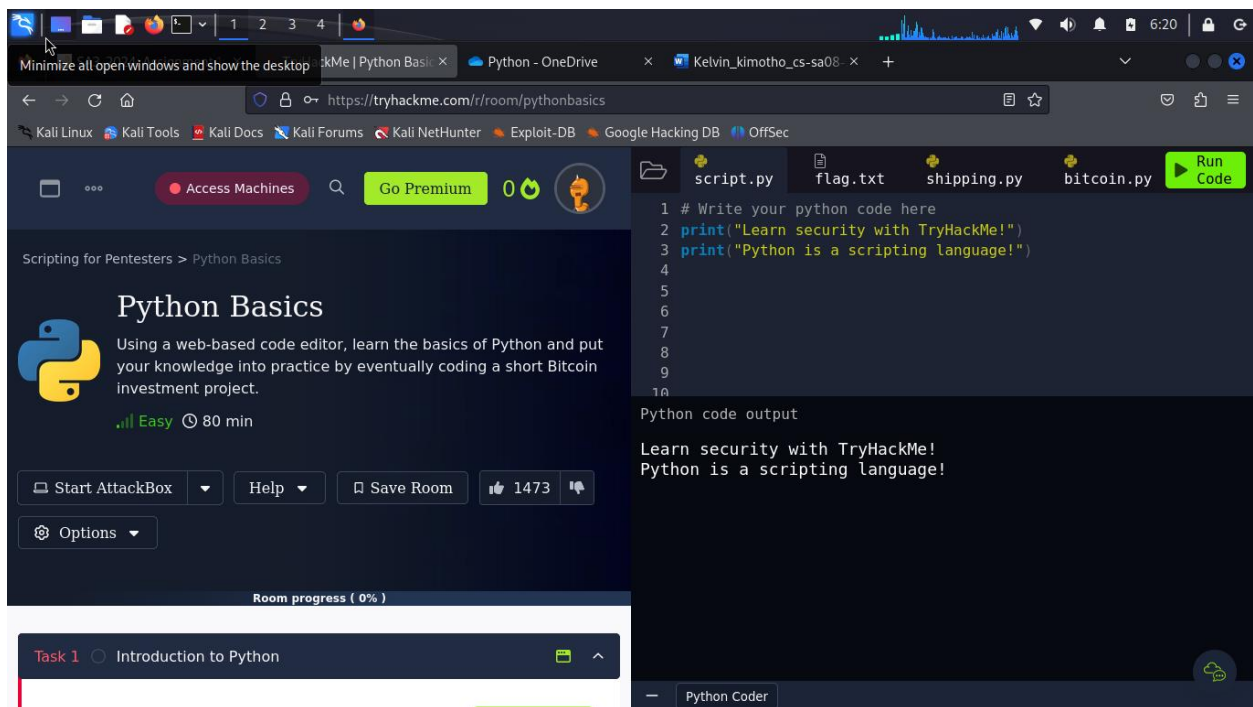
Here is my shareable link <https://tryhackme.com/p/Mr.kevin>

Introduction

Python is a scripting language. It's a programming language whose application range from web development, data science, machine learning. Our focus here mainly is to use python to craft tools that we can use maybe during penetration testing.

We use code editors when writing our scripts. E.g. Vscode, notepad, sublime text.

I tested the code editor provide by try hack me.



Hello World

This helped me get started. By writing a simple program that prints hello kelvin when run.

Below is the syntax:

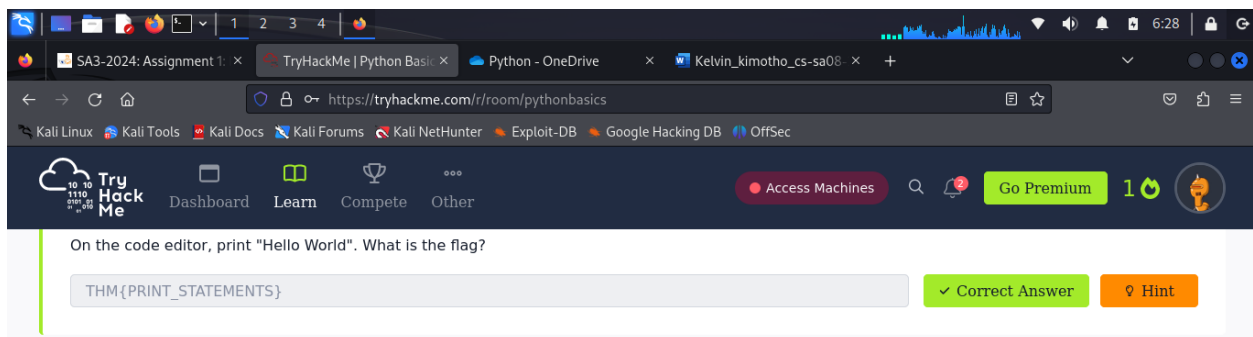
This is a comment

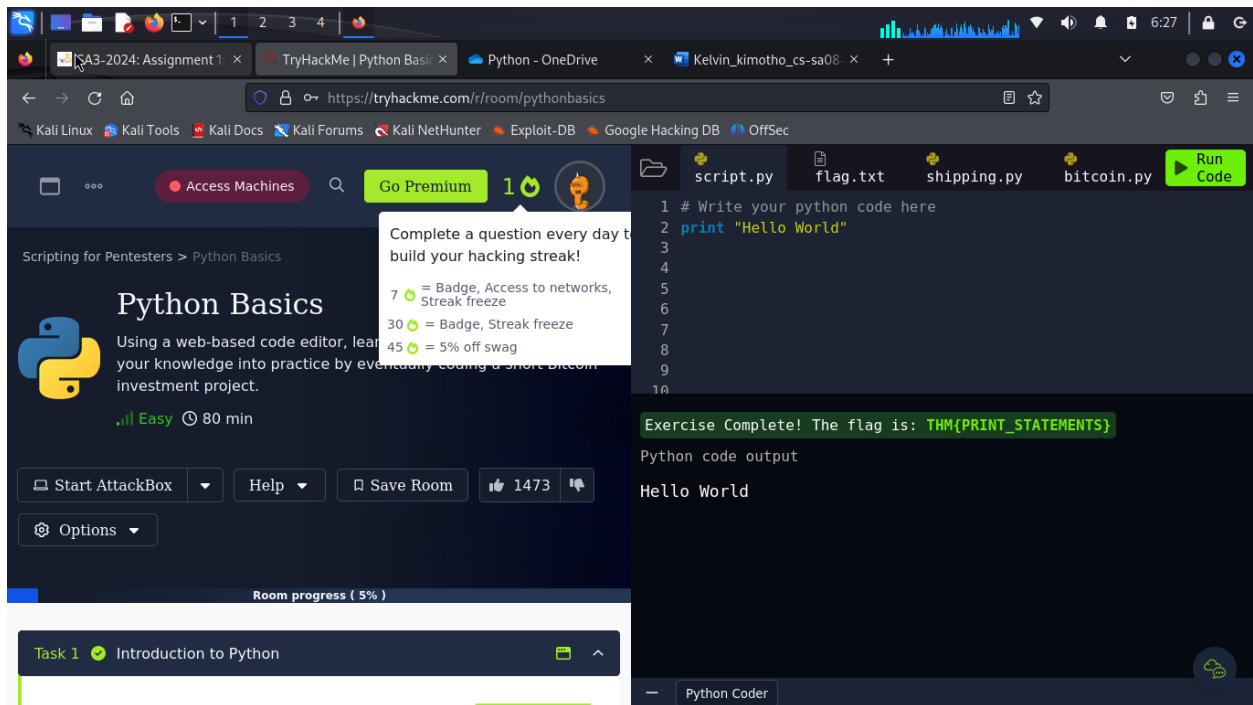
Print("Hello, kelvin")

- print() statements controls what appears on the screen.
- (), are paranthesis
- #, This symbol comments everything that comes after it and the code interpreter will ignore a commented line.
- "", are used around strings. We have to wrap strings using double quotes or even single quotes.

Question: On the code editor, print "Hello World". What is the flag?

Answer: THM{PRINT_STATEMENTS}





Mathematical Operators

- mathematical operators help us perform operations such as adding, subtracting, multiplying, and dividing; using Python.
- They include: + for addition, - for subtraction, * for multiplication, % for modulus, / for division and ** for exponential.
- Comparison operators are used to evaluate a program's condition at a particular state. They include,

Greater than >

Less than <

Equal to ==

Not Equal to !=

Greater than or equal to \geq

Less than or equal \leq

Question: In the code editor, print the result of $21 + 43$. What is the flag?

Answer: THM{ADDITION}

Question: Print the result of $142 - 52$. What is the flag?

Answer: THM{SUBTRACT}

Question: Print the result of $10 * 342$. What is the flag?

Answer: THM{MULTIPLICATION_PYTHON}

Question: Print the result of 5 squared. What is the flag?

Answer: THM{EXPONENT_POWER}

The screenshot shows a web browser window with the URL <https://tryhackme.com/r/room/pythonbasics>. The page displays a Python exercise with four questions and their corresponding answers. The answers are: THM{ADDITION}, THM{SUBTRACT}, THM{MULTIPLICATION_PYTHON}, and THM{EXPONENT_POWER}. A notification bubble says "Woop woop! Your answer is correct". The exercise is marked as complete with the flag THM{EXPONENT_POWER}. The Python code editor on the right shows the following code:

```
1 # Write your python code here
2 #print(21 + 43)
3 #print(142 - 52)
4 #print(10 * 342)
5 print(5**2)
6
7
8
9
10
```

The Python code output is 25.

Variables and Data Types

- Variables allow you to store and update data in a computer program. An example is, name=" kelvin", where name is a variable name and kelvin is the value.

Data Types define which type of data being stored in a variable. Variable types include.

- **String** - Used for combinations of characters, such as letters or symbols
- **Integer** - Whole numbers
- **Float** - Numbers that contain decimal points or for fractions
- **Boolean** - Used for data that is restricted to True or False options
- **List** - Series of different data types stored in a collection

Question: On another new line, print out the value of height. What is the flag that appears?

Answer: THM{VARIABLES}

The screenshot shows a web browser window with the URL <https://tryhackme.com/r/room/pythonbasics>. The page displays a Python exercise with the following instructions:

Answer the questions below

In the code editor, create a variable called height and set its initial value to 200.

No answer needed

On a new line, add 50 to the height variable.

No answer needed

On another new line, print out the value of height. What is the flag that appears?

THM{VARIABLES}

The code editor on the right shows the following Python code:

```
1
2 height=200
3 height=height+50
4 print(height)
5
6
7
8
9
10
```

The output of the Python code is shown as:

```
Python code output
250
```

The exercise is complete, and the flag is displayed as: **Exercise Complete! The flag is: THM{VARIABLES}**

Logical and Boolean Operators

- Logical operators allow assignment and comparisons to be made and are used in conditional testing. They include,

Equivalence `==` if `x == 5`

Less than	<	if x < 5
Less than or equal to	<=	if x <= 5
Greater than	>	if x > 5
Greater than or equal to	>=	If a>=b

- Boolean operators are used to connect and compare relationships between statements. They include,
- AND. (if x==5 AND y==10). Both conditions must be true for the entire thing to be true.
- OR. One or both conditions for the entire thing to be true. (if x==10 OR y==20) means that only one is required to be met for the program to continue.
- NOT. (if NOT y) means that It must not be y for the program to continue executing.

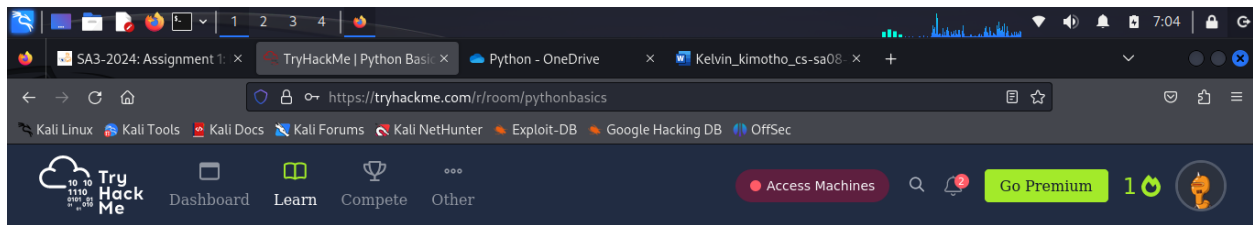
If statements

if statements allow programs to make decisions. They let a program chose a decision based on a condition.

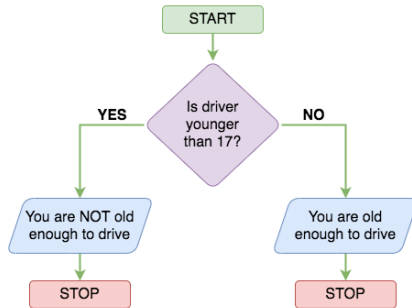
```
if age < 17:
    print('You are NOT old enough to drive')
else:
    print ('You are old enough to drive')
```

We need to follow all the rules in python. For Example, indentations are key since python do not use brackets like other languages.

A flowchart of hoe decisions are made in the above program.



- Note the indentation. Anything after the colon that is indented, is considered part of the if statement, which the program will execute.



The output when i run the code on the provided code editor.

```
1 age=17
2 if age < 17:
3     print('You are NOT old enough to drive')
4 else:
5     print('You are old enough to drive')
6
7
8
9
10
```

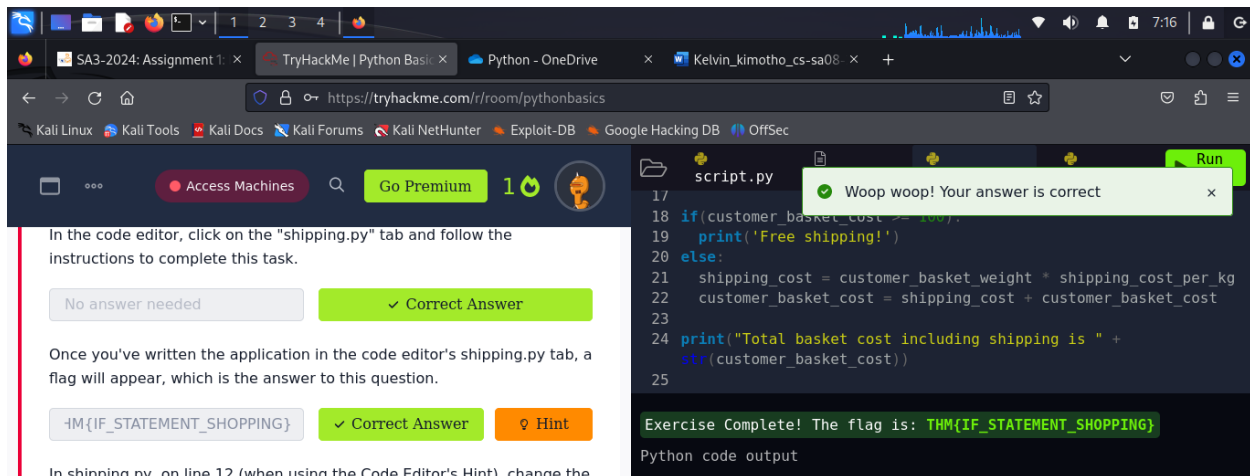
Python code output

You are old enough to drive

Python Coder

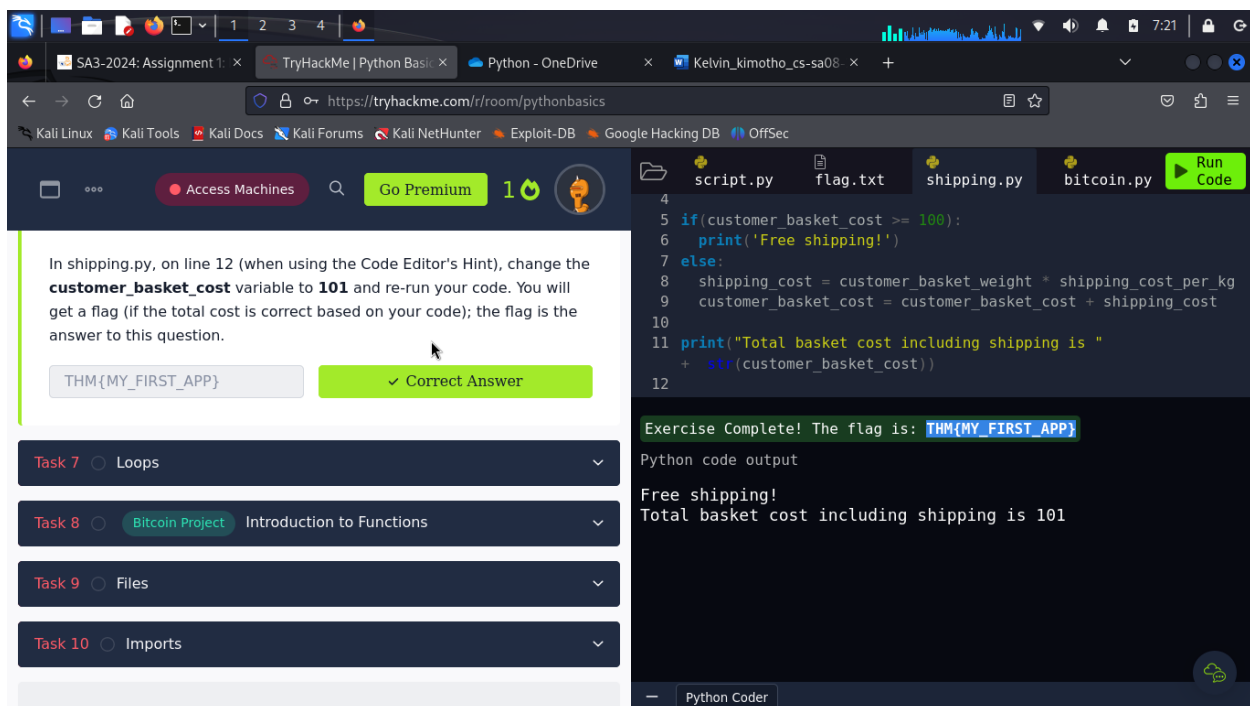
Question: Once you've written the application in the code editor's shipping.py tab, a flag will appear, which is the answer to this question.

Answer: THM{IF_STATEMENT_SHOPPING}



Question: In shipping.py, on line 12 (when using the Code Editor's Hint), change the **customer_basket_cost** variable to **101** and re-run your code. You will get a flag (if the total cost is correct based on your code); the flag is the answer to this question.

Answer: THM{MY_FIRST_APP}



Loops

loops allow programs to iterate and perform actions a number of times. There are two types of loops, for and while loops.

While Loops

Below is a structure of a while loop using an example.

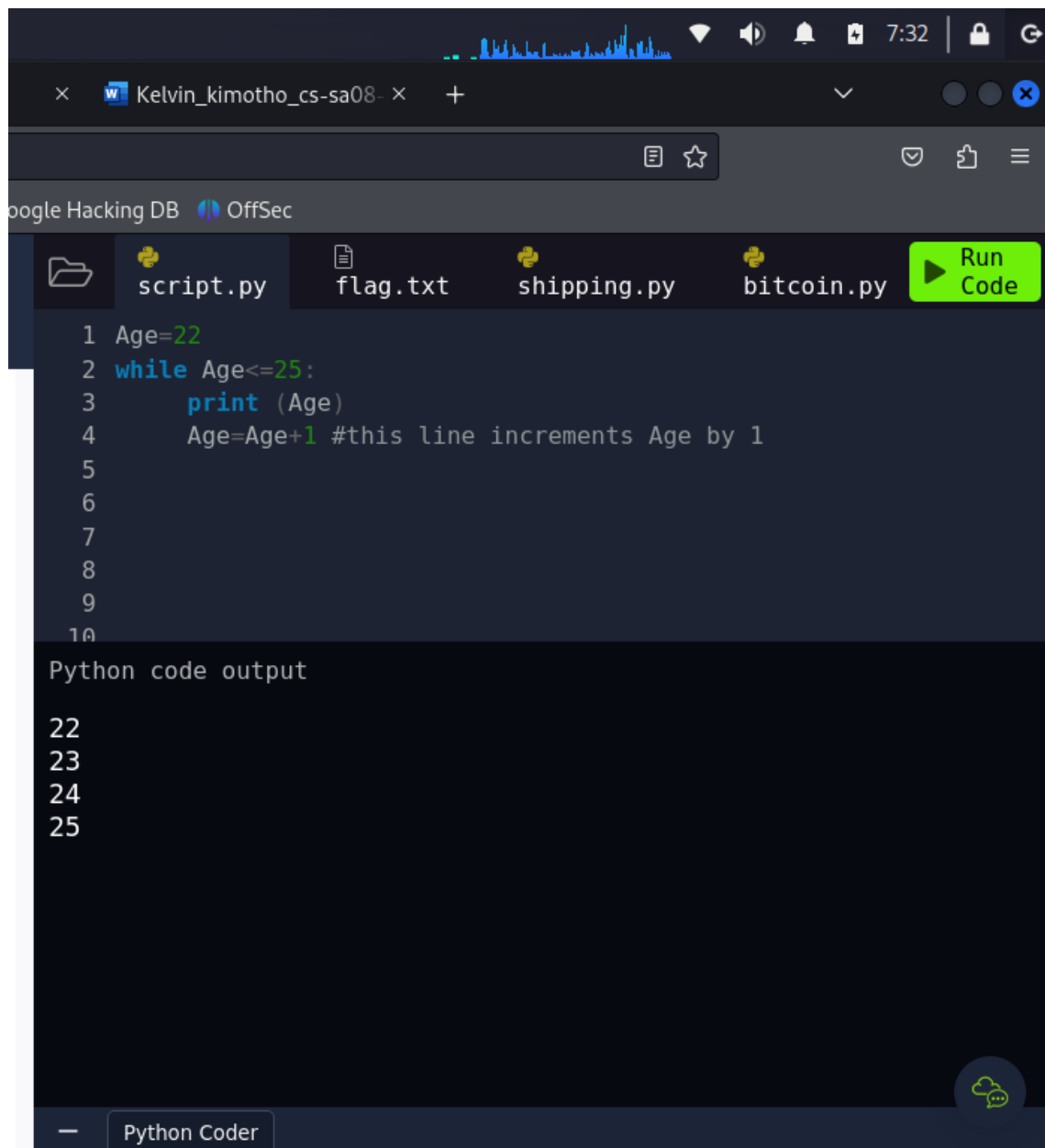
```
Age=22
```

```
While Age<=25:
```

```
    Print (Age) # prints the age as long as it meets the condition
```

```
    Age=Age+1 #this line increments Age by 1
```

I tested the program on the provided code editor.



The screenshot shows a web browser window with a dark theme. The address bar shows a URL starting with 'Kelvin_kimotho_cs-sa08-'. The page title is 'Google Hacking DB OffSec'. Below the title bar, there are tabs for 'script.py', 'flag.txt', 'shipping.py', and 'bitcoin.py'. A green 'Run Code' button is visible. The main editor area contains the following Python code:

```
1 Age=22
2 while Age<=25:
3     print (Age)
4     Age=Age+1 #this line increments Age by 1
5
6
7
8
9
10
```

Below the code editor, there is a section titled 'Python code output' which displays the results of the code execution:

```
22
23
24
25
```

At the bottom of the interface, there is a 'Python Coder' label and a small icon of a cloud with a speech bubble.

For Loops

A for loop is used to iterate over a sequence such as a list. The programing goes through a list of items and then prints them one after the other. My test example is.

```
1 students=["kelvin","jane","mike"]
2 for student in students:
3     print(student)
4
5
6
7
8
9
10
```

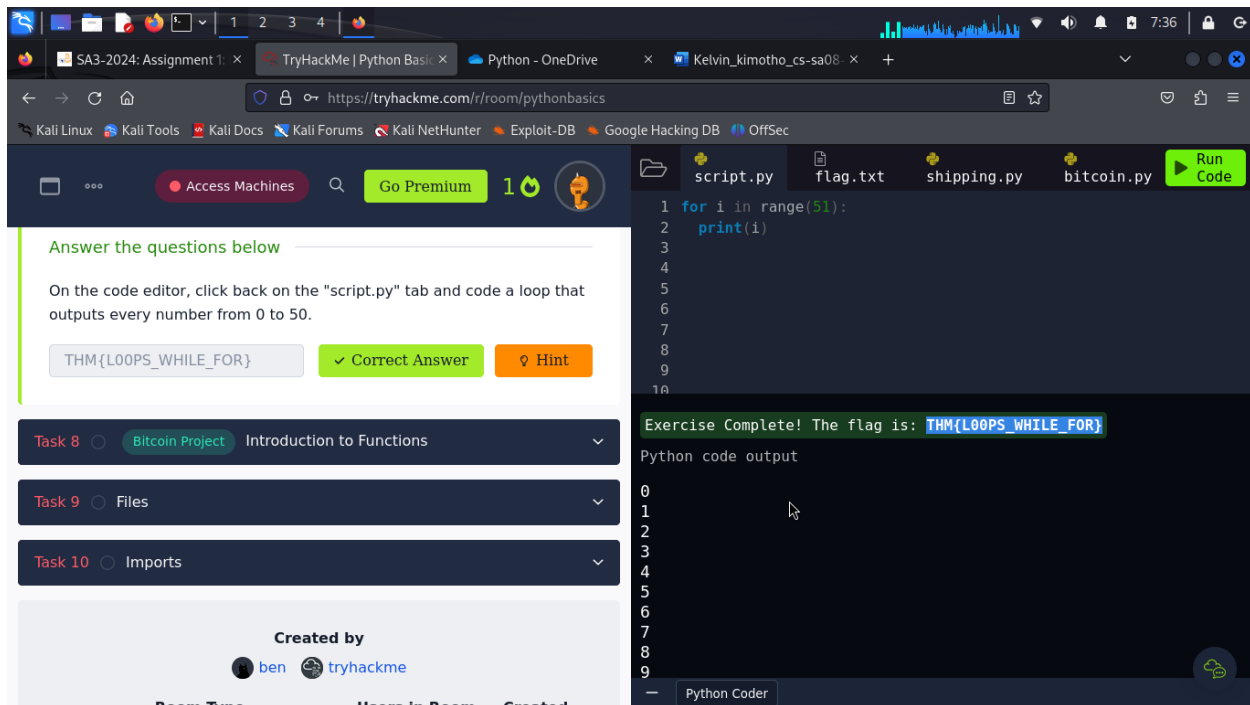
Python code output

```
kelvin
jane
mike
```

Python Coder

Question: On the code editor, click back on the "script.py" tab and code a loop that outputs every number from 0 to 50.

Answer: THM{L00PS_WHILE_FOR}



Introduction to Functions

- A function is a block of code that can be called at different places in your program.
- Functions helps avoid repetition. We can reuse a function thought the program or even export to be used in other programs.
- **def keyword** is use to define a function. For example,

```
def hello ():
```

```
    Print("Hello kelvin!")
```

- The above function prints "hello world" when called. We call a function by just typing its name. E.g hello()

I tested how the function works.

```
1 def hello():
2     print("hello kelvin")
3 hello() #function call
4
5
6
7
8
9
10
```

Python code output

hello kelvin

Python Coder

Question: You've invested in Bitcoin and want to write a program that tells you when the value of Bitcoin falls below a particular value in dollars.

In the code editor, click on the bitcoin.py tab. Write a function called **bitcoinToUSD** with two parameters: **bitcoin amount**, the amount of Bitcoin you own, and **bitcoin value usd**, the value of bitcoin in USD. The function should return usd_value, which is your bitcoin value in USD (to calculate this, in the function, you times bitcoin_amount variable by bitcoin_value_usd variable and return the value). Once you've written the bitcoinToUSD function, use it to calculate the

value of your Bitcoin in USD, and then create an if statement to determine if the value falls below \$30,000; if it does, output a message to alert you (via a print statement).

Answer: THM{BITCOIN_INVESTOR}

The screenshot shows a web browser window with the TryHackMe Python Basics exercise. The left sidebar contains the task description and a hint. The main area shows a Python code editor with the following code:

```
def bitcoinToUSD(bitcoin_amount, bitcoin_value_usd):  
    19  
    20     usd_value = bitcoin_amount * bitcoin_value_usd  
    21     return usd_value  
    22  
    23 investment_in_usd = bitcoinToUSD(investment_in_bitcoin,  
    bitcoin_to_usd)  
    24 if investment_in_usd <= 30000:  
    25     print("Investment below $30,000! SELL!")  
    26 else:  
    27     print("Investment above $30,000")
```

The output of the code is "Investment above \$30,000". The hint section shows the flag THM{BITCOIN_INVESTOR} and a "Correct Answer" button.

Files

- Python allows us to read and write into files.
- To read from a file we use a method called **open**. The format to open a file is `open(file_name,'action')`.

For example.

`F=open("filename","r").`

- Where **F** is the file handle, **r** means that we are reading from a file, **a** means that we are appending contents to the file while a **w** means that we want to overwrite whatever is in the file with new contents.

To read and print the file contents we.

- `Print(F.read())`, will print the file contents.

write() helps us write or append to a file.

close() helps us close a file after an operation.

Question: In the code editor, write Python code to read the flag.txt file. What is the flag in this file?

Answer: THM{F1LE_R3AD}

The screenshot shows a web browser with multiple tabs. The active tab is 'TryHackMe | Python Basics' with the URL 'https://tryhackme.com/r/room/pythonbasics'. Below the browser, there's a notification bar with 'Access Machines', 'Go Premium', and a 'Run' button. A green notification bubble says 'Woop woop! Your answer is correct'. The main content area shows a question: 'In the code editor, write Python code to read the flag.txt file. What is the flag in this file?'. Below the question is a text input field containing 'THM{F1LE_R3AD}' and a green 'Correct Answer' button. At the bottom, there's a 'Task 10' section with a dropdown menu set to 'Imports'. Below this is a table with information about the room.

Room Type	Users in Room	Created
Free Room. Anyone can deploy virtual machines in the room (without being	87,737	1147 days ago

On the right side, a code editor shows a Python script named 'script.py' with the following code:

```
1 flag_file=open('flag.txt')
2 print(flag_file.read())
3 flag_file.close() #closing the file after read operation
4
5
6
7
8
9
10
```

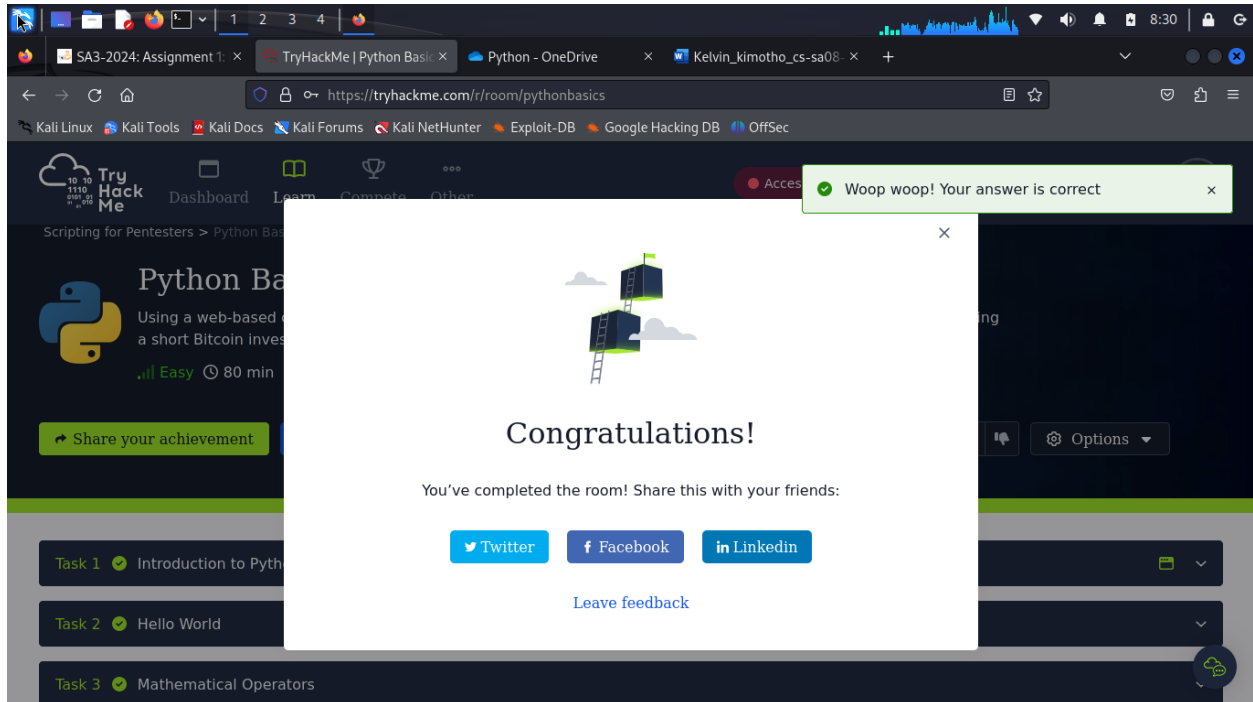
Below the code editor, the 'Python code output' section displays 'THM{F1LE_R3AD}'.

Imports

- Python gives us the ability to import libraries.
- Libraries are a collection of files that contain functions.
- By importing this library, we can use functions that are already written so we do not need to reinvent new functions.
- We use the **import** keyword followed by the library we want to import and use. Example, “import date”

We can install this library using Pip or pip3 python tools depending on the version of python we are running on our machines. E.g. “Pip3 install scrapy”

I completed the python scripting module.

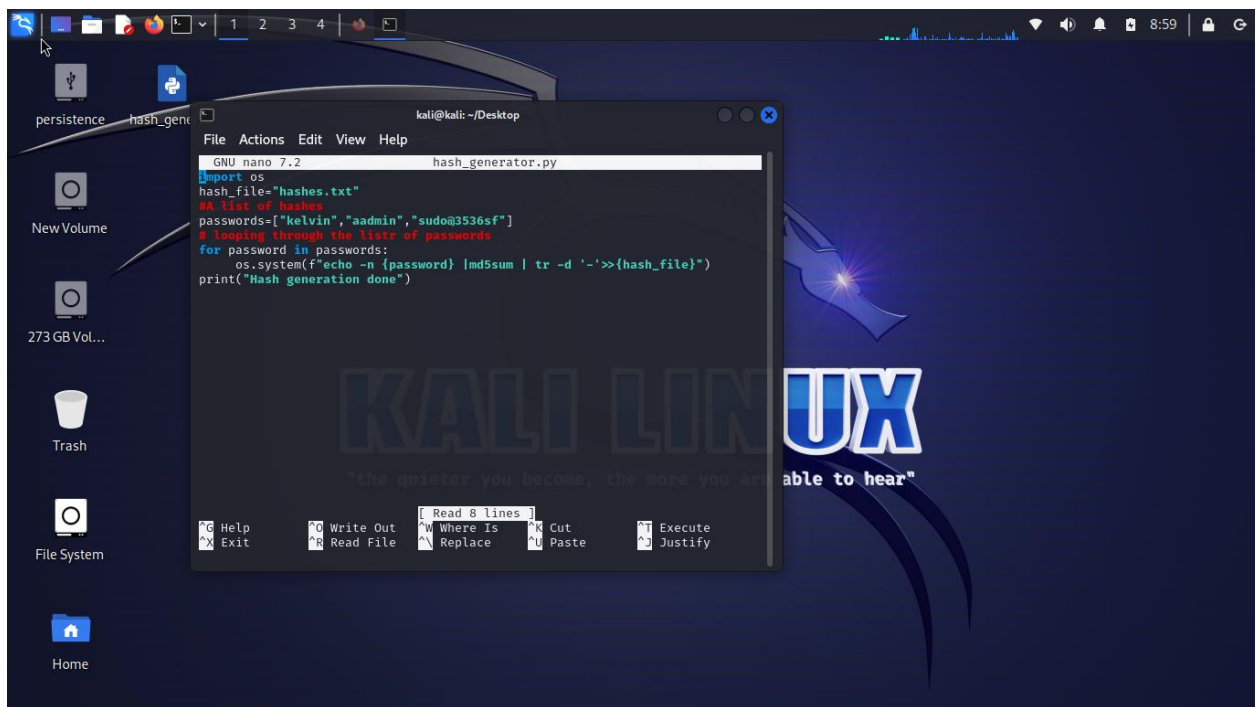


Conclusion

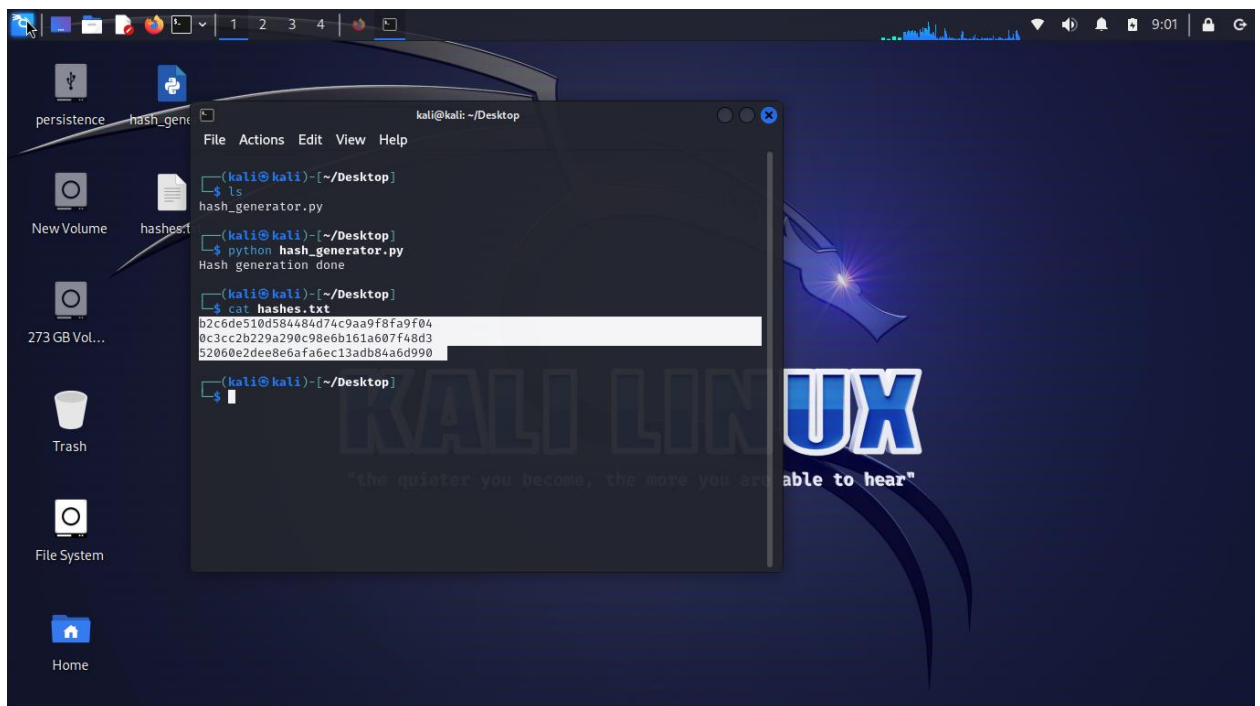
By completing python basics module, I can now craft simple programs to help me automate things for example in penetration testing work. For example, I might be having a list of password hashes I want to crack maybe using hashcat. Instead of passing one hash at a time, I can craft a simple python program to loop through the hashes passing each at a time to hashcat for cracking. I've had some experience working with python before during my course work at the university but now from a cybersecurity perspective, I can do a lot with python.

I did a simple python project on my machine. I generated md5hashes using a python program from a list of passwords.

- A program to generate md5 hashes for 3 passwords. Two weak passwords and a strong one.

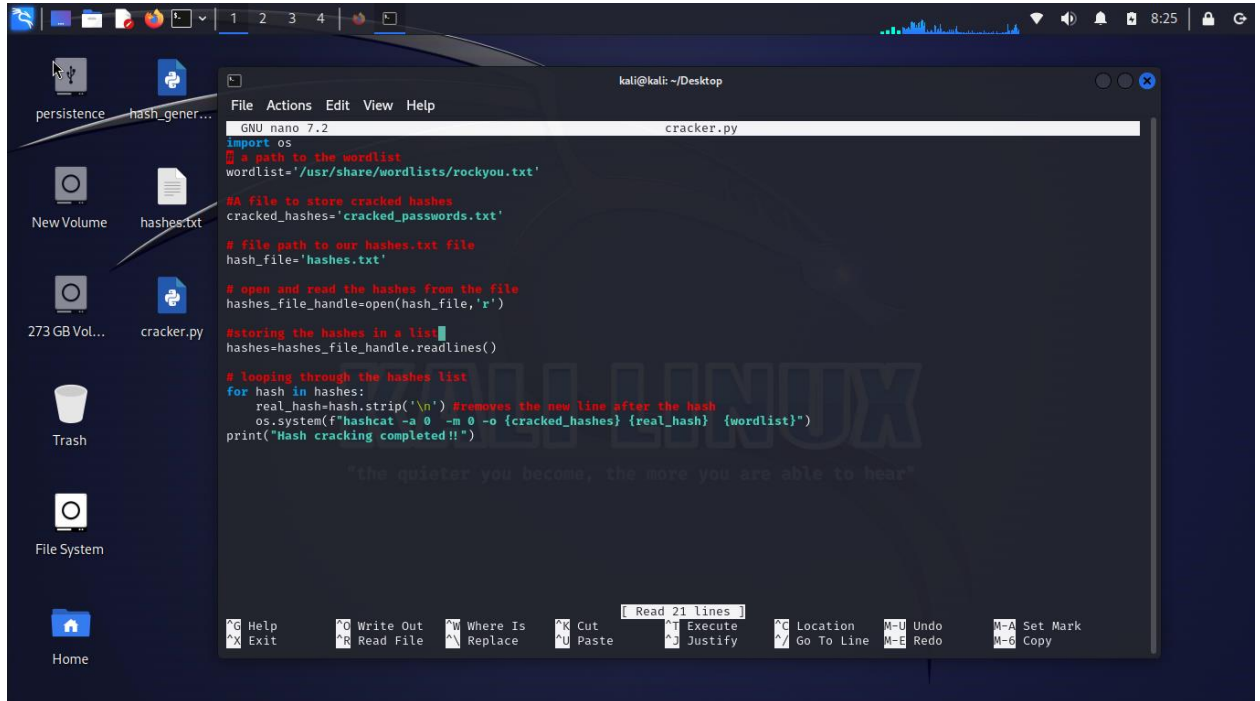


Below are the hashes to the passwords we passed into the hashing script.



I then created another python that read the hashes from the text file, iterated through them and passed them to hashcat for cracking then the output was saved in another text file.

#cracker.py



The screenshot shows a Kali Linux desktop environment. A terminal window titled 'kali@kali: ~/Desktop' is open, displaying the nano 7.2 text editor editing a file named 'cracker.py'. The code in the editor is as follows:

```
GNU nano 7.2 cracker.py
import os
# a path to the wordlist
wordlist="/usr/share/wordlists/rockyou.txt"

#A file to store cracked hashes
cracked_hashes='cracked_passwords.txt'

# file path to our hashes.txt file
hash_file='hashes.txt'

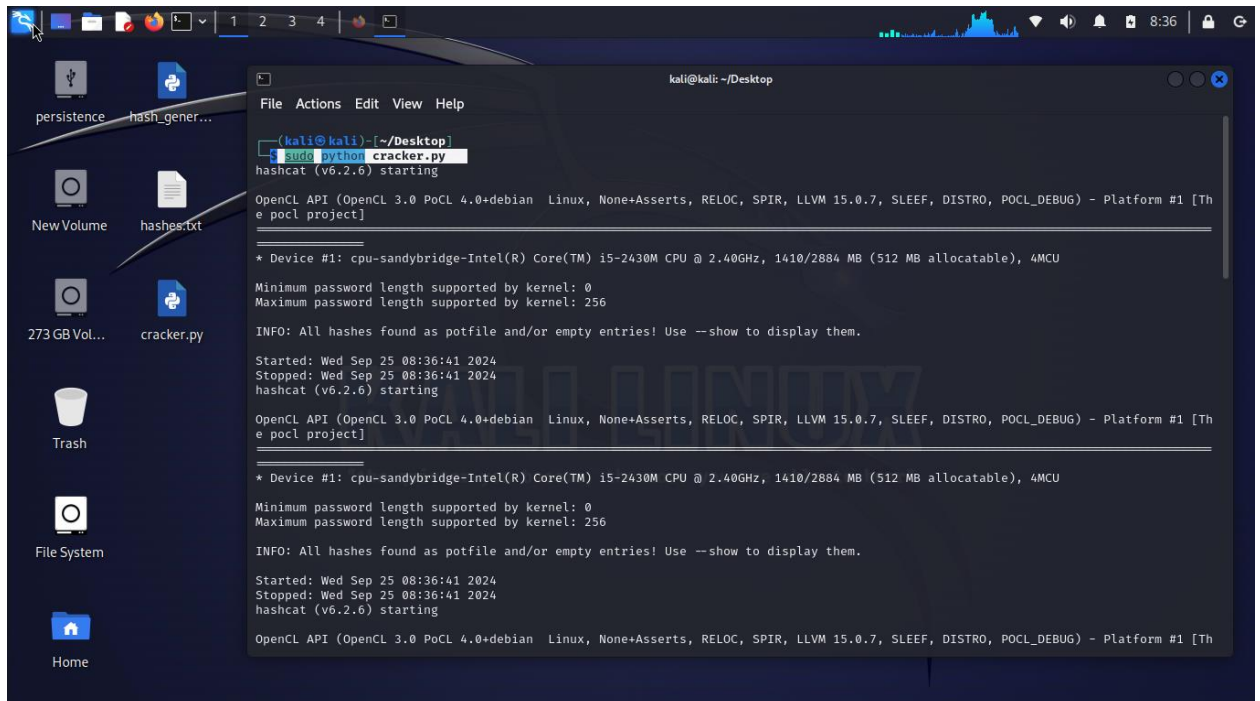
# open and read the hashes from the file
hashes_file_handle=open(hash_file,'r')

#storing the hashes in a list
hashes=hashes_file_handle.readlines()

# looping through the hashes list
for hash in hashes:
    real_hash=hash.strip('\n') #removes the new line after the hash
    os.system(f"hashcat -a 0 -m 0 -o {cracked_hashes} {real_hash} {wordlist}")
print("Hash cracking completed!!")
```

The desktop background features a large 'KALI LINUX' watermark. On the left sidebar, there are icons for 'persistence', 'hash_gener...', 'New Volume', 'hashes.txt', '273 GB Vol...', 'cracker.py', 'Trash', 'File System', and 'Home'.

Then i run the program.



The screenshot shows the same Kali Linux desktop environment. The terminal window now displays the output of running the 'cracker.py' script. The output includes the command being executed, the starting of hashcat, OpenCL API information, device details, and the execution of the hashcat command.

```
(kali@kali)~[/Desktop]
$ sudo python cracker.py
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 4.0+debian Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.7, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: cpu-sandybridge-Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz, 1410/2884 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Wed Sep 25 08:36:41 2024
Stopped: Wed Sep 25 08:36:41 2024
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 4.0+debian Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.7, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: cpu-sandybridge-Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz, 1410/2884 MB (512 MB allocatable), 4MCU

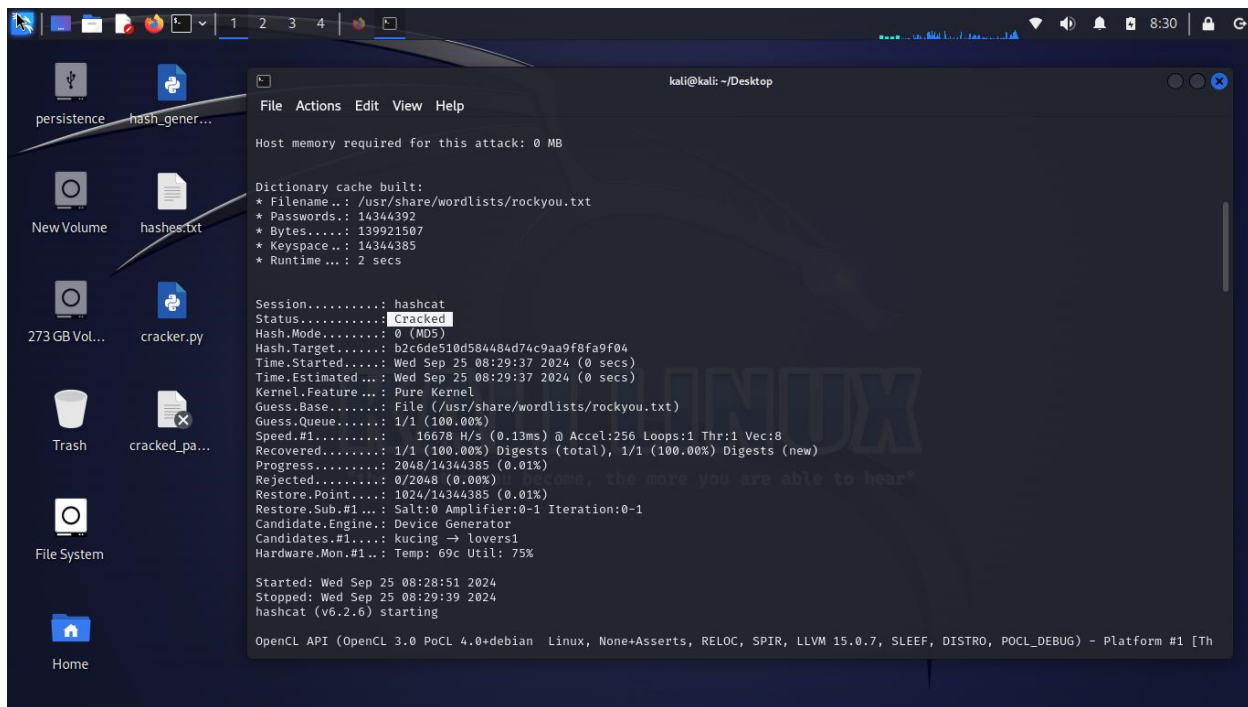
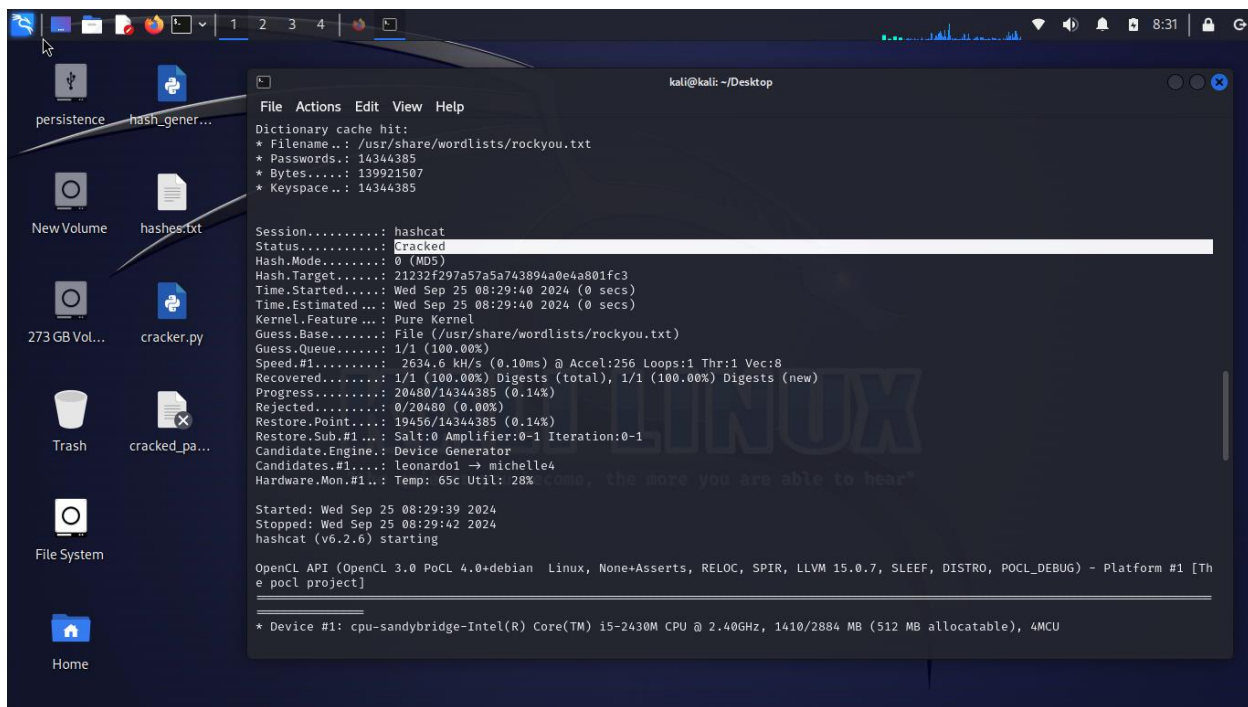
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

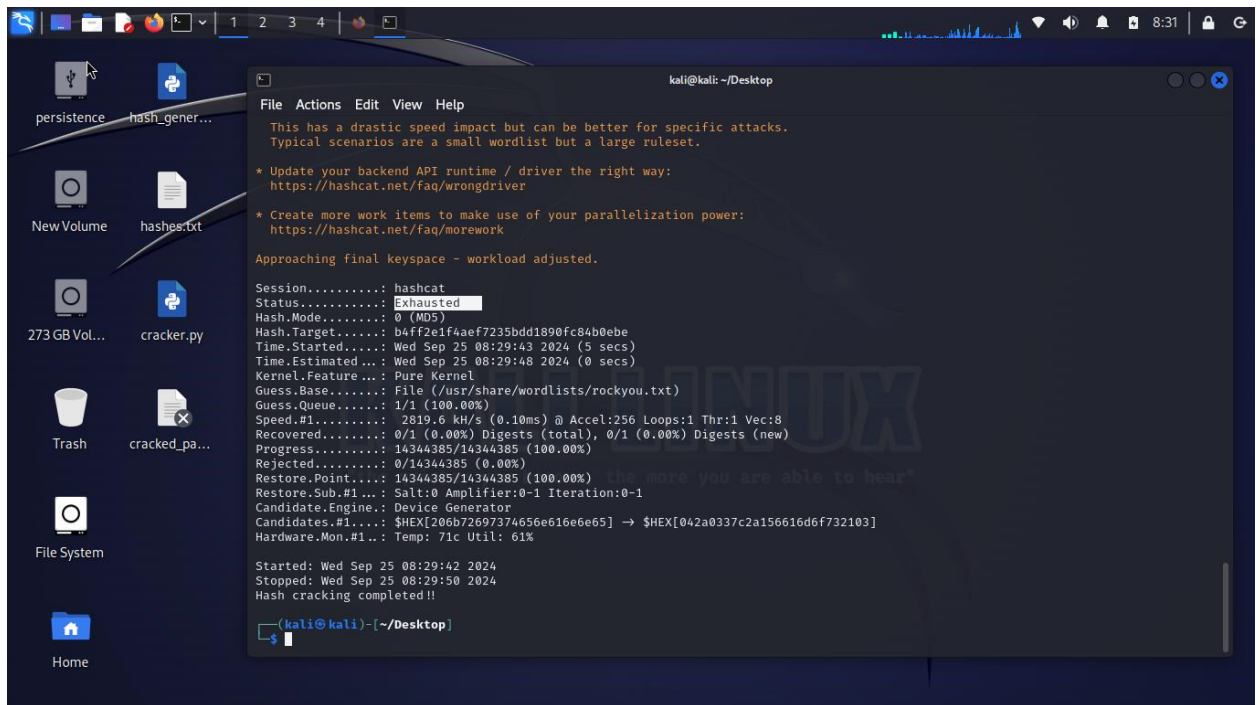
INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Wed Sep 25 08:36:41 2024
Stopped: Wed Sep 25 08:36:41 2024
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 4.0+debian Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.7, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
```

The desktop background and sidebar icons remain the same as in the previous screenshot.





Then from there i just cat the cracked_passwords.txt file to see which hashes were cracked. Two out of the three password hashes were cracked meaning they were weak passwords.

