



SECURE COMMUNICATION

**BRUTE FORCE CIPHER**

**By**

**Valery Aime Nundoya (B0099110)**

***School of Informatics***

***Department of Informatics and Engineering***

***Institute of Technology, Blanchardstown***

***Dublin 15.***

**Actual Words**  **Bachelor of Science Computing in**

**Information Technology**

**21/04/2019**

Table of Contents

[Introduction 3](#_Toc6775074)

[First Practice exercise 3](#_Toc6775075)

[The second python practical exercise 3](#_Toc6775076)

[Passing Argument Practical Exercise 4](#_Toc6775077)

[Main Exercise 5](#_Toc6775078)

[Output 6](#_Toc6775079)

[Conclusion 6](#_Toc6775080)

[References 7](#_Toc6775081)

[Figure 1 Checking Python version 3](#_Toc6770757)

[Figure 2 Hello World Printed in python 3](#_Toc6770758)

[Figure 3 Python console doing some calculations 3](#_Toc6770759)

[Figure 4 Argument Practical Exercise 1 4](#_Toc6770760)

[Figure 5 Argument Practical Exercise Output 4](#_Toc6770761)

[Figure 6 Python Array practice 4](#_Toc6770762)

[Figure 7 Bruteforce Output print 6](#_Toc6770763)

[Figure 8 Ascii Table 7](#_Toc6770764)

# Introduction

This is a python challenge lab to read in a character set of choice using dec for rot5, alpha for rot13 or caesar all for rot47 plus a ciphertext of choice to bruteforce. The cipher should then print out all possible decodings of the ciphertext given. According to (Python Language, n.d.) Rot13 algorithm is not known to many people. In its algorithm shifts, each character back or forward 13 places which also has the cipher algorithm capabilities to defer unwanted examination in python.

# First Practice exercise

This is basically to play around with some python commands on the terminal as shown below:

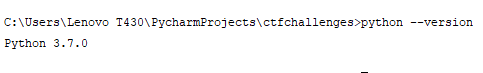


Figure 1 Checking Python version

## The second python practical exercise

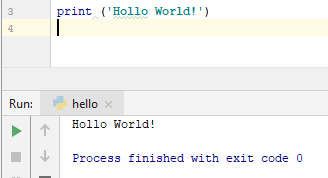


Figure 2 Hello World Printed in python

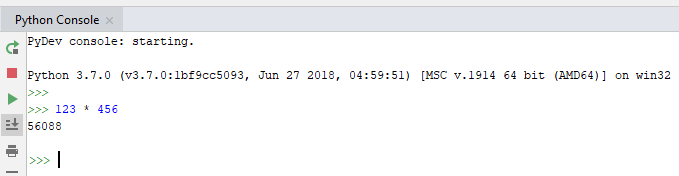


Figure 3 Python console doing some calculations

## Passing Argument Practical Exercise

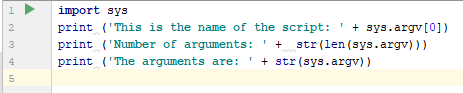


Figure 4 Argument Practical Exercise 1

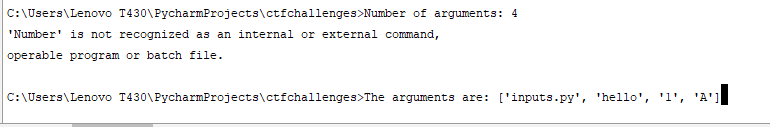


Figure 5 Argument Practical Exercise Output

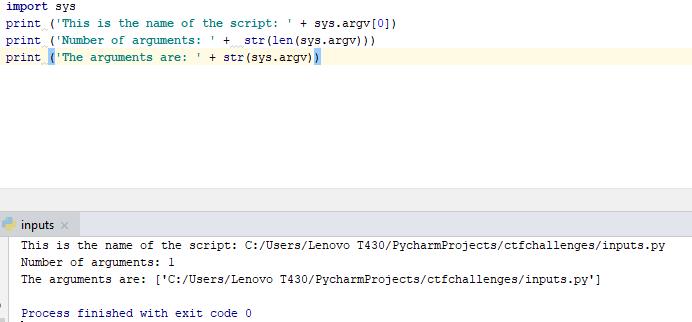


Figure 6 Python Array practice

# Main Exercise

In this exercise rot13 is used to loop over the given characters in the string s

Use d to Convert the letter in the string to a number in the ascii table

Shift number back to forward b checking in its lower case by an if statement of

If d is greater than Ord (which refer to the returns and integer from the ascii table) if is greater than 109 in the Ascii table M, it should take away 13 characters

By this stage if the character is less than capital M on the Ascii table, a character should be added unto it

Then an elif statement comes in to look at the upper-case D as shown in the rest of the code below:

*"""  
This code compare each integer representation against lowercase and uppercase letters  
then subtract or add 13 to shift characters  
"""***def** rot13(s):  
 result = **""  
  
 for** x **in** s: *# Loop over every character in the string s.* d = ord(x) *# Convert the letter in the string to a number in the ascii table  
 # d refers to the first later of the name valery .....number with ord which refers to the returns and integer from the ascii table  
  
 # Shift number back or forward checking in its lover case* **if** d >= ord(**'a'**) **and** d <= ord(**'z'**):  
 **if** d > ord(**'m'**): *# if the first letter of the chosen name is greater than the (109)ASCII TABLE M* d -= 13 *# then it should take away 13 characters* **else**: *# if the character is less than M* d += 13 *# If the ascii character is less than M it should add character unto it* **elif** d >= ord(**'A'**) **and** d <= ord(**'Z'**): *# This is looking at the upper case* **if** d > ord(**'M'**): *# if upper case D* d -= 13 *#* **else**:  
 d += 13  
  
 *# Append to result.* result += chr(d) *#  
  
 # Return transformation.* **return** result  
  
*# Test method.*print(rot13(**"VaLeRY"**)) *#*print(rot13(rot13(**"valery"**)))

## Output

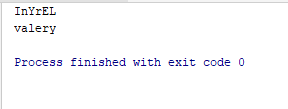


Figure 7 Bruteforce Output print

# Conclusion

After going through this bruteforce lab until printing out all the possible decodings of the given ciphertext python is an easy to go language though a little complex. All that was needed was to understand the dynamics and the algorithm. To conclude, hackers always have their aims of reaching their goals when it comes to bruteforce exercises, but at times it isn’t the case due to the demanded task. So, in this case I think it was ok.

# References

*Python Language*. (n.d.). Retrieved from https://www.dotnetperls.com/rot13-python?fbclid=IwA: https://www.dotnetperls.com/rot13-python?fbclid=IwAR2wAFF7IrvimM696qOFydbwmbIcjLqMWyfz1aeIJE\_x0-n0kQ5gZDOWzzI

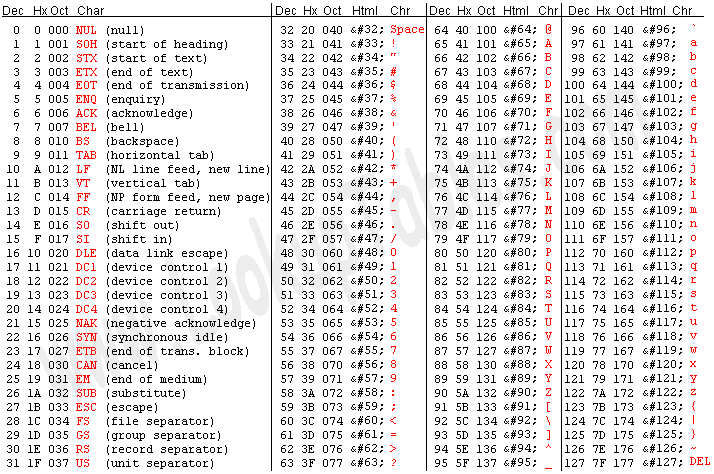


Figure 8 Ascii Table