Lab01 - Python 101 and Shift by N

**Answer the first three questions in a pdf file and upload it to Canvas. Questions 4 -6 remind you to upload your python code.**

1. **For Part 01, what’s the name of a Shift by N cipher with a key of 3?**

(5 points)

* Caesar Cipher

1. **Document (output copy and pasted into pdf file) each potential solution you arrived at for Part 03. For each attempt of the key you will include the N used in the shift and the plaintext output from the N**

(10 points)

* Attempt 1:
  + Output
    - Enter a string to be encrypted/decrypted by a N value: helloworld

Testing shift by: 0

helloworld

Testing shift by: 1

gdkknvnqkc

Testing shift by: 2

fcjjmumpjb

* + - Enter a string to be encrypted/decrypted by a N value: GTQTAAXDCHPGTQJXAIDCWDET

Testing shift by: 0

anknuurxwbjankdrucxwqxyn

Testing shift by: 1

zmjmttqwvaizmjcqtbwvpwxm

Testing shift by: 2

ylilsspvuzhylibpsavuovwl

Testing shift by: 3

xkhkrroutygxkhaorzutnuvk

Testing shift by: 4

wjgjqqntsxfwjgznqytsmtuj

Testing shift by: 5

vifippmsrwevifympxsrlsti

Testing shift by: 6

uhehoolrqvduhexlowrqkrsh

Testing shift by: 7

tgdgnnkqpuctgdwknvqpjqrg

Testing shift by: 8

sfcfmmjpotbsfcvjmupoipqf

Testing shift by: 9

rebellionsarebuiltonhope

* + As mentioned in the instructions I figured out that I could use the alphabet to my advantage by coming up with a for loop in 26 letters. And by the hint of chr and ord I figured that unicode character switching might be a better solution. So I used the equation we learned in class and took that by 26 and returned to the first letter in the alphabet which gave me all possible solutions for this particular scenario that doesn’t involve special characters, letters, or numbers.
* Attempt 2:
  + Another solution I came across is without using chr or ord and just using two different indexes and applying the shift to one like
    - newIndex = (ogIndex - shift) % 26

And then appending the new shifted character to the result

1. **Document in the pdf what the plaintext message is.**

(10 points)

* rebellionsarebuiltonhope

1. **Submit your commented code from Part one as “Lab01\_part01.py” to canvas**

(20 points)

#Part 01

#This program will prompt the user for a sentence.

#The program passes the sentence to a function named modify().

#The function modify() returns a string that shifts the characters

#in the sentence three characters to the right.

#Modify function:

# Takes in a list and then

# iterates through it shift the

# characters three to the right

# returns a string.

def modify(sentence):

#TODO: Implement Here

newsentence = ''

for i in range(len(sentence)):

# Shift position of new index, and wrapping around to a if needed

newindex = (i + 3) % len(sentence)

# Append character to the new index and to the modified sentence

newsentence += sentence[newindex]

# Return shifted sentence

return(newsentence)

#Main

def main():

sentence = input("Write a sentence without spaces: ")

#TODO: Implement Here

print(modify(sentence))

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. **Submit your commented code from Part two as “Lab01\_part02.py” to canvas**

(20 points)

#Part 02

#This program has a list of lists that will be searched.

#The program will prompt the user for a character and output

#the number of times the character appears in each of the lists.

#Frequency function:

# There are two input variables used by this function: the name of

# the list of lists and the character for which you are searching.

# The function prints (not returns) the number of occurrences of

# that character in each of the lists.

def frequency(inputs, character):

#TODO:

#print(dict((i, inputs.count(i)) for i in character))

#for i in inputs:

#print (i[0])

#for x in len(inputs[i]):

#print (inputs.count('A'))

#if (i == character):

# y = y + 1

# Iterate through inputs

for i in range(len(inputs)):

# Count occurences of a character in the array

count = inputs[i].count(character)

# Print out that what number in the array it is and the number of occurences within that array

print (f"Input [ {i} ] : {count}")

#print ("Input [ 0 ] : " + str(y))

#Main

def main():

#A list of lists

inputs = [list("AOLSLAALYJJVVRPLZBHYABDPAOBSLAZAOBURBMVAOLYAOPUNZ"),

list("ZCXNSMCPROCTYDGHCSUIRYTEBHHCJSMECWTQZCHDKRILLMSJS"),

list("WSCVKAUSAUDJAUUEAOPLAHSMACDGHAUUSGABXHAGEHASGDARU")]

charToCount = input("What character do you want to check? ")

# Checks for all occurences in inputs list

# Also if lower case letter will make uppercase

frequency(inputs, charToCount.upper())

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. **Submit your commented code from Part three as “Lab01\_part03.py” to canvas**

(35 points)

#Part 03

#This program will provide cryptanalysis on a Shift by N cipher using an exhaustive key search

#You can either hard code the ciphertext into the program (easy) or you can prompt for

#a text file or character input from the command line.

#You will need a function called analyze that will read in the ciphertext and then conduct an

#exhaustive key search that outputs its key (the N) and the answer in each trial)

#

#Example output:

#Testing shift by: 0

#qefpzixppfpsbovzlli

#Testing shift by: 1

#rfgqajyqqgqtcpwammj

#Testing shift by: 2

#sghrbkzrrhrudqxbnnk

#Look back at parts 01 and 02. They provide clues on how to implement this.

def analyze(ciphertext):

#TODO:

#26 for alphabet

for shift in range(26):

newText = ""

for char in ciphertext:

#chr for unicode, ord returns an integer value that is the unicode equivalent

decrypt = chr(((ord(char) - ord('a') - shift) % 26) + ord('a'))

newText += decrypt

# prints all shift possiblies but of course not 26 because it would be the same

print(f"Testing shift by: {shift}")

print(newText)

#Main

def main():

#TODO:

ciphertext = input("Enter a string to be encrypted/decrypted by a N value: ")

analyze(ciphertext)

if \_\_name\_\_ == "\_\_main\_\_":

main()