**Lab 4 - Strings**

**Objective:**

The objective is to practice the use of strings, string slicing, and string methods.

**Questions:**

1. Write a Python program to print each character of a string on a single line.

**2.** Write a Python program that will calculate the length of a string

(We already have a function len that does that, but we want to implement our own)

str **=** "geeks"

print(len(str))

**3.** Given the string "Monty Python":

1. Write an expression to print the first character.
2. Write an expression to print the last character.

(c) Write an expression including len to print the last character.

(d) Write an expression that prints "Monty".

**4.** Write a Python program that reads a string and prints a string that is made up of the first two characters and the last two characters. If the string has a length less than 4 the program prints a message on the screen.

For example: “hello there” will result in “here”

**5.** Given a variable S containing a string of odd length:

(a) Write an expression to print the middle character.

(b) Write an expression to print the string up to but not including the middle character

(i.e., the first half of the string).

(c) Write an expression to print the string from the middle character to the end (not

including the middle character).

**6.** Five string methods manipulate case: capitalize , title , swapcase , upper ,

and lower . Consider the strings: s1 = "concord" , s2 = "souix city" , s3 =

"HONOLULU" , and s4 = "TopHat".

1. Describe what capitalize does.
2. Describe what swapcase does.
3. Describe what upper does.
4. Describe what lower does.
5. Describe what title does.

**7.** Write a Python program that will reverse a string (using a loop, not using slicing)

def reverse\_string(str):

str1 = "" # Declaring empty string to store the reversed string

for index in str:

str1 = index + str1

return str1 # It will return the reverse string to the caller function

str = "Nneoama Onwe " # Given String

print("The original string is: ",str)

print("The reverse string is",reverse\_string(str)) # Function call

**8.** Write a Python program that will “encrypt” a string. The encryption algorithm we’ll use is add 1 to the ASCII code, so ‘a’ becomes ‘b’, ‘b’ becomes ‘c’, etc. The string ‘abc’ becomes ‘bcd’. You’ll need to use the functions ord() and chr() discussed in class

# Function to find the encrypted string

def findWord(c, n):

co = 0

# to store the encrypted string

s = [0] \* n

for index in range(n):

if (index < n / 2):

co += 1

else:

co = n - index

# after 'z', it should go to a.

if (ord(c[index]) + co <= 122):

s[index] = chr(ord(c[index]) + co)

else:

s[index] = chr(ord(c[index]) + co - 26)

print(\*s, sep = "")

# Driver code

s = "abcd"

findWord(s, len(s))

Hint: To encrypt the letter ‘a’ take the ASCII code of ‘a’ 97, add 1 (98) and find the

character with ASCII code 98 (‘b’). So ‘a’ encrypted becomes ‘b’

**9.** A palindrome is a string that reads the same forward and backwards. “Same” in a sense that: 1) case does not matter; and 2) punctuation is ignored. "Madam I'm Adam" is a palindrome. Write a program that checks if a string is palindrome. Steps:

1. Get a string from an input
2. Convert it to lowercase
3. Remove bad characters (punctuation and whitespaces)
4. Check whether the reverse of the resulting string is equal the string

Every letter is converted using the lower method. import string, brings in a series of predefined sequences:

string.digits

string.punctuation

string.whitespace

To remove all non-wanted characters use the replace method. First arg is what to replace, the second the replacement. The beginning of the solution is given below. Try to finish it...

**import** string

original\_str = input(**"Input a string: "**)

modified\_str = original\_str.lower()

bad\_chars = string.whitespace + string.punctuation

**10.** Experiment with the count method. What does it count? For example,

some\_string = "Hello world!"

some\_string.count("o")

**11.** Experiment with the strip method. What does it do? For example,

some\_string = "Hi!......"

some\_string.strip(".!")

**12.** (a) Suppose you want to print a line full of'#' characters. For simplicity, let’s say that a

line can have only 80 characters. One way is to create a long string to be printed. How

would you do it more elegantly in Python using the plus operation (+) of strings?

(b) Suppose you want to print a column full of '#' characters. For simplicity, let’s

say that a column could have only 30 characters. Similar to (a), how would you do

it more elegantly in Python using the multiply operation (\*) of strings? Hint: Use

the newline character (‘\n’).

**13.** Suppose you have a string ab\_string = 'abababababababab' . Write an

expression to remove all the b’s and create a string a string = 'aaaaaaaa' .

**14.** Although Python’s formatted printing can be cumbersome, it can often drastically

improve the readability of output. Try creating a table out of the following values:

Melting and Boiling Points of Alkanes

Name Melting Point (deg C) Boiling Point (deg C)

Methane -162 -183

Ethane -89 -172

Propane -42 -188

Butane -0.5 -135

**15. Pig Latin**

Pig Latin is a game of alterations played on words. To make the Pig Latin form of an

English word the initial consonant sound is transposed to the end of the word and an

“ay” is affixed. Specifically there are two rules:

(a) If a word begins with a vowel, append “yay” to the end of the word.

(b) If a word begins with a consonant, remove all the consonants from the beginning

up to the first vowel and append them to the end of the word. Finally, append “ay”

to the end of the word.

For example:

* dog ⇒ ogday
* scratch ⇒ atchscray
* is ⇒ isyay
* apple ⇒ appleyay

Write a program that repeatedly prompts for an English word to translate into Pig

Latin and prints the translated word. If the user enters a period, halt the program.

Hints:

* Slicing is your friend: it can pick off the first character for checking, and you can slice

off pieces and concatenate to yield the new word.

* Making a string of vowels allows the use of the **in** operator: vowels = 'aeiou' .

word = "Nneoma Onwe "

import string

# Create a list of vowels an consonants

vowels = ['a','e','i','o','u','y']

vowels += [v.upper() for v in vowels]

consonants = [x for x in string.ascii\_letters if x not in vowels]

if word[0] in consonants:

# Find the first vowel

idx = min([word.find(v) for v in vowels if word.find(v)>0])

# Split the word at this point and add 'ay'

word = word[index:] + word[:index] + 'ay'

print(word)