**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.**

for c in “Hello World”:

print(c)

**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)**

my\_str = input(“Please enter a string: ”)

count = 0

for c in “Hello World”:

count += 1

print(count)

**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".**

my\_str = “Monty Python”

# a

print(my\_str[0])

# b

print(my\_str[-1])

print(my\_str[11])

# c

print(my\_str[len(my\_str) - 1])

# d

for i in range(6):

print(my\_str[i], end=“”)

print(my\_str[:6])

monty, python = my\_str.split()

print(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”**

my\_str = input(“Please enter a string: ”)

if len(my\_str) < 4:

print(“String is too short!”)

else:

print(my\_str[:2] + my\_str[-2:])

**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).**

S = input(“Please enter a string of odd length: ”)

# a

print(S[len(S) // 2])

# b

print(S[:len(S) // 2])

# c

print(S[len(S) // 2 + 1:])

**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.**

s1 = “concord”

s2 = “souix city”

s3 = “HONOLULU”

s4 = “TopHat”

print(s1.capitalize())

print(s1.swapcase())

print(s1.upper())

print(s1.lower())

print(s1.title())

print(s2.capitalize())

print(s2.swapcase())

print(s2.upper())

print(s2.lower())

print(s2.title())

print(s3.capitalize())

print(s3.swapcase())

print(s3.upper())

print(s3.lower())

print(s3.title())

print(s4.capitalize())

print(s4.swapcase())

print(s4.upper())

print(s4.lower())

print(s4.title())

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

s1 = “concord”

# reverse = “”

# reverse = “c”

# reverse = “oc”

# reverse = “noc”

# …

# reverse = “drocnoc”

# slicing example

print(s1[::-1])

for c in s1:

reverse = c + reverse

print(reverse)

# Or

s1 = “concord”

for i in range (len(s1) – 1, -1, -1):

reverse += s1(1)

print(reverse)

**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**

**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’**

my\_str = “abc”

# result = “bcd”

for c in my\_str:

print(chr(ord(c) + 1))

**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 argis 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**

import string

my\_str = input(“Please enter a string: “)

my\_str = my\_str.lower()

bad\_chars = string.whitespace + string.punctuation

for c in my\_str:

my\_str = my\_str.replace(c, “”)

print(my\_str)

if my\_str == my\_str[::-1]:

print(“The string is a palindrome”)

else:

print(“The string is NOT a palindrome”)

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

**some\_string = "Hello world!"**

**some\_string.count("o")**

some\_string = "Hello world!"

print(some\_string.count("o"))

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

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

**some\_string.strip(".!")**

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

print(some\_string.strip(".!"))

Result: Removes dots from the start and end of the string.

**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’).**

# a

my\_string = “”

for i in range(80):

# my\_str += “str”

my\_string += “#”

# my\_string = \n

print(my\_string)

# b

my\_string = “#\n” + 30

print(my\_string)

**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' .**

ab\_string = 'abababababababab'

result = “”

for c in ab\_string:

if c == “b”:

continue

else:

result += c

print(ab\_string[::2])

**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**

# 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

print(“Melting and Boiling Points of Alkanes”)

print(“{:14s} {:28s} {:21s}”.format(“Name”, “Melting Points (deg C)”, “Boiling Point (deg C)”))

print(“{:14s} {:<-28d} {:<-21d}”.format(“Methane”, -162, -183))

print(“{:14s} {:<-28d} {:<-21d}”.format(“Ethane”, -89, -172))

print(“{:14s} {:<-28d} {:<-21d}”.format(“Propane”, -42, -188))

print(“{:14s} {:<-28.2f} {:<-21d}”.format(“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 = input(“Please enter a word or a . to stop: ”)

vowels = ‘aeiou’

while word != “.”:

if word[0] in vowels:

print(word = “yay”)

else:

i = 1

while word[i] not in vowels:

i += 1

if i == len(word):

print(“word + “ay”)

break

else:

result = word[i:] + word[0:1] + “ay”)

word = input(“Please enter a word or a . to stop: ”)