Lab 12

Strings

Objective:

The objective of this lab will be to learn about strings and with the help of examples and learning tasks.

Activity Outcomes:

The activities provide hands - on practice with the following topics

- Implement a string and its related operations

Instructor Note:

As a pre-lab activity, read Chapter 4 and 9 from the textbook “Python Basics: A Practical Introduction to Python 3, 2021”.

1) Useful Concepts

A string is a collection of text. Anything in Python surrounded by ‘’ or “” is called a string.

1. Single line strings are surrounded by single or double commas while we can write multiline strings as well using triple commas.

```
x = 'single line string'
x = "single line string"

y = '''multiline string
              multiline string '''
```

2. We can access any string character by specifying its index in the string. Indexes define the position of any character in the string and they start from 0. So the first character would be present on the 0th index. The syntax is as follows

```
<string_name>[index]
```

3. We can access any string Length by using the len() function and it will return the total no of characters in the string Suppose S1=’Hello’

```
Len(< S1> )
5
```

4. We can also take out a chunk of a string. This is known as slicing. This is done by using the index operator and specifying two indexes separated by a : . The first is the starting index and the other is the ending index.

```
<string_name>[start_index:end_index]
```

5. We can join two strings together by using the + operator. This is known as concatenation.

```
result = <string1> + <string2>
```

6. Repetition as the name suggest means to multiply the items or Repeat Sequence s1 n times. n copies of sequence s are concatenated The syntax is follows

```
s1 = s1 x n
Or
s1 = n x s1
```

7. Membership testing: is done by using ***in*** or ***not in*** operator to find the presence of an element in the sequence

```
x in s # True if element x is in sequence s.
x not in  s # True if element x is not in sequence s.
```

8. Suppose you want to print a message with quotation marks in the output. Can you write a statement like this?
print("He said, "John's program is easy to read"")

No, this statement has an error. Python thinks the second quotation mark is the end of the string and does not know what to do with the rest of the characters. To overcome this problem, Python uses a special notation to represent special characters, as shown in below Table 3.3. This special notation, which consists of a backslash (\) followed by a letter or a combination of digits, is called an escape sequence.

Python Escape Sequences

Character Escape Sequence	*Name*	*Numeric Value*
\b	Backspace	8
\t	Tab	9
\n	Linefeed	10
\f	Formfeed	12
\r	Carriage Return	13
\\	Backslash	92
\'	Single Quote	39
\"	Double Quote	34

Operation	**Description**
○ len(s)	Length of sequence s, i.e., the number of elements in s.
○ min(s)	Smallest element in sequence s.
○ max(s)	Largest element in sequence s.
○ sum(s)	Sum of all elements in sequence s.
○ <, <=, >, >=, =, !=	Compares two sequences.

2) Solved Lab Activities

Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
1	*10*	*Low*	*CLO-7*
2	*10*	*Low*	*CLO-7*
3	*10*	*Low*	*CLO-7*
4	*10*	*Medium*	*CLO-7*
5	*10*	*Medium*	*CLO-7*
6	*10*	*Medium*	*CLO-7*

Activity 1:

Python program to tell the number of characters in a string.

Solution:

```
x                    =                    "This                    is                    Python"
print(len(x))
```

Output

```
14
```

```
The value of x is: 2
```

```
The value of x is: 3
```

Activity 2:

Python program to illustrate using a string with escaped characters

```
x = "\"This is Python\", said Amanda"
print(x)
```

Output

```
"This is Python", said Amanda
```

Activity 3:

Python program to illustrate string concatenation, indexing, and slicing.

```
string1 = 'Australia'
string2 = 'Pakistan won the world cup in 2012'

result = string1 + ' vs ' + string2[:8] + ' Match ' + string2[30]
print(result)
```

Output

```
Australia vs Pakistan match 2
```

Activity 4:

Python program to illustrate membership testing .

```
>>> s1 = "Welcome"
>>> "come" in s1
True
>>> "come" not in s1
False
```

Here is another

```
s = input("Enter a string: ")
if "Python" in s:
print("Python", "is in", s)
else:
print("Python", "is not in", s)
```

Output

```
python is in Welcome to Python.
```

Activity 5:

A string is a palindrome if it reads the same forward and backward. The words "mom," "dad," and "noon," for instance, are all palindromes

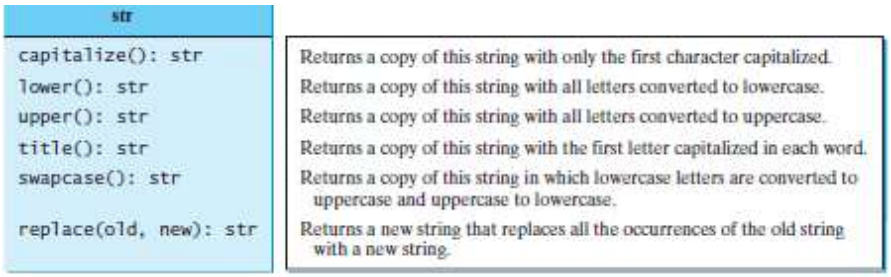
```
# Prompt the user to enter a string
s = input("Enter a string: ")
s = s.strip()           # Remove white spaces

low = 0                 # The index of the first character in the string
high = len(s)-1         # The index of the last character in the string

while low < high:
    print(low, "    ", high)
    if s[low] == s[high]:
        low += 1
        high -= 1
    else:
        print("Not Palindrome")
        break

if low >= high:
    print("Its a Palindrome")
```

Output

```
Enter a string: MoM
```

```
It’s a Palindrome
```

Activity 6:



str	
capitalize(): str	Returns a copy of this string with only the first character capitalized.
lower(): str	Returns a copy of this string with all letters converted to lowercase.
upper(): str	Returns a copy of this string with all letters converted to uppercase.
title(): str	Returns a copy of this string with the first letter capitalized in each word.
swapcase(): str	Returns a copy of this string in which lowercase letters are converted to uppercase and uppercase to lowercase.
replace(old, new): str	Returns a new string that replaces all the occurrences of the old string with a new string.

Output

```
1 >>> s = "welcome to python"
2 >>> s1 = s.capitalize()
3 >>> s1
4 'Welcome to python'
5 >>> s2 = s.title()
6 >>> s2
7 'Welcome To Python'
8 >>> s = "New England"
9 >>> s3 = s.lower()
10 >>> s3
11 'new england'
12 >>> s4 = s.upper()
13 >>> s4
14 'NEW ENGLAND'
15 >>> s5 = s.swapcase()
16 >>> s5
17 'nEW eNGLAND'
18 >>> s6 = s.replace("England", "Haven")
19 >>> s6
20 'New Haven'
21 >>> s
22 'New England'
23 >>>
```

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficulty and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

Write a program to which replaces the first character of a string with the character 'z'

Lab Task 2

Write a Python program to remove the characters which have odd index values of a given string

Lab Task 3

Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string
Sample String : 'abc', 'xyz'
Expected Result : 'xyc abz'

Lab Task 4

Write a Python program to print a specified list after removing the 0th, 4th and 5th elements.

Lab 13

Loops + Nested Loops

Objective:

The objective of this lab will be to learn about loops and nested loops with the help of examples and learning tasks.

Activity Outcomes:

The activities provide hands - on practice with the following topics

- Implement a while loop
- Implement a for-in loop
- Implement nested loops

Instructor Note:

As a pre-lab activity, read Chapter 6 from the textbook "Python Basics: A Practical Introduction to Python 3, 2021".

1) Useful Concepts

A loop is a block of code that gets repeated over and over again either a specified number of times or until some condition is met which is why a loop runs only a finite number of times. If we specify a condition that does not get fulfilled ever during loop iterations then it would go on forever and the program would freeze.

1. One kind of a loop is a while loop. The syntax is given below. We specify a condition in the round brackets after a while just like we do for the if statement. This loop keeps on running while the statement passed is true(hence the name while).

```
while(<condition>):
    // some code to repeat
```

2. Another kind of a loop is a for-in loop. The syntax is given below. A for-in loop iterates over a sequence(list, string or range). The variable we use after **for**, gets the value of the next item in the sequence in each iteration while the variable after **in** is a sequence which we want iterated on.

```
for                    <item_name>                    in                    <sequence_name>:
    //                 some                 code                 to                 repeat
```