Strings in Python (Class XI Computer Science)

1. Introduction to Strings:

A **string** in Python is a **sequence of characters** enclosed within single quotes (`' '`), double quotes (`" "`), or triple quotes (`''' '''` or `""" """`).

Examples:

```
name = 'Alakshya'
city = "Dehradun"
paragraph = '''This is a
            multi-line string.'''
```

- **Strings are Immutable:**
  Once created, a string **cannot be changed**. Any modification creates a new string.

---

3. String Operations

A. Concatenation (+): Combining two or more strings is known as concatenation.

```
a = "Hello"
b = "World"
print(a + " " + b)   # Output: Hello World
```

B. Repetition (*): It repeats the string multiple times.

```
print("Hi! " * 3)   # Output: Hi! Hi! Hi!
```

C. Membership Operators (`in`, `not in`): It checks whether a substring exists in another string.

```
s = "computer"
print("put" in s)        # True
print("Comp" not in s)   # True
```

D. String Slicing ✂: Slicing means extracting a portion (substring) of a string using **index positions**.

Syntax: `string[start : end : step]`

- `start` → index to begin (default = 0)
- `end` → index to stop (excluded)
- `step` → increment (default = 1)

Examples:

```
s = "PYTHON"
print(s[0:3])      # Output: PYT
print(s[2:])       # Output: THON
print(s[:4])       # Output: PYTH
print(s[::2])      # Output: PTO
print(s[::-1])     # Output: NOHTYP (reversed string)
```

4. Traversing a String Using Loops:

String traversal in Python refers to the process of iterating through each character of a string, typically to perform an operation on each character or to analyze the string's content. You can use loops to access each character in a string.

Using for loop:

```
word = "HELLO"
for ch in word:
    print(ch)
```

Using while loop:

```
word = "HELLO"
i = 0
while i < len(word):
    print(word[i])
    i += 1
```

5. Built-in String Functions and Methods

Strings in Python have a rich set of built-in methods. Remember: **Strings are immutable, so these methods return a new string and do not modify the original.** Here are some **commonly used string methods** in Python.

Function / Method	Description	Example
`len(s)`	Returns length of string	`len("India") → 5`
`capitalize()`	Converts first character to uppercase	`"hello".capitalize() → "Hello"`
`title()`	Converts first letter of each word to uppercase	`"welcome to school".title() → "Welcome To School"`
`lower()`	Converts all characters to lowercase	`"HELLO".lower() → "hello"`
`upper()`	Converts all characters to uppercase	`"hello".upper() → "HELLO"`
`count(sub)`	Returns number of occurrences of substring	`"banana".count("a") → 3`
`find(sub)`	Returns index of first occurrence; `-1` if not found	`"python".find("t") → 2`
`index(sub)`	Like find(), but gives error if not	`"python".index("y") → 1`

Function / Method	Description	Example
	found	
`endswith(sub)`	Checks if string ends with substring	`"hello".endswith("lo") → True`
`startswith(sub)`	Checks if string starts with substring	`"hello".startswith("he") → True`
`isalnum()`	True if all characters are letters or digits	`"abc123".isalnum() → True`
`isalpha()`	True if all characters are alphabets	`"abc".isalpha() → True`
`isdigit()`	True if all characters are digits	`"123".isdigit() → True`
`islower()`	True if all characters are lowercase	`"hello".islower() → True`
`isupper()`	True if all characters are uppercase	`"HELLO".isupper() → True`
`isspace()`	True if string contains only whitespace	`" ".isspace() → True`
`lstrip()`	Removes spaces from left	`" hello".lstrip() → "hello"`
`rstrip()`	Removes spaces from right	`"hello ".rstrip() → "hello"`
`strip()`	Removes spaces from both sides	`" hello ".strip() → "hello"`
`replace(old,new)`	Replaces substring with another	`"good morning".replace("morning","evening") → "good evening"`
`join(iterable)`	Joins elements using string as separator	`" ".join(["Python","is","fun"]) → "Python is fun"`
`partition(sep)`	Splits into 3 parts: before, separator, after	`"hello@world".partition("@") → ('hello', '@', 'world')`
`split(sep)`	Splits string into list using separator	`"a,b,c".split(",") → ['a','b','c']`

Important Questions on String:

Q1. Write a Python program to enter a string and print number of upper and lower case characters in it.

```
# Input from user
text = input("Enter a string: ")

# Initialize counters
upper_count = 0
lower_count = 0

# Loop through each character in the string
for ch in text:
    if ch.isupper():
        upper_count += 1
    elif ch.islower():
        lower_count += 1
```

```
# Display the result
print("Number of uppercase characters:", upper_count)
print("Number of lowercase characters:", lower_count)
```

Q2. Write a Python program to enter a string and check if it is palindrome or not.

```
# Input from user
text = input("Enter a string: ")

# Remove spaces and convert to lowercase for uniform comparison
clean_text = text.replace(" ", "").lower()

# Check if string reads same forward and backward
if clean_text == clean_text[: : -1] :
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

Q3. Predict the output of the following Python code:

```
text = "PythonProgramming"
print(text[0:6])
print(text[-7:])
print(text[2:15:3])
print(text[::-1])
print("gram" in text)
```

Output:

```
Python
ramming
tnoai
gnimmargorPnohtyP
True
```

Q4. What is the output of the following expression ?

```
Sports="Paralympic Games"
print (Sports.split("m"))
```

Output:

```
['Paraly', 'pic Ga', 'es']
```