**STRING AND ITS OPERATIONS -**

**Definition of String in Python**

A string in Python is a sequence of characters enclosed within quotes.

It is one of the most commonly used data types in Python for storing text data.

Strings are immutable (once created, they cannot be changed).

Strings can be indexed, sliced, and iterated.

Ways to Define Strings in Python

You can define a string in the following ways:

a) Single Quotes ' '

name = 'Alice'

b) Double Quotes " "

greeting = "Hello"

c) Triple Quotes ''' ''' or """ """ (for multiline strings)

multiline = '''This is a

multiline

string.'''

d) Using str() constructor

s = str(1234) # Converts number to string

**Uses of Strings in Python**

1.Strings are used in:

2.Displaying messages

3.Storing text input/output

4.Manipulating data in files, databases

5.Web development (HTML, URLs, etc.)

6.Data analysis and machine learning (labels, categories)

**String Operations with Theory & Examples**

A) Concatenation (+)

Used to join two or more strings.

first = "Hello"

second = "World"

print(first + " " + second) # Output: Hello World

B) Repetition (\*)

Repeats the string multiple times.

word = "Hi"

print(word \* 3) # Output: HiHiHi

C) Indexing

Access individual characters using index (starting from 0).

text = "Python"

print(text[0]) # Output: P

print(text[-1]) # Output: n (last character)

D) Slicing

Extract part of the string. Syntax: string[start:end]

msg = "Programming"

print(msg[0:6]) # Output: Progra

print(msg[3:]) # Output: gramming

print(msg[:4]) # Output: Prog

E) Length (len())

Returns number of characters in string.

s = "Hello"

print(len(s)) # Output: 5

F) Membership Operators (in, not in)

print('P' in "Python") # Output: True

print('z' not in "Python") # Output: True

G) Iteration through String

for letter in "Hey":

print(letter)

H) String Methods

1. lower() and upper()

s = "Hello"

print(s.lower()) # hello

print(s.upper()) # HELLO

2. strip()

Removes leading/trailing whitespace.

s = " Hello "

print(s.strip()) # Hello

3. replace(old, new)

s = "Hello World"

print(s.replace("World", "Python")) # Hello Python

4. split()

Splits the string into a list.

s = "apple,banana,grape"

print(s.split(',')) # ['apple', 'banana', 'grape']

5. find()

Returns the index of the first occurrence.

s = "Programming"

print(s.find('g')) # Output: 3

6. count()

Counts number of occurrences.

s = "banana"

print(s.count('a')) # Output: 3

I) String Formatting

Using f-strings (Python 3.6+)

name = "Alice"

age = 19

print(f"My name is {name} and I am {age} years old.")

Using .format()

print("My name is {} and I am {} years old.".format(name, age))

J) Escape Characters

print("Hello\nWorld")

print("She said, \"Python is fun!\"")

………………………………………………

Ex - 1

#Python program to count occurrences of 'a' in a word

word "banana"

# The word to check

count = word.count('a') # Counting the occurrences of 'a'

#Display the result

print(f"The letter r'a' appears {count) times in the word '{word)'.")

Output - The letter 'a' appears 3 times in the word 'banana'.

Ex - 2

is palindrome = lambda word: word == word[::-1]

Ex-3

1. any\_lowercase1(s):

a. What it does: The function checks whether any character in the string s is lowercase. If it finds a lowercase character, it immediately returns True. the first character is not lowercase, it returns Fal se without checking the rest of the string.

b. Issue: It only checks the first character and doesn't evaluate the entire string.

2. any\_lowercase2(s):

a.What it does: It appears to check whether characters in the string s are lowercase. However, the condition 'c'.islower() checks whether the character literal 'c' (not the variable c) is lowercase, which is always True since 'c' is a lowercase letter.

b. Issue: It doesn't check the string s. Instead, it always returns 'True (as a string) regardless of the input.

3. any\_lowercase3(s):

a. What it does: The function assigns the result of c. is lower () to the variable flag for each character in the string s, and then immediately returns the value of flag.

b. Issue: It only evaluates the first character in the string g and returns whether it is lowercase. The rest of the string is never checked.

4. any\_lowercase4(s):

a. What it does: The function uses a varighle flad to keep track of whether any character in the string is lowercase. It uses the or operator, which evaluates to True if any lowercase character is found. The loop continues until all characters are checked, and flag reflects the correct result.

b. Outcome: This function works correctly and checks the entire string.

returns True if there's at least one lowercase character, otherwise False

5. any\_lowercase5(s):

a. What it does: The function checks whether all characters in the string are lowercase by returning False as soon as it finds a non-lowercase character. If it finishes the loop without finding a non-lowercase character, it returns True.

b. Issue: This function doesn't check for the presence of any lowercase letters-it checks whether the entire string consists of lowercase letters.

Ex- 4

def rotate\_word(word, shift):

"""Rotates each letter in a word by a fixed number of places (shift).

Preserves the case of the letters and ignores non-alphabetic characters.

Parameters:

word (str): The string to rotate.

shift (int): The number of places to shift each letter.

Returns:

str: The rotated string.

rotated ="

for char in word:

If char.isalpha(): #Check if the character is a letter

if char.islower():

# Rotate lowercase letters

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

elif char.isupper():

# Rotate uppercase letters

new\_char = chr(((ord(char) - ord('A') + shift) % 26) + ord('A"))

rotated += new\_char

else:

# Non-alphabetic characters remain unchanged

rotated += char

return rotated

# Example usage

print(rotate\_word("cheer", 7)) # Output: "jolly"

print(rotate\_word("melon", -10)) # Output: "cubed"

print(rotate\_word("HAL", -1)) # Output: "IBM