A **string** is a sequence of characters. It can contain letters, numbers, symbols, and spaces. Strings are one of the most commonly used data types in Python, and they are essential for handling text-based data.

**Key Characteristics of Strings:**

1. **Immutable**: Strings in Python are immutable, which means once a string is created, it cannot be changed. Any operation that appears to modify a string actually creates a new string.
2. **Defined in Quotes**: Strings are created by enclosing characters in either single quotes ('...'), double quotes ("..."), or triple quotes ('''...''' or """..."""). Triple quotes are typically used for multi-line strings.

single\_quote\_string = 'Hello'

double\_quote\_string = "Hello"

multi\_line\_string = '''This is a multi-line string''

1. **Indexing**: Each character in a string has an index, starting from 0. You can access individual characters using their index.

**my\_string = "Python"**

**print(my\_string[0]) # Output: 'P'**

**print(my\_string[1]) # Output: 'y'**

1. **Slicing**: You can retrieve a portion (or "slice") of a string using the slicing notation start:end. The start is inclusive, but the end is exclusive.

my\_string = "Python"

print(my\_string[0:3]) # Output: 'Pyt' (characters from index 0 to 2)

print(my\_string[2:]) # Output: 'thon' (from index 2 to the end)

print(my\_string[:4]) # Output: 'Pyth' (from start to index 3)

1. **Negative Indexing**: Negative numbers can be used to access characters from the end of the string. -1 refers to the last character, -2 to the second-to-last, and so on.

my\_string = "Python"

print(my\_string[-1]) # Output: 'n' (last character)

print(my\_string[-2]) # Output: 'o' (second-to-last character)

my\_str = "pythonprogramming"

print(my\_str[0:10])    # Output: pythonprog

print(my\_str[0:11:2])  #ptopor

print(my\_str[0:])      #pythonprogramming

print(my\_str[:14])     #pythonprogramm

print(my\_str[:])       #pythonprogramming

print(my\_str[0::3])    #phpgmn

print(my\_str[5:1])     # nothing

print(my\_str[5:1:-1])  #noht

# print(my\_str[18])    #IndexError: string index out of rang

print("........Negative Indexing Examples....")

my\_string = "PythonProgramming"

# Slice from index -1 (last character) to -6

print(my\_string[-1:-6:-1]) #output: gnimm

print(my\_string[-1:-6])    #output: empty

print(my\_string[-6:-1])    #output: ammin

print(my\_string[-14:-10])  #output: honP

print(my\_string[-1:-12:-2]) #output: gimroP

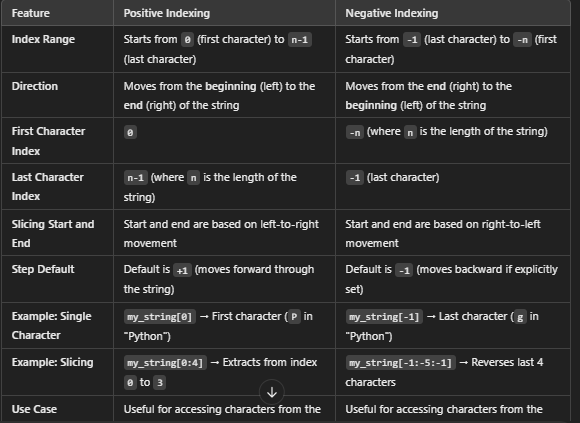
print(my\_string[-5:])       #output: mming

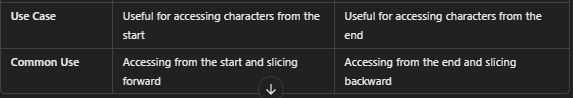
print(my\_string[-8:-3])    #output: gramm

print(my\_string[:-12]) #output: Pytho

print(my\_string[::-1])

print(len(my\_string))





**Basic Operations with Strings:**

1. **Concatenation**: Strings can be combined (or concatenated) using the + operator.

str1 = "Hello"

str2 = "World"

combined = str1 + " " + str2 # Output: 'Hello World'

1. **Repetition**: You can repeat a string multiple times using the \* operator.

python

Copy code

repeated = "Ha" \* 3 # Output: 'HaHaHa'

1. **Length**: The len() function returns the number of characters in a string.

my\_string = "Python"

print(len(my\_string)) # Output: 6

1. **Membership**: You can check if a certain substring is present in a string using the in keyword.

my\_string = "Python programming"

print("Python" in my\_string) # Output: True

print("Java" in my\_string) # Output: False

1. **String Comparison**: You can compare strings using comparison operators like ==, !=, <, >, etc.

print("apple" == "apple") # Output: True

print("apple" != "Apple") # Output: True (Python is case-sensitive)

1. **Escape Characters**: If you need to include special characters like newlines (\n) or tabs (\t) in your string, you can use escape sequences.

my\_string = "This is line one.\nThis is line two."

print(my\_string)

# Output:

# This is line one.

# This is line two.

Common escape sequences:

* + \n for newline
  + \t for tab
  + \' for single quote
  + \" for double quote
  + \\ for backslash

1. **Raw Strings**: By prefixing a string with r or R, you create a **raw string** where escape sequences are ignored. This is especially useful when dealing with paths or regular expressions.

raw\_string = r"C:\Users\Name\Documents"

print(raw\_string) # Output: C:\Users\Name\Documents

