# What are strings?

In python, anything that you enclose between single or double quotation marks is considered a string. A string is essentially a sequence or array of textual data. Strings are used when working with Unicode characters.

Example

name = "Harry"

print("Hello, " + name)

Output

Hello, Harry

Note: It does not matter whether you enclose your strings in single or double quotes, the output remains the same.

Sometimes, the user might need to put quotation marks in between the strings. Example, consider the sentence: He said, “I want to eat an apple”.

How will you print this statement in python?: He said, "I want to eat an apple". We will definitely use single quotes for our convenience

print('He said, "I want to eat an apple".')

Multiline Strings

If our string has multiple lines, we can create them like this:

a = """Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua."""

print(a)

## Accessing Characters of a String

In Python, string is like an array of characters. We can access parts of string by using its index which starts from 0.  
Square brackets can be used to access elements of the string.

print(name[0])

print(name[1])

## String Slicing & Operations on String

Length of a String

We can find the length of a string using len() function.

Example:

fruit = "Mango"

len1 = len(fruit)

print("Mango is a", len1, "letter word.")

Output:

Mango is a 5 letter word.

## String as an array

A string is essentially a sequence of characters also called an array. Thus we can access the elements of this array.

Example:

pie = "ApplePie"

print(pie[:5])

print(pie[6]) #returns character at specified index

Output:

Apple

i

Note: This method of specifying the start and end index to specify a part of a string is called slicing.

Slicing Example:

pie = "ApplePie"

print(pie[:5]) #Slicing from Start

print(pie[5:]) #Slicing till End

print(pie[2:6]) #Slicing in between

print(pie[-8:]) #Slicing using negative index

Output:

Apple

Pie

pleP

ApplePie

**String Methods in Python**

Python provides a rich set of built-in string methods to manipulate and analyze text data. These methods help in changing the format, searching within text, validating content, and more. Below is a comprehensive guide with simple examples and explanations.

**Note:** Strings are immutable

**1. Case Conversion Methods**

**upper()**

Converts all characters in a string to **uppercase** (capital letters). Useful when you want to standardize text or make comparisons without worrying about letter cases.

str1 = "AbcDEfghIJ"

print(str1.upper())

**Output:**

ABCDEFGHIJ

**lower()**

Converts all characters in a string to **lowercase**. Often used to make strings uniform before comparison or processing.

str1 = "AbcDEfghIJ"

print(str1.lower())

**Output:**

abcdefghij

**capitalize()**

Converts **only the first character** of the string to uppercase and the rest to lowercase. This is commonly used to format names or sentences.

str1 = "hello"

print(str1.capitalize())

str2 = "hello WorlD"

print(str2.capitalize())

**Output:**

Hello

Hello world

**title()**

Capitalizes the **first letter of each word** in the string. Useful for formatting titles or names.

str1 = "he's name is dan."

print(str1.title())

**Output:**

He'S Name Is Dan.

Note: It may treat apostrophes as word separators.

**swapcase()**

**Swaps the case** of each letter: lowercase becomes uppercase, and uppercase becomes lowercase.

str1 = "Python is COOL!"

print(str1.swapcase())

**Output:**

pYTHON IS cool!

This method is helpful in formatting or toggling letter cases quickly.

**2. Whitespace & Character Removal**

**strip()**

Removes **extra spaces** or whitespace characters (like tabs and newlines) from the **start and end** of the string.

str2 = " Silver Spoon "

print(str2.strip())

**Output:**

Silver Spoon

Helpful when processing input data or cleaning text.

**rstrip()**

Removes **trailing characters** (characters from the end of the string). By default, it removes whitespace, but you can specify other characters.

str3 = "Hello!!!"

print(str3.rstrip("!"))

**Output:**

Hello

This is useful when you want to clean up punctuation or formatting characters.

**3. Search & Replace Methods**

**replace()**

Replaces **all occurrences** of one substring with another. Useful for correcting or updating words in a sentence.

str2 = "Silver Spoon"

print(str2.replace("Sp", "M"))

**Output:**

Silver Moon

**find()**

Returns the **index** of the **first occurrence** of the specified substring. If not found, it returns -1. It doesn’t cause an error, which makes it safe to use.

str1 = "He's name is Dan."

print(str1.find("is")) # Found

print(str1.find("Daniel")) # Not found

**Output:**

10

-1

**index()**

Works like find(), but **raises an error** (ValueError) if the substring is not found. Use only when you're sure the text exists.

str1 = "He's name is Dan."

print(str1.index("Dan")) # Found

print(str1.index("Daniel")) # Error

**Output:**

13

ValueError: substring not found

**count()**

Counts how many times a substring appears in the string. Helpful for analyzing text.

str2 = "Abracadabra"

print(str2.count("a"))

**Output:**

4

**4. String Validation Methods**

These methods return True or False based on the contents of the string.

**isalnum()**

Returns True if the string contains only **letters and numbers**, and has no spaces or symbols.

str1 = "Welcome123"

print(str1.isalnum())

**Output:**

True

**isalpha()**

Returns True if the string contains only **letters**. If it has numbers or spaces, it returns False.

str1 = "Hello"

print(str1.isalpha())

**Output:**

True

**islower() / isupper()**

Check if all characters are **lowercase** or **uppercase**, respectively.

print("hello".islower()) # True

print("HELLO".isupper()) # True

These are useful when validating text formats.

**istitle()**

Returns True if the string is in **title case** (i.e., each word starts with a capital letter).

print("World Health Org".istitle()) # True

print("hello world".istitle()) # False

**isspace()**

Returns True if the string contains **only whitespace characters** (spaces, tabs, newlines).

print(" ".isspace()) # True

print("\t".isspace()) # True

Good for checking if user input is empty or blank.

**5. Splitting & Alignment**

**split()**

Splits the string into a **list of words or parts**, based on a separator.

str2 = "Silver Spoon"

print(str2.split(" "))

**Output:**

['Silver', 'Spoon']

Useful when working with CSV, logs, or user input.

**6. Prefix/Suffix Checks**

**startswith() / endswith()**

Check if a string **starts or ends** with a specific substring.

str1 = "Python is fun!"

print(str1.startswith("Py")) # True

print(str1.endswith("!")) # True

Great for filtering data or validating text patterns.