**Experiment Number 5**

**Title - Program to Search whether all characters in one string are available into another string**

**Theory-**

A string is a data structure in Python that represents a sequence of characters. Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes. The computer does not understand the characters; internally, it stores manipulated character as the combination of the 0's and 1's. Each character is encoded in the ASCII or Unicode character. Python strings are also called the collection of Unicode characters. It is an immutable data type, meaning that once you have created a string, you cannot change it. Strings are used widely in many different applications, such as storing and manipulating text data, representing names, addresses, and other types of data that can be represented as text. Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string.

**Creating String in Python**

We can create a string by enclosing the characters in single-quotes or double- quotes. Python also provides triple-quotes to represent the string, but it is generally used for multiline string or **docstrings**.

**Access String Characters in Python**

We can access the characters in a string in three ways.

* **Indexing:** One way is to treat strings as a [list](https://www.programiz.com/python-programming/list) and use index values. For example,

greet = 'hello'

# access 1st index element

print(greet[1]) # "e"

* **Negative Indexing:** Similar to a list, Python allows [negative indexing](https://www.programiz.com/python-programming/list#negative-indexing) for its strings. For example,

greet = 'hello'

# access 4th last element

print(greet[-4]) # "e"

* **Slicing:** Access a range of characters in a string by using the slicing operator colon :. For example,

greet = 'Hello'

# access character from 1st index to 3rd index

print(greet[1:4]) # "ell"

## String Special Operators

Assume string variable **a** holds 'Hello' and variable **b** holds 'Python', then –

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Concatenation - Adds values on either side of the operator | a + b will give HelloPython |
| \* | Repetition - Creates new strings, concatenating multiple copies of the same string | a\*2 will give -HelloHello |
| [] | Slice - Gives the character from the given index | a[1] will give e |
| [ : ] | Range Slice - Gives the characters from the given range | a[1:4] will give ell |
| in | Membership - Returns true if a character exists in the given string | H in a will give 1 |
| not in | Membership - Returns true if a character does not exist in the given string | M not in a will give 1 |
| r/R | Raw String - Suppresses actual meaning of Escape characters. The syntax for raw strings is exactly the same as for normal strings with the exception of the raw string operator, the letter "r," which precedes the quotation marks. The "r" can be lowercase (r) or uppercase (R) and must be placed immediately preceding the first quote mark. | Print(r'\n') prints \n and print(R'\n') prints \n |

## Python Strings are immutable

In Python, strings are immutable. That means the characters of a string cannot be changed. For example, the below code will produce an error

message = 'Hola Amigos'

message[0] = 'H'

print(message)

## Built-in String Methods

Python includes the following built-in methods to manipulate strings –

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [capitalize()](https://www.tutorialspoint.com/python/string_capitalize.htm)  Capitalizes first letter of string |
| 2 | [center(width, fillchar)](https://www.tutorialspoint.com/python/string_center.htm)  Returns a space-padded string with the original string centered to a total of width columns. |
| 3 | [count(str, beg= 0,end=len(string))](https://www.tutorialspoint.com/python/string_count.htm)  Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given. |
| 4 | [decode(encoding='UTF-8',errors='strict')](https://www.tutorialspoint.com/python/string_decode.htm)  Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding. |
| 5 | [encode(encoding='UTF-8',errors='strict')](https://www.tutorialspoint.com/python/string_encode.htm)  Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'. |
| 6 | [endswith(suffix, beg=0, end=len(string))](https://www.tutorialspoint.com/python/string_endswith.htm)  Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise. |
| 7 | [expandtabs(tabsize=8)](https://www.tutorialspoint.com/python/string_expandtabs.htm)  Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided. |
| 8 | [find(str, beg=0 end=len(string))](https://www.tutorialspoint.com/python/string_find.htm)  Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise. |
| 9 | [index(str, beg=0, end=len(string))](https://www.tutorialspoint.com/python/string_index.htm)  Same as find(), but raises an exception if str not found. |
| 10 | [isalnum()](https://www.tutorialspoint.com/python/string_isalnum.htm)  Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise. |
| 11 | [isalpha()](https://www.tutorialspoint.com/python/string_isalpha.htm)  Returns true if string has at least 1 character and all characters are alphabetic and false otherwise. |
| 12 | [isdigit()](https://www.tutorialspoint.com/python/string_isdigit.htm)  Returns true if string contains only digits and false otherwise. |
| 13 | [islower()](https://www.tutorialspoint.com/python/string_islower.htm)  Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise. |
| 14 | [isnumeric()](https://www.tutorialspoint.com/python/string_isnumeric.htm)  Returns true if a unicode string contains only numeric characters and false otherwise. |
| 15 | [isspace()](https://www.tutorialspoint.com/python/string_isspace.htm)  Returns true if string contains only whitespace characters and false otherwise. |
| 16 | [istitle()](https://www.tutorialspoint.com/python/string_istitle.htm)  Returns true if string is properly "titlecased" and false otherwise. |
| 17 | [isupper()](https://www.tutorialspoint.com/python/string_isupper.htm)  Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise. |
| 18 | [join(seq)](https://www.tutorialspoint.com/python/string_join.htm)  Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string. |
| 19 | [len(string)](https://www.tutorialspoint.com/python/string_len.htm)  Returns the length of the string |
| 20 | [ljust(width[, fillchar])](https://www.tutorialspoint.com/python/string_ljust.htm)  Returns a space-padded string with the original string left-justified to a total of width columns. |
| 21 | [lower()](https://www.tutorialspoint.com/python/string_lower.htm)  Converts all uppercase letters in string to lowercase. |
| 22 | [lstrip()](https://www.tutorialspoint.com/python/string_lstrip.htm)  Removes all leading whitespace in string. |
| 23 | [maketrans()](https://www.tutorialspoint.com/python/string_maketrans.htm)  Returns a translation table to be used in translate function. |
| 24 | [max(str)](https://www.tutorialspoint.com/python/string_max.htm)  Returns the max alphabetical character from the string str. |
| 25 | [min(str)](https://www.tutorialspoint.com/python/string_min.htm)  Returns the min alphabetical character from the string str. |
| 26 | [replace(old, new [, max])](https://www.tutorialspoint.com/python/string_replace.htm)  Replaces all occurrences of old in string with new or at most max occurrences if max given. |
| 27 | [rfind(str, beg=0,end=len(string))](https://www.tutorialspoint.com/python/string_rfind.htm)  Same as find(), but search backwards in string. |
| 28 | [rindex( str, beg=0, end=len(string))](https://www.tutorialspoint.com/python/string_rindex.htm)  Same as index(), but search backwards in string. |
| 29 | [rjust(width,[, fillchar])](https://www.tutorialspoint.com/python/string_rjust.htm)  Returns a space-padded string with the original string right-justified to a total of width columns. |
| 30 | [rstrip()](https://www.tutorialspoint.com/python/string_rstrip.htm)  Removes all trailing whitespace of string. |
| 31 | [split(str="", num=string.count(str))](https://www.tutorialspoint.com/python/string_split.htm)  Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given. |
| 32 | [splitlines( num=string.count('\n'))](https://www.tutorialspoint.com/python/string_splitlines.htm)  Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed. |
| 33 | [startswith(str, beg=0,end=len(string))](https://www.tutorialspoint.com/python/string_startswith.htm)  Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise. |
| 34 | [strip([chars])](https://www.tutorialspoint.com/python/string_strip.htm)  Performs both lstrip() and rstrip() on string. |
| 35 | [swapcase()](https://www.tutorialspoint.com/python/string_swapcase.htm)  Inverts case for all letters in string. |
| 36 | [title()](https://www.tutorialspoint.com/python/string_title.htm)  Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase. |
| 37 | [translate(table, deletechars="")](https://www.tutorialspoint.com/python/string_translate.htm)  Translates string according to translation table str(256 chars), removing those in the del string. |

|  |  |
| --- | --- |
| 38 | [upper()](https://www.tutorialspoint.com/python/string_upper.htm)  Converts lowercase letters in string to uppercase. |
| 39 | [zfill (width)](https://www.tutorialspoint.com/python/string_zfill.htm)  Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero). |
| 40 | [isdecimal()](https://www.tutorialspoint.com/python/string_isdecimal.htm)  Returns true if a unicode string contains only decimal characters and false otherwise. |

### ****Advantages of String in Python:****

* Strings are used at a larger scale i.e. for a wide areas of operations such as storing and manipulating text data, representing names, addresses, and other types of data that can be represented as text.
* Python has a rich set of string methods that allow you to manipulate and work with strings in a variety of ways. These methods make it easy to perform common tasks such as converting strings to uppercase or lowercase, replacing substrings, and splitting strings into lists.
* Strings are immutable, meaning that once you have created a string, you cannot change it. This can be beneficial in certain situations because it means that you can be confident that the value of a string will not change unexpectedly.
* Python has built-in support for strings, which means that you do not need to import any additional libraries or modules to work with strings. This makes it easy to get started with strings and reduces the complexity of your code.
* Python has a concise syntax for creating and manipulating strings, which makes it easy to write and read code that works with strings.

### ****Drawbacks of String in Python:****

* When we are dealing with large text data, strings can be inefficient. For instance, if you need to perform a large number of operations on a string, such as replacing substrings or splitting the string into multiple substrings, it can be slow and consume a lot resources.
* Strings can be difficult to work with when you need to represent complex data structures, such as lists or dictionaries. In these cases, it may be more efficient to use a different data type, such as a list or a dictionary, to represent the data.

Exercise –

1. Write a program to print the initials of a name with last name in full.
2. Write a program to check if both halves of the string have same set of characters in Python.
3. Write a program to reverse words in a given String in Python.
4. Write a program to remove all duplicates words from a given sentence.
5. Write a program to sort words of sentence in ascending order.
6. Write a program to accept the strings which contains all vowels.
7. Write a program to check if a string contains any special character