

CONTENTS



(Learning Outcomes) 😊

STRING in Python

XI CS 2020-21

Introduction : Definition of String, Creation String, Traversal of String

Operations on String – concatenation/joining, repetition/replication, membership;

Functions/methods–

len(), capitalize(), title(), upper(), lower(), count(), find(), index(), isalnum(), islower(), isupper(), isspace(), isalpha(), isdigit(), split(), partition(), strip(), lstrip(), rstrip(), replace();

Extra Functions not in the curriculum:

startswith(), endswith(), join(), swapcase()

String slicing

EXTRA PROGRAMS ON STRING: *Palindrome of a string*

STRING

INTRODUCTION

What is a STRING ?

CONTENTS

→ Strings Intro:

String Definition

Indexing in a String

Creation of a String

Traversal of a String.

→ Operations on a String

→ Functions/methods

→ Strings Slicing;

```
>>> 'hi'
'hi'
>>> "hi"
'hi'
>>> '''hi'''
'hi'
>>> """hi"""
'hi'
```

- String is a sequence of characters, which is enclosed between either single (' ') or double quotes (" ") or triple quotation(" " " ").
- Python treats both single and double quotes same.

→ Each item/element has its own index number
→ Index of first item is 0 and the last item is n- 1. Here n is number of items in a String.
→ Strings are immutable (i.e. unmodifiable) .
Thus, every time we perform something on the string that changes it, Python will internally create a new string rather than modifying the old string in place.
→ Traversing through a string refers to iterating through the elements of a string one character at a time.

STRINGS ARE IMMUTABLE

CONTENTS

→ Strings Intro: **String Definition**

Indexing in a String
Creation of a String

Traversal of a String.

→ Operations on a String

→ Functions/methods

→ Strings Slicing;

Strings are immutable (i.e. unmodifiable) .

REASON: Every time we perform something on the string that changes it, Python will internally create a new string rather than modifying the old string in place.

```
>>> s="India"
>>> id(s)
50819712
>>> s=s+'n'
>>> s
'Indian'
>>> id(s)
50819552
```

NOTE: New string created at a different id after modifying the string **S**

DATA TYPE: STRING in Python

CONTENTS

→ Strings Intro:

String Definition

Indexing in a String

Creation of a String

Traversal of a String.

→ Operations on a String

→ Functions/methods

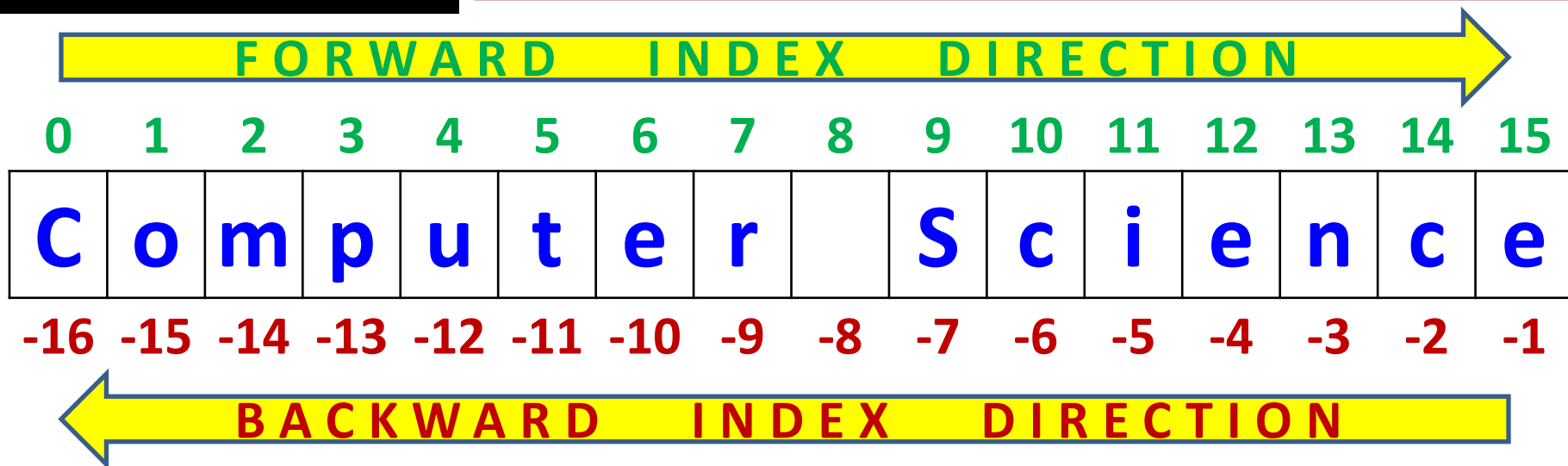
→ Strings Slicing;

The index (also called subscript) is the numbered position of a letter in the string. In python, indices begin from

0,1,2,... upto (length-1) in the *forward direction* and

-1,-2,-3,..... (-length) in the *backward direction*.

where length is the length of the string.




Creation of String

Python allows to have two string types:

1) Single line strings (Basic strings)

Text enclosed in single quote 'India' or in double quotes "Bharat" are normally single line strings i.e. they must terminate in one line.



```
>>> s='India'
>>> s
'India'
>>> s="Bharat"
>>> s
'Bharat'
```

2) Multiline strings

Text spread across multiple lines as one single string.

Two ways of treating multiline strings:-

(i) By adding a backslash at the end of normal single quote/double quote strings.



```
>>> s="Welcome\
to the\
World of\
wonders"
>>> s
'Welcometo theWorld ofwonders'
```

(ii) By typing the text in triple quotation or apostrophe mark (No backslash needed at the end of line)



```
>>> Str='''Welcome
to the
world of wonders.'''
>>> Str
'Welcome\nto the\nworld of wonders.'
>>> A="""Welcome
to the
world of wonders."""
>>> A
'Welcome\nto the\nworld of wonders.'
```

Traversal of a STRING

CONTENTS

→ Strings Intro:

String Definition

Indexing in a String

Creation of a String

Traversal of a String

→ Operations on a String

→ Functions/methods

→ Strings Slicing;

Traversing through string

Traversing refers to iterating through the elements of a string one character at a time.

```
>>> s="Hand wash"
>>> for i in s:
    print(i)
```

H
a
n
d

w
a
s
h

```
>>> s="Use Sanitizer"
>>> for i in s:
    print(i,end="")
```

Use Sanitizer

OPERATIONS ON STRING

Concatenation / Joining of STRING

CONTENTS

→ Strings Intro:

→ Operations on String concatenation/ joining

,repetition/replication,
membership

→ Functions/methods

→ Strings Slicing;



operator

is used for concatenation/joining of strings

```
>>> "I"+"love"+"my"+"Country"
'IlovelyCountry'
>>> "I"+" "+"love"+" "+"my"+" "+"Country"
'I love my Country'
```

```
>>> '2'+'1'
'21'
>>> "A"+"5"
'A5'
>>> '569'+'WEST'
'569WEST'
>>> 2+1
3
```

```
>>> "KV1"+"KPA"
'KV1KPA'
```

```
>>> "B"+5
```

```
Traceback (most recent call last):
```

```
File "<pyshell#15>", line 1, in <module>
```

```
"B"+5
```

```
TypeError: can only concatenate str (not "int") to str
```

Repetition or Replication STRING

CONTENTS

→ Strings Intro:

→ Operations on String

concatenation/
joining,

repetition/ replication,

membership

→ Functions/methods

→ Strings Slicing;

***** operator

is used for concatenation/joining of strings

```
>>> "India"*4
'IndiaIndiaIndiaIndia'
>>> "Wash hand"*3
'Wash handWash handWash hand'
>>> "502"*6
'502502502502502502'
```

```
>>> 4*"India"
'IndiaIndiaIndiaIndia'
>>> 3*"2"
'222'
```

```
>>> 2*3
6
>>> "2"*3
'222'
```

```
>>> "KVS"*3
'KVSKVSKVS'
```

```
>>> "2"*"3"
```

Traceback (most recent call last):

File "<pyshell#26>", line 1, in <module>

"2"*"3"

TypeError: can't multiply sequence by non-int of type 'str'

MEMBERSHIP (**in/not in**) STRING

CONTENTS

→ Strings Intro:

→ Operations on String

concatenation/ joining
,repetition/replication,

membership

,Comparison of String

→ Functions/methods

→ Strings Slicing;

Membership operator tests for Membership in a sequence. Returns Boolean **True** or **False**

OPERATOR	DESCRIPTION
in	Results to True value if it finds a variable at the LHS of in operator in the sequence mentioned in the RHS of the in operator, else it returns False
not in	Results to True value if it does not find a variable at the LHS of in operator in the sequence mentioned in the RHS of the in operator, else it returns False

```
>>> "a" in "India"
```

```
True
```

```
>>> "A" in "India"
```

```
False
```

```
>>> "A" in "IndiA"
```

```
True
```

```
>>> "b" in "India"
```

```
False
```

```
>>> "a" not in "India"
```

```
False
```

```
>>> "Hi" not in "India"
```

```
True
```

```
>>> substr="India"
```

```
>>> str="India is my country"
```

```
>>> substr in str
```

```
True
```

```
>>> substr not in str
```

```
False
```

Comparison of STRING

CONTENTS

Comparison Operators <,>,<=,>=,!=,==

→ Strings Intro:

→ Operations on String

concatenation/ joining
,repetition/replication,
membership

, Comparison of String

→ Functions/methods
→ Strings Slicing;

Characters	Ordinal values (Unicode values)
"0" to "9"	48 to 57
"A" to "Z"	65 to 90
"a" to "z"	97 to 122

FOR STRING DATA TYPES →

In ASCII value,

A=65,B=66,C=67.....Z=90

a=97,b=98,c=99..... z=122

' ' = 32(*blank or white space*)

```
>>> 'A'=='a'
False
>>> 'Map'=='map'
False
>>> 'Map'=='Map'
True
>>> 'Map'<='map'
True
```

```
>>> 'KV'!='kv'
True
>>> 'KV'>'Kv'
False
>>> 'one'==' one'
False
>>> 'ANT'<='ANTS'
True
```

NOTE:

ord() function is used to get the ASCII value of a character

chr() function is used to get the equivalent character of the ASCII no.

NOTE:

ASCII stands for American Standard Code for Information Interchange.

```
>>> ord('a')
97
>>> chr(65)
'A'
```

STRING FUNCTIONS/ METHODS

STRING FUNCTION: len()

CONTENTS

→ Strings Intro:

→ Operations on String

→ **Functions/
methods**

len(), capitalize(), title(),
count(), find(), index(),
upper(), lower(), islower(),
isupper(), isspace(), isdigit(),
isalpha(), isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(), endswith(),
join(), swapcase()

→ Strings Slicing;

len() → function counts the no. of
elements/items/characters in the String.

```
>>> str1='India'
>>> str2='wins'
>>> str3='T20'
>>> str4='MATCH'
>>> str5='2021'
>>> str6='Wi-Fi'
>>> str7='season2'
>>> str8='  '
>>> str9=''
>>> str10='-+*/'
```

```
>>> len(str1)
5
>>> len(str2)
4
>>> len(str3)
3
>>> len(str4)
5
>>> len(str5)
4
>>> len(str6)
5
>>> len(str7)
7
>>> len(str8)
2
>>> len(str9)
0
>>> len(str10)
4
```

```
>>> s="I love my country - India"
>>> len(s)
25
```

STRING FUNCTION: **capitalize()**

CONTENTS

→ Strings Intro:
→ Operations on String
→ **Functions/
methods**

len(), **capitalize()**,
title(), count(),
find(), index(), upper(),
lower(), islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(), partition(),
strip(), lstrip(), rstrip(),
replace(), startswith(),
endswith(), join(),
swapcase()

→ Strings Slicing;

capitalize() → Returns a copy of the *string* with its first character capitalized.

```
>>> str1='India'  
>>> str2='wins'  
>>> str3='T20'  
>>> str4='MATCH'  
>>> str5='2021'  
>>> str6='Wi-Fi'  
>>> str7='season2'  
>>> str8=' '  
>>> str9=''  
>>> str10='-+*/'
```

```
>>> str1.capitalize()  
'India'  
>>> str2.capitalize()  
'Wins'  
>>> str3.capitalize()  
'T20'  
>>> str4.capitalize()  
'Match'  
>>> str5.capitalize()  
'2021'  
>>> str6.capitalize()  
'Wi-fi'  
>>> str7.capitalize()  
'Season2'  
>>> str8.capitalize()  
' '  
>>> str9.capitalize()  
' '  
>>> str10.capitalize()  
'-+*/'
```

```
>>> s="I love my country - India"  
>>> s.capitalize()  
'I love my country - india'
```

Only the character at index 0 of the string is capitalized.
That is only the first character of the full string is capitalized.

STRING FUNCTION: title()

CONTENTS

→ Strings Intro:

→ Operations on String

→ **Functions/
methods**

len(), capitalize(),

title(), count(),

find(), index(), upper(),

lower(), islower(), isupper(),

isspace(), isdigit(), isalpha(),

isalnum(), split(),

partition(), strip(), lstrip(),

rstrip(), replace(),

startswith(), endswith(),

join(), swapcase()

→ Strings Slicing;

title() → Converts the first character of each word to upper case

```
>>> str1='India'
>>> str2='wins'
>>> str3='T20'
>>> str4='MATCH'
>>> str5='2021'
>>> str6='Wi-Fi'
>>> str7='season2'
>>> str8=' '
>>> str9=''
>>> str10='-+*/'
```

```
>>> str1.title()
'India'
>>> str2.title()
'Wins'
>>> str3.title()
'T20'
>>> str4.title()
'Match'
>>> str5.title()
'2021'
>>> str6.title()
'Wi-Fi'
>>> str7.title()
'Season2'
>>> str8.title()
' '
>>> str9.title()
''
>>> str10.title()
'-+*/'
```

```
>>> s="I love my country - India"
>>> s.title()
'I Love My Country - India'
```

Every word's first character is shown in Capital letters.

STRING FUNCTION: count()

CONTENTS

→ Strings Intro:

→ Operations on String

→ **Functions/
methods**

len(), capitalize(), title(),

count(),

find(), index(), upper(),

lower(), islower(), isupper(),

isspace(), isdigit(), isalpha(),

isalnum(), split(), partition(),

strip(), lstrip(), rstrip(),

replace(), startswith(),

endswith(), join(),

swapcase()

→ Strings Slicing;

count() → Returns the number of times a specified value occurs in a string

```
>>> STR="Use handwash Use sanitizer Use Mask"
```

```
>>> STR.count("Use")
```

```
3
```

'Use' occurred 3 times in the given string STR

```
>>> STR.count("Mask")
```

```
1
```

'Mask' occurred 1 time in the given string STR

```
>>> STR.count("India")
```

```
0
```

'India' is not there in the given string STR. So, 0 times.

STRING FUNCTION: find()

CONTENTS

→ Strings Intro:
→ Operations on String

→ Functions/ methods

len(), capitalize(), title(),

count(), **find()**,

index(), upper(), lower(),
islower(), isupper(), isspace(),
isdigit(), isalpha(), isalnum(),

split(), partition(), strip(),
lstrip(), rstrip(), replace(),
startswith(), endswith(), join(),
swapcase()

→ Strings Slicing;

find() → Searches the string for a specified value and returns the position of where it was found

```
>>> s='use handwash , use sanitizer, use mask'
>>> substr='use'
>>> s.find(substr)
0
>>> s.find(substr,3,30)
15
>>> s.find(substr,5,50)
15
>>> s.find(substr,5,10)
-1
```

Substr 'USE' is found at index 0

Substr 'USE' is found at index 15 between index no. 3 and 30

Substr 'USE' is found at index 15 between index no. 3 and 30

Substr 'USE' is NOT found at between index no. 5 and 10. So, -1 is returned when it is not found.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37		
u	s	e		h	a	n	d	w	a	s	h		,		u	s	e		s	a	n	i	t	i	z	e	r		,		u	s	e			m	a	s	k

STRING FUNCTION: index()

CONTENTS

→ Strings Intro:
→ Operations on String
→ **Functions/
methods**

len(), capitalize(), title(),
count(), find(),

index(), upper(),
lower(), islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(), partition(),
strip(), lstrip(), rstrip(),
replace(), startswith(),
endswith(), join(),
swapcase()

→ Strings Slicing;

index() → Searches the string for a specified value and returns the position of where it was found

```
>>> STR="Use handwash Use sanitizer Use Mask"
>>> STR.index("Use")
0
>>> STR.index("h")
4
>>> STR.index("as")
9
>>> STR.index("sanitizer")
17
>>> STR.index("safety")
Traceback (most recent call last):
  File "<pyshell#69>", line 1, in <module>
    STR.index("safety")
ValueError: substring not found
```

STRING FUNCTION: upper()

CONTENTS

→ Strings Intro:
→ Operations on String
→ **Functions/
methods**

len(), capitalize(), title(),
count(), find(),
index(), **upper()**,
lower(), islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(), endswith(),
join(), swapcase()
→ Strings Slicing;

upper() →

Returns a copy of the *string* converted to upper case

```
>>> str1='India'  
>>> str2='wins'  
>>> str3='T20'  
>>> str4='MATCH'  
>>> str5='2021'  
>>> str6='Wi-Fi'  
>>> str7='season2'  
>>> str8=' '  
>>> str9=''   
>>> str10='-+*/'
```

```
>>> str1.upper()  
'INDIA'  
>>> str2.upper()  
'WINS'  
>>> str3.upper()  
'T20'  
>>> str4.upper()  
'MATCH'  
>>> str5.upper()  
'2021'  
>>> str6.upper()  
'WI-FI'  
>>> str7.upper()  
'SEASON2'  
>>> str8.upper()  
' '  
>>> str9.upper()  
' '  
>>> str10.upper()  
'-+*/'
```

STRING FUNCTION: **lower()**

CONTENTS

→ Strings Intro:

→ Operations on String

→ **Functions/
methods**

len(), capitalize(), title(),
count(), find(),
index(), upper(),

lower(), islower(),
isupper(), isspace(), isdigit(),
isalpha(), isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(), endswith(),
join(), swapcase()

→ Strings Slicing;

lower() →

Returns a copy of the *string* converted to lowercase.

```
>>> str1='India'
```

```
>>> str2='wins'
```

```
>>> str3='T20'
```

```
>>> str4='MATCH'
```

```
>>> str5='2021'
```

```
>>> str6='Wi-Fi'
```

```
>>> str7='season2'
```

```
>>> str8='  '
```

```
>>> str9=''
```

```
>>> str10='-+*/'
```

```
>>> str1.lower()  
'india'
```

```
>>> str2.lower()  
'wins'
```

```
>>> str3.lower()  
't20'
```

```
>>> str4.lower()  
'match'
```

```
>>> str5.lower()  
'2021'
```

```
>>> str6.lower()  
'wi-fi'
```

```
>>> str7.lower()  
'season2'
```

```
>>> str8.lower()  
'  '
```

```
>>> str9.lower()  
''
```

```
>>> str10.lower()  
'-+*/'
```

STRING FUNCTION: **islower()**

CONTENTS

→ Strings Intro:

→ Operations on a String

→ **Functions/**

methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),

islower(), isupper(),

isspace(), isdigit(),
isalpha(), isalnum(),
split(), partition(), strip(),
lstrip(),rstrip(), replace(),
startswith(), endswith(),
join(), swapcase()

→ Strings Slicing;

Returns **True** if all cased characters in the *string* are lowercase. There must be at least one character. It returns **False** otherwise.

****All the characters must be in small letters for the function to return True.**

**** Blank spaces or any other special characters or empty string returns False.**

```
>>> str1='India'
>>> str2='wins'
>>> str3='T20'
>>> str4='MATCH'
>>> str5='2021'
>>> str6='Wi-Fi'
>>> str7='season2'
>>> str8=' '
>>> str9=''
>>> str10='-+*/'
```

```
>>> str1.islower()
False
>>> str2.islower()
True
>>> str3.islower()
False
>>> str4.islower()
False
>>> str5.islower()
False
>>> str6.islower()
False
>>> str7.islower()
True
>>> str8.islower()
False
>>> str9.islower()
False
>>> str10.islower()
False
```

STRING FUNCTION: isupper()

CONTENTS

→ Strings Intro:
→ Operations on a String
→ **Functions/
methods**

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), **isupper()**,
isspace(), isdigit(),
isalpha(), isalnum(),
split(), partition(), strip(),
lstrip(), rstrip(), replace(),
startswith(), endswith(),
join(), swapcase()
→ Strings Slicing;

Returns **True** if all cased characters in the *string* are uppercase.
There must be at least one character. It returns **False** otherwise.
****All the characters must be in CAPITAL letters for the function to return True.**
**** Blank spaces or any other special characters or empty string returns False.**

```
>>> str1='India'  
>>> str2='wins'  
>>> str3='T20'  
>>> str4='MATCH'  
>>> str5='2021'  
>>> str6='Wi-Fi'  
>>> str7='season2'  
>>> str8=' '  
>>> str9=''  
>>> str10='-+*/'
```

```
>>> str1.isupper()  
False  
>>> str2.isupper()  
False  
>>> str3.isupper()  
True  
>>> str4.isupper()  
True  
>>> str5.isupper()  
False  
>>> str6.isupper()  
False  
>>> str7.isupper()  
False  
>>> str8.isupper()  
False  
>>> str9.isupper()  
False  
>>> str10.isupper()  
False
```

STRING FUNCTION: isspace()

CONTENTS

→ Strings Intro:
→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),

isspace(), isdigit(),
isalpha(), isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(), endswith(),
join(), swapcase()

→ Strings Slicing;

isspace() → Returns True if all characters in the string are whitespaces.

**** Any string other than only whitespaces returns False.**

**** An empty string also returns False**

```
>>> str1='India'
```

```
>>> str2='wins'
```

```
>>> str3='T20'
```

```
>>> str4='MATCH'
```

```
>>> str5='2021'
```

```
>>> str6='Wi-Fi'
```

```
>>> str7='season2'
```

```
>>> str8=' '
```

```
>>> str9=''
```

```
>>> str10='-+*/'
```

```
>>> str1.isspace()  
False
```

```
>>> str2.isspace()  
False
```

```
>>> str3.isspace()  
False
```

```
>>> str4.isspace()  
False
```

```
>>> str5.isspace()  
False
```

```
>>> str6.isspace()  
False
```

```
>>> str7.isspace()  
False
```

```
>>> str8.isspace()  
True
```

```
>>> str9.isspace()  
False
```

```
>>> str10.isspace()  
False
```


STRING FUNCTION: isdigit()

CONTENTS

→ Strings Intro:

→ Operations on a String

→ **Functions/
methods**

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(), isspace(),

isdigit(), isalpha(),
isalnum(), split(), partition(),
strip(), lstrip(), rstrip(),
replace(), startswith(),
endswith(), join(), swapcase()

→ Strings Slicing;

Returns True if all characters in the string are digits
**All the characters must be digits for the function to return True.

** Blank spaces or any other special characters or empty string returns False.

```
>>> str1='India'
```

```
>>> str2='wins'
```

```
>>> str3='T20'
```

```
>>> str4='MATCH'
```

```
>>> str5='2021'
```

```
>>> str6='Wi-Fi'
```

```
>>> str7='season2'
```

```
>>> str8=' '
```

```
>>> str9=''
```

```
>>> str10='-+*/'
```

```
>>> str1.isdigit()  
False
```

```
>>> str2.isdigit()  
False
```

```
>>> str3.isdigit()  
False
```

```
>>> str4.isdigit()  
False
```

```
>>> str5.isdigit()  
True
```

```
>>> str6.isdigit()  
False
```

```
>>> str7.isdigit()  
False
```

```
>>> str8.isdigit()  
False
```

```
>>> str9.isdigit()  
False
```

```
>>> str10.isdigit()  
False
```

STRING FUNCTION: isalpha()

CONTENTS

→ Strings Intro:
→ Operations on a String
→ **Functions/**
methods
len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(), isspace(),
isdigit(), **isalpha()**,
isalnum(), split(), partition(),
strip(), lstrip(), rstrip(),
replace(), startswith(),
endswith(), join(), swapcase()
→ Strings Slicing;

Returns True if all characters in the string are in the alphabet

****** All the characters must be alphabet for the function to return True.

****** Blank spaces or any other special characters or digits with alphabet or empty string returns False.

```
>>> str1='India'  
>>> str2='wins'  
>>> str3='T20'  
>>> str4='MATCH'  
>>> str5='2021'  
>>> str6='Wi-Fi'  
>>> str7='season2'  
>>> str8=' '  
>>> str9=''  
>>> str10='-+*/'
```

```
>>> str1.isalpha()  
True  
>>> str2.isalpha()  
True  
>>> str3.isalpha()  
False  
>>> str4.isalpha()  
True  
>>> str5.isalpha()  
False  
>>> str6.isalpha()  
False  
>>> str7.isalpha()  
False  
>>> str8.isalpha()  
False  
>>> str9.isalpha()  
False  
>>> str10.isalpha()  
False
```

STRING FUNCTION: **isalnum()**

CONTENTS

→ Strings Intro:
→ Operations on a String
→ **Functions/
methods**

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(), isspace(),
isdigit(), isalpha(),

isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(), endswith(),
join(), swapcase()

→ Strings Slicing;

Returns True if all characters in the string are alphanumeric

******All the characters must be alphabet or number or a combination of alphabet and number for the function to return True.

****** Blank spaces or presence of any other special characters or empty string returns False.

```
>>> str1='India'  
>>> str2='wins'  
>>> str3='T20'  
>>> str4='MATCH'  
>>> str5='2021'  
>>> str6='Wi-Fi'  
>>> str7='season2'  
>>> str8=' '  
>>> str9=''  
>>> str10='-+*/'
```

```
>>> str1.isalnum()  
True  
>>> str2.isalnum()  
True  
>>> str3.isalnum()  
True  
>>> str4.isalnum()  
True  
>>> str5.isalnum()  
True  
>>> str6.isalnum()  
False  
>>> str7.isalnum()  
True  
>>> str8.isalnum()  
False  
>>> str9.isalnum()  
False  
>>> str10.isalnum()  
False
```

STRING FUNCTION: split()

CONTENTS

→ Strings Intro:

→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(),
isalpha(), isalnum(),

split(), partition(),
strip(), lstrip(),rstrip(),
replace(), startswith(),
endswith(), join(),
swapcase()

→ Strings Slicing;

split() → Splits the string at the specified separator, and **returns a List**.

- i) The default character for splitting is whitespace.
- ii) One can specify their own character on the basis of which splitting is to be done.

```
>>> S1="I love my country India !"
>>> S1.split()
['I', 'love', 'my', 'country', 'India', '!']
>>> S1.split("-")
['I love my country India !']
>>> S2="I-love-my-India"
>>> S2.split("-")
['I', 'love', 'my', 'India']
```

Split returns a list by separating the elements through whitespaces by default

Here, Split returns a list by separating the elements through '-'

STRING FUNCTION: **partition()**

CONTENTS

→ Strings Intro:

→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(),
isalpha(), isalnum(),
split(), **partition()**,
strip(), lstrip(), rstrip(),
replace(), startswith(),
endswith(), join(),
swapcase()

→ Strings Slicing;

Returns a Tuple where the string is parted into three parts.

The parts returned as the elements of the Tuple are:

- i) Portion before the argument from the whole string
- ii) The argument sent with the partition() function
- iii) Portion after the argument in the whole string.

```
>>> S1="I love my country India !"  
>>> S1.partition("my")  
( 'I love ', 'my', ' country India !' )
```

(i) Portion before
argument

(ii) The
argument

(iii) Portion after argument

Here, "my" is the argument sent with the partition function.

```
>>> S="I love my school"  
>>> S.partition()  
Traceback (most recent call last):  
  File "<pyshell#1>", line 1, in <module>  
    S.partition()  
TypeError: partition() takes exactly one argument (0 given)
```

**Partition()
needs atleast
one argument**

STRING FUNCTION: **strip()**

Returns a trimmed version of the string. Trimming is done only when cases are matched with any of the characters in the argument from both sides of the string (beginning and end).

CONTENTS

→ Strings Intro:

→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(), partition(),

strip(), lstrip(), rstrip(),
replace(), startswith(),
endswith(), join(),
swapcase()

→ Strings Slicing;

```
>>> Str='   know python bytes   '
>>> Str.strip()
'know python bytes'
>>> Str
'   know python bytes   '
```

By default, if there is no argument mentioned then, whitespaces are removed from both the beginning of the string and end of the whole string. But, no change in the original string as Strings are immutable. Cases matched.

```
>>> S="Atmanirbhar Bharat"
>>> S.strip('A')
'tmanirbhar Bharat'
```

~~Atmanirbhar Bharat~~

```
>>> S.strip('Atar')
'manirbhar Bh'
```

~~Atmanirbhar Bharat~~

```
>>> S.strip('mAta')
'nirbhar Bhar'
```

~~Atmanirbhar Bharat~~

```
>>> S.strip('manirt')
'Atmanirbhar Bh'
```

~~Atmanirbhar Bharat~~

```
>>> S.strip('nira')
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.strip('Bhar')
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.strip('nmtAarhB')
'irbhar '
```

~~Atmanirbhar Bharat~~

STRING FUNCTION: **lstrip()**

CONTENTS

→ Strings Intro:

→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(),
isalpha(), isalnum(),
split(), partition(), strip(),

lstrip(), rstrip(),

replace(), startswith(),
endswith(), join(),
swapcase()

→ Strings Slicing;

Returns a LEFT trimmed version of the string. Trimming is done only when cases are matched with any of the characters in the argument from LEFT side of the string (beginning).

```
>>> S=" know python bytes "  
>>> S.lstrip()  
'know python bytes '  
>>> S  
' know python bytes '
```

By default, if there is no argument mentioned then, whitespaces are removed from the beginning of the string only (LEFT side). But, no change in the original string as Strings are immutable. **Cases matched.**

```
>>> S='Atmanirbhar Bharat'
```

```
>>> S.lstrip('A')  
'tmanirbhar Bharat'
```

~~A~~tmanirbhar Bharat

```
>>> S.lstrip('Atr')  
'manirbhar Bharat'
```

~~A~~~~r~~tmanirbhar Bharat

```
>>> S.lstrip('mAtr')  
'nirbhar Bharat'
```

~~A~~~~r~~~~t~~manirbhar Bharat

```
>>> S.lstrip('manirt')  
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.lstrip('nira')  
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.lstrip('Bhar')  
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.lstrip('nmtAarhB')  
'irbhar Bharat'
```

~~A~~~~r~~~~t~~~~m~~~~n~~manirbhar Bharat

STRING FUNCTION: **rstrip()**

CONTENTS

→ Strings Intro:

→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(),
isalpha(), isalnum(),
split(), partition(), strip(),
lstrip(), **rstrip()**,
replace(), startswith(),
endswith(), join(),
swapcase()

→ Strings Slicing;

Returns a RIGHT trimmed version of the string. Trimming is done only when cases are matched with any of the characters in the argument from RIGHT side of the string (end).

```
>>> S="  know python bytes  "
>>> S.rstrip()
' know python bytes'
>>> S
' know python bytes  '
```

By default, if there is no argument mentioned then, whitespaces are removed from the end of the string only (RIGHT side). But, no change in the original string as Strings are immutable. **Cases matched.**

```
>>> S='Atmanirbhar Bharat'
```

```
>>> S.rstrip('A')
```

```
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.rstrip('Aar')
```

```
'Atmanirbhar Bh'
```

Atmanirbhar Bharat

```
>>> S.rstrip('mAta')
```

```
'Atmanirbhar Bhar'
```

Atmanirbhar Bharat

```
>>> S.rstrip('manirt')
```

```
'Atmanirbhar Bh'
```

Atmanirbhar Bharat

```
>>> S.rstrip('nira')
```

```
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.rstrip('Bhar')
```

```
'Atmanirbhar Bharat'
```

Atmanirbhar Bharat

```
>>> S.rstrip('nmtAarhB')
```

```
'Atmanirbhar '
```

Atmanirbhar Bharat

STRING FUNCTION: replace()

CONTENTS

→ Strings Intro:

→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(),
isalpha(), isalnum(),
split(), partition(), strip(),
lstrip(), rstrip(),

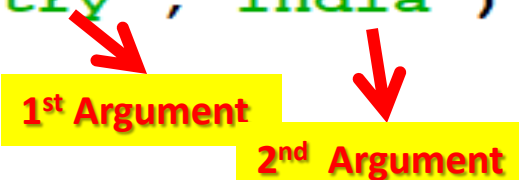
replace(),

startswith(), endswith(),
join(), swapcase()

→ Strings Slicing;

Replace() → Returns a string where a specified value is replaced with a specified value

```
>>> S='I love my country'
>>> S.replace('country', 'India')
'I love my India'
>>> S
'I love my country'
```



The first argument in the replace () function is replaced by the second argument.

```
>>> Str="I love games.I love poetry.I love dance"
>>> Str.replace("I", "We")
'We love games.We love poetry.We love dance'
```

```
>>> Original_str="India is great. India is beautiful.India is united"
>>> Newstr=Original_str.replace("India", "Bharat")
>>> Newstr
'Bharat is great. Bharat is beautiful.Bharat is united'
>>> Original_str
'India is great. India is beautiful.India is united'
```

Strings are immutable. Hence, there is no change in the Original_str. But, when it is assigned to another string, Newstr, the change is visible.

STRING FUNCTION: startswith()

CONTENTS

→ Strings Intro:

→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),

startswith(),

endswith(), join(),
swapcase()

→ Strings Slicing;

startswith() → Returns true if the string starts with the specified value

```
>>> Str='Know Python Bytes'
>>> Str.startswith('K')
True
>>> Str.startswith('Know')
True
>>> Str.startswith('k')
False
>>> Str.startswith('kn')
False
>>> Str.startswith('Kn')
True
>>> Str.startswith('Use')
False
```

STRING FUNCTION: endswith()

CONTENTS

→ Strings Intro:
→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(),

endswith(), join(),
swapcase()

→ Strings Slicing;

endswith() → Returns true if the string ends with the specified value

```
>>> Str='Know Python Bytes'
>>> Str.endswith('s')
True
>>> Str.endswith('Bytes')
True
>>> Str.endswith('tes')
True
>>> Str.endswith('ES')
False
>>> Str.endswith('Python Bytes')
True
>>> Str.endswith('Mask')
False
```

STRING FUNCTION: join()

CONTENTS

→ Strings Intro:
→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(), endswith(),

join(), swapcase()

→ Strings Slicing;

Join() → Joins the elements of an iterable to the end of the string

```
>>> Str="Use Handwash"  
>>> '$'.join(Str)  
'U$s$e$ $H$a$n$d$w$a$s$h'
```

Joining
String
iterable
elements

```
>>> L=['Tech','Talk','By','Payal']  
>>> s='*'.join(L)  
>>> s  
'Tech*Talk*By*Payal'
```

Joining
List
iterable
elements

```
>>> T=("Know","Python","Bytes")  
>>> S='#@'.join(T)  
>>> S  
'Know#@Python#@Bytes'
```

Joining
Tuple
iterable
elements

STRING FUNCTION: swapcase()

CONTENTS

→ Strings Intro:
→ Operations on a String

→ Functions/ methods

len(), capitalize(), title(),
count(), find(),
index(), upper(), lower(),
islower(), isupper(),
isspace(), isdigit(), isalpha(),
isalnum(), split(),
partition(), strip(), lstrip(),
rstrip(), replace(),
startswith(), endswith(),
join(), **swapcase()**
→ Strings Slicing;

Swapcase() →

Swaps cases, lower case becomes upper case and vice versa

```
>>> s="Stay Safe, Stay Happy"  
>>> s.swapcase()  
'sTAY sAFE, sTAY hAPPY'
```

```
>>> str="The sUN, the mOON"  
>>> str.swapcase()  
'tHE Sun, THE Moon'
```

STRING SLICING



STRING slicing

L[start:stop] creates a string slice (substring) out of the given string with elements falling between indexes start and stop, not including value at the stop

L[start:stop:step] creates a string slice (substring) out of the given string with elements falling between indexes start and stop, not including value at the stop index, and skipping step elements in between.

NOTE: In absence of any start, stop, step values, we have Default start index as the beginning, stop index as the last index and step value as +1.

```
>>> Str="KNOW PYTHON BYTES!"  
>>> Str  
'KNOW PYTHON BYTES!'
```

FORWARD INDEX DIRECTION

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
K	N	O	W		P	Y	T	H	O	N		B	Y	T	E	S	!
-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

BACKWARD INDEX DIRECTION

STRING slicing

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
K	N	O	W		P	Y	T	H	O	N		B	Y	T	E	S	!
-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> Str[3:10]
```

```
'W PYTHO'
```

```
>>> Str[5:50]
```

```
'PYTHON BYTES!'
```

```
>>> Str[-4:6]
```

```
''
```

```
>>> Str[-4:25]
```

```
'TES!'
```

```
>>> Str[10:30]
```

```
'N BYTES!'
```

```
>>> Str[-10:12]
```

```
'HON '
```

```
>>> Str[::-1]
```

```
'!SETYB NOHTYP WONK'
```

Start index 3, stop index is (10-1=9)

Start index 5, stop index is 50 which is beyond the last index in forward direction. Hence, it will stop at the last index

Start index -4(backward), stop index is (6-1=5 (forward)). Here, it is not getting any value in the range, hence empty.

Start index -4(which is 'T' as per the backward index,) stop index is 25 which is beyond the last index. Hence, it will stop at the last index.

Start index 10, stop index is 30 which is beyond the last index in forward direction. Hence, it will stop at the last index

Start index -10(which is 'H' as per the backward index,) stop index is 12-1=11 (forward indexing). Hence, it will stop at the (stop index-1), an empty whitespace ' ' is at 11 index no.

The entire sequence of characters get **reversed** as the default start and stop values are taken and step is given as -1

STRING slicing

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
K	N	O	W		P	Y	T	H	O	N		B	Y	T	E	S	!
-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> Str[-3:-12]
''
```

Start index -3(backward), stop index is -12(backward). Here, it is not getting any value in the range as no step value mentioned in reverse. Hence empty.

```
>>> Str[-12:-3]
'YTHON BYT'
```

Start index -12(which is 'Y' as per the backward index,) stop index is -3 (which takes values till index $-3-1 = -4$). Here, -4 is 'T'

```
>>> Str[-3:-12:-1]
'ETYB NOHT'
```

Start index -3(backward), stop index is -12(backward). Here, it is getting value in the range as step value is -1 mentioned in reverse.

```
>>> Str[-20:-5]
'KNOW PYTHON B'
```

Start index -20(which is beyond backward index,) stop index is -5 which takes value till $(-5-1=-6)$. Here, -6 is 'N'.

```
>>> Str[1:13:2]
'NWPTO '
```

Start index 1, stop index is 13 (till $13-1=12$), step value is 2 in forward direction. Hence, it will fetch/take alternate elements with a skip of 1 value in between

```
>>> Str[2:40:5]
'OTB!'
```

```
>>> Str[-1:-12:-3]
'!T H'
```

Try to understand yourself.

```
>>> Str[::3]
'KWYOB'
```

Default start and stop is beginning and end index of the list. Here, step is 3.

DATA TYPE: working with some more **STRING** examples

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
C	o	m	p	u	t	e	r		S	c	i	e	n	c	e	!	!	!
-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> s="Computer Science!!!"
>>> s
'Computer Science!!!'
>>> s[3]
'p'
>>> s[-3]
'!!'
>>> s[-6]
'n'
>>> s[3:]
'puter Science!!!'
>>> s[:12]
'Computer Sci'
>>> s[-8:]
'ience!!!'
>>> s[:-13]
'Comput'
```

```
>>> s[5:16]
'ter Science'
>>> s[7:11]
'r Sc'
>>> s[-4:-8]
''
>>> s[-8:-2]
'ience!'
>>> s[-2:6]
''
>>> s[6:-2]
'er Science!'
>>> print("I Love "+s)
I Love Computer Science!!!
>>> s*2
'Computer Science!!!Computer Science!!!'
```

PROGRAMS

ON

STRING

```
#To check whether a string is palindrome or not
string=input("Enter a string:")
n=len(string)
mid=n//2
rev=-1
for a in range(mid):
    if string[a]==string[rev]:
        a+=1
        rev-=1
    else:
        print(string,"is not a palindrome")
        break
else:
    #Loop else clause, gets executed if the loop terminates normally.
    print(string,"is a palindrome")
```

HINT: TRY OUT IN
ANOTHER WAY:
If `string==string[::-1]`

```
Enter a string:saras
saras is a palindrome
```

OUTPUT