```
                        String Handling
                        ---------------
What is String?
---------------
        a string is a character or sequence of characters or collection words.

in python, the string datatype having 'str' class

in python, the string object is a sequence/ordered,iterable and immutable object.

how to represent the string objects in python?
---------------------------------------------
        we can represent the string object in python in two ways, they are

        1).single-line strings
                a). by using single-single quotes
                        ex: 'siva'

                b). by using single-double quotes
                        ex: "siva"

        2).multi-line strings
                a). by using tripple- single quotes
                        ex: '''siva'''

                b). by using tripple- double quotes
                        ex: """siva"""

ex:
---
        strrep.py
        ---------
x='siva'
y="krishna"
z='''siva
vagdevi technologies
ameerpet
hyderbad
telangana'''
a="""krishna
entri app
hyderabad
telangana"""
print(x)
print('*'*20)
print(y)
print('*'*20)
print(z)
print('*'*20)
print(a)
```

```
        output
        ------
C:\Users\DELL\Desktop>python strrep.py
siva
*******************
krishna
*******************
siva
vagdevi technologies
ameerpet
hyderbad
telangana
*******************
krishna
entri app
hyderabad
telangana
```

note:
----
a quote inside different quote is possible but a quote inside same quote is not possible.

ex2:
--

```
        strrep.py
        ---------
x='hai "siva" krishna'
y="hai 'siva' krishna"
z='''siva
"vagdevi technologies"
'ameerpet'
"""hyderbad"""
telangana'''
a="""krishna
"entri app"
'''hyderabad'''
'telangana'"""
print(x)
print('*'*20)
print(y)
print('*'*20)
print(z)
print('*'*20)
print(a)
```

```
        output
        ------
C:\Users\DELL\Desktop>python strrep.py
```

```
hai "siva" Krishna
******************
hai 'siva' Krishna
******************
siva
"vagdevi technologies"
'ameerpet'
"""hyderbad"""
telangana
******************
krishna
"entri app"
'''hyderabad'''
'telangana'

ex3:
---
        strrep.py
        ---------
x='hai 'siva'krishna'
print(x)

        output
        ------
c:\Users\DELL\Desktop>python strrep.py

syntaxError: invalid syntax.Perhaps you forgot a comma?

ex4:
---
        strrep.py
        ---------
x="hai "siva" krishna"
print(x)

        output
        ------
c:\Users\DELL\Desktop>python strrep.py
SyntaxError: invalid syntax. Perhaps you forgot a comma?

ex5:
---
        strrep.py
        ---------
x="""hai
"""siva"""
krishna"""
print(x)

        output
        ------
c:\Users\DELL\Desktop>python strrep.py
SyntaxError: invalid syntax. Perhaps you forgot a comma?

ex6:
---
        strrep.py
        ---------
x='''hai
'''siva'''
krishna'''
print(x)

        output
        ------
c:\Users\DELL\Desktop>python strrep.py
SyntaxError: invalid syntax. Perhaps you forgot a comma?

ex7:
---
        strrep.py
        ---------
x='''hai 'siva'"krishna" '''
print(x)

        output
        ------
C:\Users\DELL\Desktop>python strrep.py
hai 'siva' "krishna"

ex8:
---
        strrep.py
        ---------
x='hai\nsiva krishna\ngood afternoon'
print(x)

        output
        ------
c:\Users\DELL\Desktop>python strrep.py
hai
siva krishna
good afternoon

ex9:
---
        strrep.py
        ---------
x='hai\tsiva\tkrishna'
print(x)
```

```
        output
        ------
c:\Users\DELL\Desktop>python strrep.py
hai     siva    krishna
```

string object supporting indexing concept, in python indexing can be categorized into two types, they are

        1). positive/forward indexing

        2). negative/backward indexing

```
                        indexing
                           |
                ---------------------------
                |                               |
        forward/positive                Backward/Negative
                |                               |
takes from Left to Right        takes from Right to Left
        (or)                                (or)
takes from Top to Bottom        takes from Bottom to Top
          |                                  |
starts with '0'                 starts with '-1'
```

in string object, each and every character having unique indexing.

we can retreive one by one character from the string object by using indexing concept.

```
>>> x="siva krishna"
>>> x[2]
'v'
>>> x[-10]
'v'
>>> x[9]
'h'
>>> x[-3]
'h'
>>> x[4]
' '
>>> x[-8]
' '
```

in python, the string object supporting slicing concept.

        we can retreive morethan one character from the string object at a time, in that case we are using slicing concept.

        scenario-1

```
        ----------
        if we want to print/retreive first-N characters from the string object in
that case we are using following syntax,

        strobj[ :stop]

                here stop index value is a exclusive value
                        by default starting from '0'
                        by default increment by '1'

>>> x="siva krishna"
>>> x[:4]
'siva'
>>> x[:-9]
'siva'


        scenario-2
        ----------
        if we want to print/retreive last-N characters from the string object, in
that case we are using following syntax,

        strobj[start: ]

                here start index value is a inclusive
                by default increment by '1'

>>> x="siva krishna"
>>> x[-7:]
'krishna'
>>> x[5:]
'krishna'

        scenario-3
        ----------
        if we want to print/retreive some middle characters from the string object,
in that case we are using following syntax,

        strobj[start:stop]

                here start index is a inclusive value
                     stop index is a exclusive value
                     by default Increment by '1'

>>> x=" siva krishna"
>>> x[5:10]
'krish'
>>> x[-7:-2]
'krish'
>>> x[5:-2]
```

```
'krish'
>>> x[-7:10]
'krish'
```

note:
----
        strobj[start:stop:step], if we are not passing any start,stop and step value, in that case our original string exactly printed.

```
>>> x="siva"
>>> x
'siva'
>>> x[::]
'siva'
>>> x{ : : ]
'siva'
>>> x[::1]
'siva'
```

        strobj[start:stop:step], if we are not passing startand stop value but we are passing step value is negative, in that case our string printed from right to left.

```
>>> x="siva"
>> x[::-1]
'avis'
```

ex:
---
wap to print the given string in reverse order?

```
        strrev.py
        ---------
x=input("enter your string: ")
print(" the reverse order of the given '%s' is: '%s' "%(x,x[::-1]))
```

```
        output
        ------
C:\Users\DELL\Desktop>python strrev.py
enter your string: siva
the reverse order of the given 'siva' is:'avis'

C:\Users\DELL\Desktop>python strrev.py
enter your string: vagdevi technologies
the reverse order of the given 'vagdevi technologies' is: 'seigolonhcet ivedgav'
```

ex2:
---
wap to print even indexing charecters from the given string?

```
        evenindexing.py
        ---------------
x=input("enter your string: ")
print(x[::2])

        output
        ------
C:\Users\DELL\Desktop>python evenindexing.py
enter your string: sivakrishna
svkiha

C:\Users\DELL\Desktop>python evenindexing.py
enter your string: vagdevi
vgei
```

ex3:
---
wap to print the odd indexing charecters from the given string object?

```
        oddindexing.py
        -------------
x=input("enter your string: ")
print(x[1::2])

        output
        ------
C:\Users\DELL\Desktop>python oddindexing.py
enter your string: sivakrishna
iarsn

C:\Users\DELL\Desktop>python oddindexing.py
enter your string: vagdevi
adv
```

working with builtin functions:
----
len()
-----
to return the no.of charecters/elements/items from the given iterable object.

```
        len(iterableobj)

>>> x="siva"
>>> y=[5,3,7,2,9,1]
>>> z=236
>>> len(x)
4
>>> len(y)
6
```

```
>>> len(z)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'int' has no len()
```

ord()
----
to return the ASCII value of the given charecter

```
        ord(char)
```

```
>>> ord('a')
97
>>> ord('A')
65
>>> ord('9')
57
>>> ord('@')
64
>>> ord('ab')
TypeError: ord() expected a character, but string of lenght 2 found
```

chr()
----
to return the charecter of the given ASCII value

```
        chr(ASCIIvalue)
```

```
>>> chr(65)
'A'
>>> chr(97)
'a'
>>> chr(64)
'@'
>>> chr(57)
'9'
```

min()
----
to return the minimum charecter/element/item from the given iterable object.

```
                min(iterableobj)
```

```
>>> x="siva"
>>> y=[5,3,7,2,9,1]
>>> z=236
>>> min(x)
'a'
>>> min(y)
1
```

```
>>> min(z)
TypeError: 'int' object is not iterable
```

max()
----
to return the maximum charecter/element/item from the given iterable object.

```
        max(iterableobj)
```

```
>>> x="siva"
>>> y=[5,3,7,2,9,1]
>>> z=236
>>> max(x)
'v'
>>> max(y)
9
>>> max(z)
TypeError: 'int' object is not iterable
```

sorted()
----
to return the charecters/element/item in sorting order from the given iterable object.

by default the sorted() to return the output as ascending order.

by default the sorted() to return the list format as a output.

```
        surted(iterableobj,reverse=False)
```

```
>>> x="siva"
>>> y=[5,3,7,2,9,1]
>>> z=236
>>> sorted(x)
['a', 'i', 's', 'v']
>>> sorted(y)
[1, 2, 3, 5, 7, 9]
>>> sorted(z)
TypeError: 'int' object is not iterable
```

if we want to print the charecters/elements/items in decending order from the given iterable object in that case we are changing the reverse attribute value False to True.

```
        sorted(iterableobj,reverse=True)
```

```
>>> x="siva"
>>> y=[5,3,7,2,9,1]
>>> z=236
>>> sorted(x,reverse=True)
```

```
['v', 's', 'i', 'a']
>>> sorted(y,reverse=True)
[9, 7, 5, 3, 2, 1]
>>> sorted(z,reverse=True)
TypeError: 'int' object is not iterable
```

differences between Function and Method?

                Function
                ----
1). we can define any one logic to perform a particular operation from outside the class, is known as a function.

2).function is not executed automatically, whenever we are calling a function then only function logic is executed.

we can call the function directly by using function name

3). we can use functions on any object

                Method
                ------
1). we can define any one logic to perform a particular operation within/ inside the class, is known as a method.

2).method is not executed automatically, whenever we are calling a method then only method logic is executed.

we can call the method with the help of object

3). we can methods only on the particular objects

note:
----
if we want to display all the available string class methods with syntax and description,by using help()

```
        help(classname)
                help(str)

        help(classname.methodname)
                help(str.upper)

        help(functionname)
                help(len)
```

if we want to display list of all the available properties/methods of any class or any module by using dir()

```
        dir(classname)
```

```
                dir(Str)

        dir(modulename)
                dir(keyword)

working with string class methods
---------------------------------
        in python every built in class having two types of methods, they are

        1). magic/special/dunder(double-underscores) methods

        2). normal methods


magic/special/dunder(double-underscores) methods:
------------------------------------------------
        any method name which contains both prefix and suffix double under-scores,
that type of methods are called magic/special/dunder(double-underscores) methods.

        these methods are executed automatically whenever we are performing that
particular operation.

ex1:
---
>>> x="siva"
>>> y="krishna"
>>> x+y
'sivakrishna'
>>> x._add_(y)
'sivakrishna'

ex2:
---
>>> a="siva"
>>> b="krishna"
>>> c="siva"
>>> a==b
False
>>> a._eq_(b)
False
>>> a==c
True
>>> a._eq_(c)
True
>>> a!=b
True
>>> a._ne_(b)
True
>>> a!=c
False
```

```
>>> a._ne_(c)
False
```

ex3:

```
>>> x="krishna"
>>> len(x)
7
>>> x._len_()
7
```

Normal methods
----
        these methods are not executed automatically, whenever we are calling these methods then only these methods are executed.

capitalize()
----
to return the first-word-first charecter in capital/upper case remaing string in lowercase.

```
        strobj.capitalize()
```

```
>>> x="hai"
>>> y="good afternoon"
>>> z="123rama"
>>> a="hello rama good afternoon"
>>> c="hello RAMA"
>>> x.capitalize()
'hai'
>>> y.capitalize()
'Good afternoon'
>>> z.capitalize()
'123rama'
>>> a.capitalize()
'Hello rama good afternoon'
>>> c.capitalize()
'Hello rama'
```

title()
------
to return the each and every word first alphabet in capital/upper case, the remaining string in lower case.

```
        strobj.title()
```

```
>>> x="hai"
>>> y="good afternoon"
>>> z="123rama"
>>> a="hello rama good afternoon"
>>> c="hello RAMA"
>>> x.title()
'Hai'
>>> y.title()
'Good Afternoon'
>>> z.title()
'123Rama'
>>> a.title()
'Hello Rama Good Afternoon'
>>> c.title()
'Hello Rama'
```

count()
------
to return the no.of occurencess of given charecter/substr in a string object.

```
        strobj.count(char/substr)

>>> x="hai siva krishna hai"
>>> x.count('a')
4
>>> x.count('h')
3
>>> x.count('s')
2
>>> x.count('k')
1
>>> x.count('b')
0
>>> x.count('hai')
2
```

center()
----
to print the string at center

```
        strobj.center(width,fillchar)

                here width is a output string length
                        by default fill char is space-charecter
                        the fillchar must be one character long only

>>> x="siva"
>>> x.center(2)
'siva'
>>> x.center(9)
'   siva   '
>>> x.center(13,'@')
'@@@@@siva@@@@'
```

```
>>> x.center(15, '@#')
TypeError: The fill character must be exactly one character long

ljust()
-------
ljust means left-justification

to print the string at left-side

        strobj.ljust(width,filchar)

>>> x="siva"
>>> x.ljust(0)
'siva'
>>> x.ljust(9)
'siva          '
>>> x.ljust(13, 'a')
'sivaaaaaaaaaaaaa'
>>> x.ljust(15,'ab')
TypeError: The fill character must be exactly one character long

rjust()
------
rjust means right-justification

to print the string at right side

        strobj.rjust(width,fillchar)

>>> x="siva"
>>> x.rjust(4)
'siva'
>>> x.rjust(9)
'       siva'
>>> x.rjust(13,'7')
'77777777777siva'
>>> x.rjust(15, '78')
TypeError: The fill character must be exactly one character long

zfill()
------
to fill with zero's at left-side

        strobj.zfill(width)

>>> x="siva"
>>> x.zfill(2)
'siva'
>>> x.zfill(9)
'00000siva'
```

```
>>> x
'siva'

find()
------
to return the positive index of first-occurencess of given charecter in a string
object from left to right.

        strobj.find(char/substr)

>>> x="siva krishna"
>>> x.find('a')
3
>>> x.find('i')
1

rfind()
------
to return the positive index of first-occurencess of given charecter in a string
object from right to left.

        strobj.rfind(char/substr)

>>> x="siva krishna"
>>> x.rfind('a')
11
>>> x.rfind('i')
7

index()
------
to return the positive index of first- occurencess of given charecter in a string
object from right to left.

        strobj.index(char/substr)

>>> x="siva krishna"
>>> x.index('a')
3
>>> x.index('i')
1

rindex()
------
to return the positive index of first- occurencess of given charecter in a string
object from right to left.

        strobj.rindex(char/substr)

>>> x="siva krishna"
>>> x.rindex('a')
11
>>> x.rindex('i')
7
```

note:
----
if character/substr is find/match in that case there's no difference between find() and index().

```
>>> y="siva"
>>> y.find('v')
2
>>> y.index('v')
2
```

if character/substr is not find/match in that case only we are identify difference between find() and index().

```
>>> y="siva"
>>> y.find('k')
-1
>>> y[-1]
'a'
>>> y.index('k')
ValueError: substring not found
```

ex:

```
>>> x="hai siva krishna hai"
>>> x.index('a')
1
>>> x.index('a',2)
7
>>> x.index('a',8)
15
>>> x.index('a',8,15)
ValueError: substring not found
>>> x.index('a',8,16)
15
```

ex:
--
wap to return all the possible indexes of given charecter.substr in a string object?

```
x=input("enter your string: ")
ch=input("enter your character: ")
for i in range(len(x)):
        if x[i]==ch:
           print(i)

outputs:
-------
enter your string: hai siva krishna hai
enter your charecter: a
1
7
15
18

enter your string: banana
enter your charecter: a
1
3
5

enter your string: hai siva krishna hai
enter your charecter: h
0
13
17
```

ex:

```
>>> x="hai siva krishna hai"
>>> x.index('hai')
0
>>> x.rinder('hai')
17
```

strip()
------
to remove all the possible occurencess of given charecters in a string object at begening or ending or both places.

```
        strobj.strip(char's)
```

by default the strip takes the charecters are white-space charecters.

```
>>> x="siva krishna"
>>> x.strip('s')
'iva krishna'
>>> x
'siva krishna'
>>> x.strip('a')
'siva krishn'
>>> x
```

```
'siva krishna'
>>> x.strip('siva')
' krishn'
>>> x
'siva krishna'

>>> y="mmadam"
>>> y.strip('m')
'ada'
>>> y
'madam'
>>> y.strip('ma')
'd'
>>> y
'madam'
>>> y.strip('dma')
''

>>> z=" siva "
>>> z.strip()
'siva'
>>> z
' siva '

>>> a="\n\t siva\t\n"
>>> a.strip()
'siva'
>>> a
'\n\t siva\t\n'
```

lstrip()
----

to remove all the possible occurencess of given charecters in a string object at begening position only.

```
        strobj.lstrip(char's)
```

```
>>> y= "madam"
>>> y.lstrip('m')
'adam'
>>> y
'madam'
>>> y.lstrip('am')
'dam'
>>> y
'madam'
>>> y.lstrip('dma')
''
>>> y
```

```
'madam'


>>> z=" siva "
>>> z.lstrip()
'siva '
>>> z
' siva '

>>> a=\n\t siva\t\n"
>>> a.lstrip()
'siva\t\n'
>>> a
'\n\t siva\t\n'
```

rstrip()
----
to remove all the possible occurencess of given charecters in a string object at ending position only.

```
        strobj.rstrip(char's)
```

```
>>> y="madam"
>>> y.rstirp('m')
'mada'
>>> y
'madam'
>>> y.rstirp('am')
'mad'
>>> y
'madam'
>>> y.rstirp('dma')
''
>>> y
'madam'


>>> z=" siva "
>>> z.rstrip()
' siva'
>>> z
'siva'

>>> a="\n\t siva\n\t"
>>> a.rstrip()
'\n\t siva'
>>> a
'\n\t siva\n\t'
```

```
split()
------
to split the string into words based on given seperator/delimiter.

by default separator is 'space' charecter

the split() to return the output as list format

        strobj.split(seperator)

>>> x="hai siva krishna good afternoon"
>>> x.split()
['hai', 'siva', 'krishna', 'good', 'afternoon']

>>> y="hai siva krishna good afternoon"
>>> y.split(' ')
['hai', 'siva', 'krishna', 'good', 'afternoon']

>>> z="hai,siva krishna,good afternoon"
>>> y.split(',')
['hai', 'siva krishna', 'good afternoon']

>>> a="hai siva krishna good afternoon"
>>> a.split('i')
['ha', 's', 'va kr', 'shna good afternoon']

ex:
---
wap to return the no.of words in a given string?

x=input("enter your string: ")
sep=input("enter your seperator: ")
print(len(x.split(sep)))

outputs:
-------
enter your string: hai siva krishna good afternoon
enter your separator:
5


enter your string: hai,siva krishna,good afternoon
enter your separator: ,
3

ex2:
---
x= input("enter your string: ")
sep= input(" enter your seperator: ")
print(" the no.of words in a given string based on '%s' seperator\
are: %d words"%(sep,len(x.split(sep))))

output:
------
enter your string: hai siva krishna good afternoon
enter your separator:
the no.of words in a given string based on ' ' seperator are: 5 words

enter your string: hai,siva krishna,good afternoon
enter your separator: ,
the no.of words in a given string based on ', ' seperator are: 3 words


join()
-----
to join the elements from the given iterable object with our string object.

        strobj.join(iterableobj)

>>> x="siva"
>>> y=" "
>>> y.join(x)
's i v a'
>>> a=["D", "Siva"]
>>> b="."
>>> b.join(a)
'D.siva'

lower()
------
to convert a string into lower -case.

        strobj.lower()

>>> x="siva"
>>> y="RAMA"
>>> z="KrIsHnA"
>>> a="siva@123"
>>> b="RAMA@123"
>>> x.lower()
'siva'
>>> y.lower()
'rama'
>>> z.lower()
'krishna'
>>> a.lower()
'siva@123'
>>> b.lower()
'rama@123'
```

```
upper()
------
to convert a string into upper-case.

        strobj.upper()

>>> x="siva"
>>> y="RAMA"
>>> z="KrIsHnA"
>>> a="siva@123"
>>> b="RAMA@123"
>>> x.upper()
'SIVA'
>>> y.upper()
'RAMA'
>>> z.upper()
'KRISHNA'
>>> a.upper()
'SIVA@123'
>>> b.upper()
'RAMA@123'

swapcase()
---------
to swap(exchange) the cases i.e.. to convert lower to upper and vice-versa.

        strobj.swapcase()

>>> x="siva"
>>> y="RAMA"
>>> z="KrIsHnA"
>>> a="siva@123"
>>> b="RAMA@123"
>>> x.swapcase()
'SIVA'
>>> y.swapcase()
'rama'
>>> z.swapcase()
'kRiShNa'
>>> a.swapcase()
'SIVA@123'
>>> b.swapcase()
'rama@123'

replace()
--------
to replace the existed char/substr with new char/substr.

        strobj.replace(old char/ substr, new char/substr)
```

```
>>> x="siva krishna"
>>> x.replace('i','u')
'suva krushna'
>>> x
'siva krishna'
>>> x.replace('siva','rama')
'rama krishna'
>>> x
'siva krishna'
>>> x.replace('krishna','ram')
'siva ram'
>>> x
'siva krishna'
```

startswith()
----

to check whether the string object starts with given char/substr or not.

```
        strobj.startswith(char/substr)
```

```
>>> x="siva krishna"
>>> x.startswith('s')
True
>>> x.startswith('a')
False
>>> x.startswith('siv')
True
>>> x.startswith('sia')
False
```

endswith()
----

to check whether the string object ends with given char/substr or not.

```
        strobj.endswith(char/substr)
```

```
>>> x="siva krishna"
>>> x.endswith('a')
True
>>> x.endswith('s')
False
>>> x.endswith('hna')
True
>>> x.endswith('sna')
False
```

islower()
----

to check whether our string object contains only lowercase alphabets or not?

```
        strobj.islower()


>>> x="siva"
>>> y="RAMA"
>>> z="KrIsHnA"
>>> x.islower()
True
>>> y.islower()
False
>>> z.islower()
False
```

isupper()
----
to check whether our string object contains only uppercase alphabets or not?

```
        strobj.isupper()

>>> x="siva"
>>> y="RAMA"
>>> z="KrIsHnA"
>>> x.isupper()
False
>>> y.isupper()
True
>>> z.isupper()
False
```

isspace()
----
to check whether given string object contains only white-spaces or not.

```
        strobj.isspace()

>>> x=" "
>>> y=""
>>> z=" siva "
>>> a="\t"
>>> b="\n"
>>> c=" "
>>> x.isspace()
True
>>> y.isspace()
False
>>> z.isspace()
False
>>> a.isspace()
True
>>> b.isspace()
```

```
True
>>> c.isspace()
True
```

istitle()
----
to check whether given string object is follow the title case or not?

```
        strobj.istitile()
```

```
>>> x="Hai"
>>> y="Good afternoon"
>>> z="Rama Krishna"
>>> a="123Rama"
>>> b="Hello RAMA"
>>> x.istitle()
True
>>> y.istitle()
False
>>> z.istitle()
True
>>> a.istitle()
True
>>> b.istitle()
False
```

isalnum()
----
to check whether our string object contains only alphabets or digits or both combination of alphabets and digits or not.

```
        strobj.isalnum()
```

```
>>> x="siva"
>>> y="RAMA"
>>> z="KrIsHnA"
>>> a="siva123"
>>> b="siva@123"
>>> c="123"
>>> x.isalnum()
True
>>> y.isalnum()
True
>>> z.isalnum()
True
>>> a.isalnum()
True
>>> b.isalnum()
False
>>> c.isalnum()
```

```
True

isalpha()
---------
to check whether given string object contains only alphabets or not

        strobj.isalpha()

>>> x="siva"
>>> y="RAMA"
>>> z="KrIsHnA"
>>> a="siva123"
>>> b="siva@123"
>>> x.isalpha()
True
>>> y.isalpha()
True
>>> z.isalpha()
True
>>> a.isalpha()
False
>>> b.isalpha()
False

isdigit()
---------
to check whether given string object contains only digits or not

        strobj.isdigit()

>>> a="siva123"
>>> b="siva@123"
>>> c="123"
>>> a.isdigit()
False
>>> b.isdigit()
False
>>> c.isdigit()
True

casefold()
--------
if we want to perform the caseless comparisions in that case we are using
casefold()

the casefold() internally to convert the string into lowercase and compare the
strings.

        strobj.casefold()
```

```
>>> x="rama"
>>> y="rama"
>>> z="Rama"
>>> x==y
True
>>> x==z
False
>>> x==z.casefold()
True

format()
-------
to print the string into our required format.

        strobj.format()

>>> "{}{}".format("hai","siva","krishna")
'haisiva'

>>> "{1} {2}".format("hai","siva","krishna")
'siva krishna'

>>> "{1} {2} {0}".format("hai","siva","krishna")
'siva krishna hai'

>>> "{} {} {}".format("hai","siva","krishna")
'hai siva krishna'

>>> "{} {} {} {}".format("hai","siva","krishna")
IndexError: Replacement index 3 out of range for positional args tuple

>>> "{} {2} {1}" .format("hai","siva","krishna")
ValueError: cannot switch from automatic field numbering to manual field
specification

>>> "{0} {2} {}" .format("hai","siva","krishna")
ValueError: cannot switch from manual field specification to automatic field
numbering

ex:
---
name=input("enterb your name: ")
age=int(input("enter your age: "))
print("my name is", name,"and my age is",age)
print("my name is %s and my age is %d"%(name,age))
print("my name is {} and my age is {}".format(name,age))
print(f"my name is {name} and my age is {age}")

output:
------
```

```
enterb your name: siva
enter your age: 29
my name is siva and my age is 29
my name is siva and my age is 29
my name is siva and my age is 29
my name is siva and my age is 29
```

note:
----
from python3.9 versions onwards, in string class to introduce two new methods, they are

```
        removeprefix()
        removesuffix()
```

```
>>> x="aabbaa"
>>> a.lstrip('a')
'siva123'
>>> x="aabbaa"
>>> x.lstrip('a')
'bbaa'
>>> x
'aabbaa'
>>> x.removeprefix('a')
'abbaa'
>>> x
'aabbaa'
>>> x.rstrip('a')
'aabb'
>>> x
'aabbaa'
>>> x.removesuffix('a')
'aabba'
>>> x
'aabbaa'
```