String Manipulation

- Predefined class for string is <class 'str'>
- String is an immutable object
- String is collection of A|N values but in the string object data will organized in the form an array, when we read data from string object using indexing or slicing
- String can represented by using 'ś' or "s" or '''shashi''' or """sssit"""

#string literals in python are 2 types

#single line literals and multi-line literals

#single line literals are represented by using ' ' or " "

```
s1='welcome to sssit'

print("Type is : ",type(s1))

print("Data is : ",s1)


s2="have a nice day"

print("type is : ",type(s2))
print("Data is : ",s2)
```

#Multi-line Literals are represented by using ‘’’…’’’ and “”””….”””””

```
s3='''
Have
A
Good
Day '''
print("Type is : ",type(s3))
print("Data is : ",s3)

s4="""
Have
A
```

```
Great

Session """

print("Type is : ",type(s4))

print("Data is : ",s4)
```

Example for Single line literal

```
s='Have \

A \

Nice \

Day '

print("Data is : ",s)
```

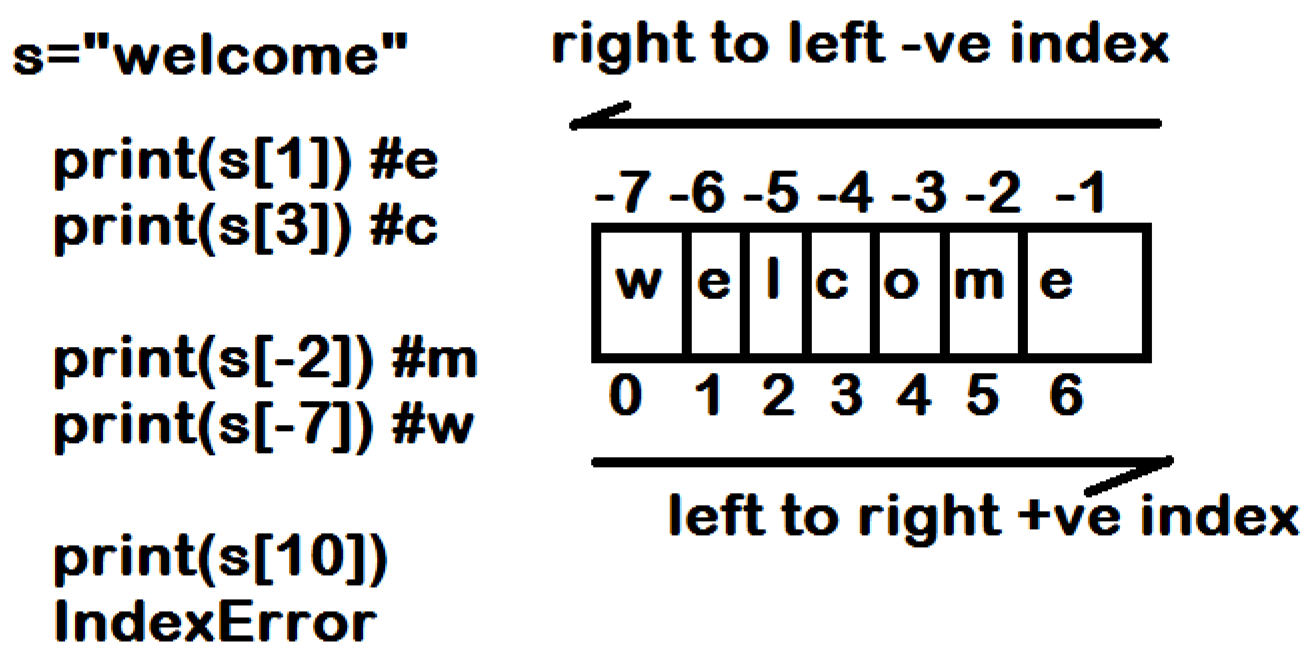
Example:

```
s1="Have a \"nice\" Day "

s2='Have a "nice" Day'

#Exp.output : Have a "nice" Day

print("Data is : ",s1)

print("Data is : ",s2)
```

```
s3='Have a \'Good\' Day'

s4="Have a 'Good' Day"

print("Data is : ",s3)

print("Data is : ",s4)
```

Example Using RAWString :

```
s="\nHello \n\n My \t\t Dear \n\n Miss U.."

print("Data is : ",s)

s1=r"\nHello \n\n My \t\t Dear \n\n Miss U.."

print("Data is : ",s1)
```



```
s="welcome"

print(s[1]) #e
print(s[3]) #c

print(s[-2]) #m
print(s[-7]) #w

print(s[10])
IndexError
```

s="WELCOME"

#emoc

print(s[-1:-5:-1])

#even position

print(s[0:7:2])

#odd position

print(s[1:7:2])

#Reversing the String

s="welcome"

print(s[-1:-8:-1])

print(s[-1: :-1])

print(s[::-1])

<u>Checking Relationship between String:</u>

In C-Language to compare strings we used to work with strcmp( ), but in python we have to use relational operators in order to check the relationship [>,>=,<,<=,==,!=]

- **In python string comparison is possible by using its UNICODE character in simple words English dictionary order**

```
print(" 'A'>'a'   ? : ",'A'>'a') # 65>97 False
print(" 'A'=='a' ? : ",'A'=='a') #65==97 False
print(" 'A'<'a' ? : ",'A'<'a') #65<97 True
```

Example:

```
s1=input("Enter a string ")
s2=input("Enter a string ")
if s1>s2:
    print("Biggest is : ",s1)
elif s1<s2:
    print("Biggest is : ",s2)
else:
    print("Both Are same ")
```

Example2: #String palindrome or not

```
n=input("Enter any string ") #n=sai

r=n[::-1]

print("Reverse : ",r)

if r==n:
    print("Polin")
else:
    print("Not Polin")
```

len( )
Syn: len(iterable)

- **It is used return total no.of.objects are existed in the iterable [str | list | tuple | set | dict ..] Object**
    - **Eg: s="welcome" → <class 'str'> iterable**
    - **len(s) → 7**

Example :

```
#Syn: len(iterable) -> int
s1="welcome"
n=len(s1)
print("Length is : ",n)
print("Length is :",len(s1))
```

Checking Sub Strings :
In order to ensure the existence of sub string then we have to member ship operators [in | not in]

```
s="welcome"
print(" 'e' is existed in 'welcome' ? : ",'e' in s)
```

Example 2:

```
main=input("Enter Main String ") #hello my dear
sub=input("Enter sub String ")  #my

if sub in main:
   print(sub ,"is Existed in ",main)
else:
   print(sub,"is not existed in ",main)

#Concatenation and Repetition
#str+str -> str [concatenation]
#str+xxx --> Error
#xxx+str --> Error
a="Sai"
b="Baba"
c=a+b
print("Concatenation : ",c)

#Ex2:
i="Moon"   #<class 'str'>
j=123      #<class 'int'>
#k=i+j   Error
k=i+str(j)   #Moon123
print("Result is : ",k)
```

Repetition [*]:

```
#str*int
print("A "*5)
#int*str
print(5*"A ")
#str*int*int
print("A"*3*3)
#int*str*int
print(3*"A"*3)
```

Working with Methods:

id( ) | type( ) | print( ) | exit( ) | len( ) -> Functions

Complex

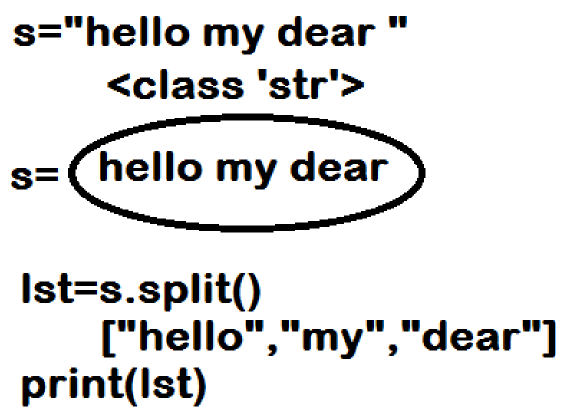
Eg: c=(10+20j) #<class 'complex'>

print(c.real) and print(c.imag) #here in this context real and imag are not methods rather they are attributes

<u>Splitting and Joining :</u>

syn: S.split([chars]) -> list

- **It is used to split the string object at the specified delimiter ,but default delimiter is space " "**



```
s="hello my dear "
      <class 'str'>
s= hello my dear

lst=s.split()
     ["hello","my","dear"]
print(lst)
```

Ex2:

```
s="hello my.dear"
lst=s.split(".")
print("Result is ",lst)
```

joining:

Syn:S.join(iterable) -> str

- **Used to convert list object to string object**

```
lst=["ramesh","sudha","roja","manasa"]
s="-".join(lst)
print("Result is : ",s)
```

Ex2:

```
s="w e l c o m e"
lst=s.split()
print("Type is : ",type(lst))
print("Result is : ",lst)
```

```
s2="".join(lst)
print("Type is " ,type(s2))
print("Result is :",s2)
```

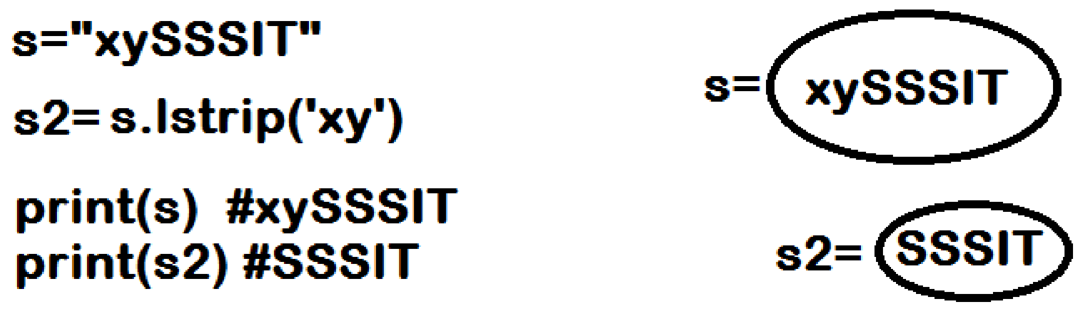
Tailoring String :

lstrip( ) | rstrip( ) | strip( )

lstrip( ):

Syn: S.lstrip([chars]) -> str

- It will return a string object by strip [Erase] the specified chars at left of string object[content] lly rstrip( ) will erase right hand side
- If we fail to specify the chars to trim [Erase] then it will erase empty spaces if exists



```
s="xySSSIT"
s2=s.lstrip('xy')
print(s)  #xySSSIT
print(s2) #SSSIT
```

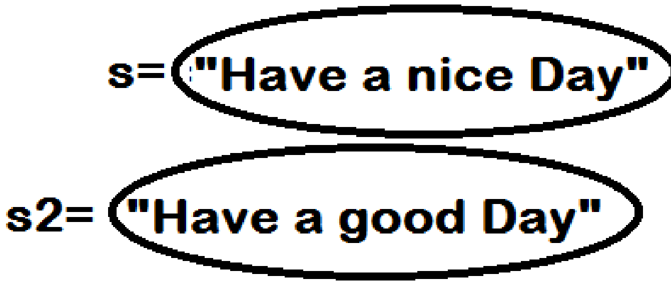
```
#S.strip([chars]) -> str
x="    SSSIT      "
print("Data is (%s) " %x)
y=x.strip()
print("Result is (%s) " %y)
```

Ex :

```
first_name=input("Enter First Name : ")
last_name=input("Enter Last Name : ")
#S.strip([chars]) -> str

fn=first_name.strip()
ln=last_name.strip()

full_name=fn+" "+ln
print("FullName is : ",full_name)

#Replace()
Syn: S.replace(str1,str2) -> str
```

- **It will return new string object by replacing old string with new string**

```
 S.replace(str old,str new) -> str

  s="Have a nice Day"
  s2=s.replace("nice","good")

 print("Data ",s)
   #Have a nice Day

 print("Result ",s2)
   #Have a good Day
```



-

Ex :

```
sal=input("Enter salary per month ")
sal=sal.replace(",","")
s=int(sal)
asal=s*12
print("Ann.Salary is : ",asal)
```

index( ) | rindex( )

Syn: S.index(sub[,start,end]]) -> int

- **Index will return index position of the first occurrence of the specified sub,if the specified sub string not existed then will return “ValueError”**

```
s="welcome"
pos=s.index('e')
print(" 'e' Found @ : ",pos)
pos=s.index("e",3,7)
print(" 'e' Found @ : ",pos)
```

Syn: rindex(sub[,start[,end]]) -> int

- **It will also return +ve index position only of the specified substring but searching process from right side**

find( )
Syn: S.find(sub[,start,end]]) -> int
rfind( )
Syn: S.rfine(sub[,start,end]]) -> int

Index( ) and find() will work same but when the specified sub string is not existed then index( ) will return "Value Error", where as find() will return "-1"

```
#S.find(sub[,start,[end] ]) -> int |-1

s="welcome"
pos=s.find("e")
print(" 'e' Found @ : ",pos)

pos=s.find("e",3,7)
print(" 'e' found @ : ",pos)
```

```
    pos=s.find("E")
    print("Result is : ",pos)
```

count()

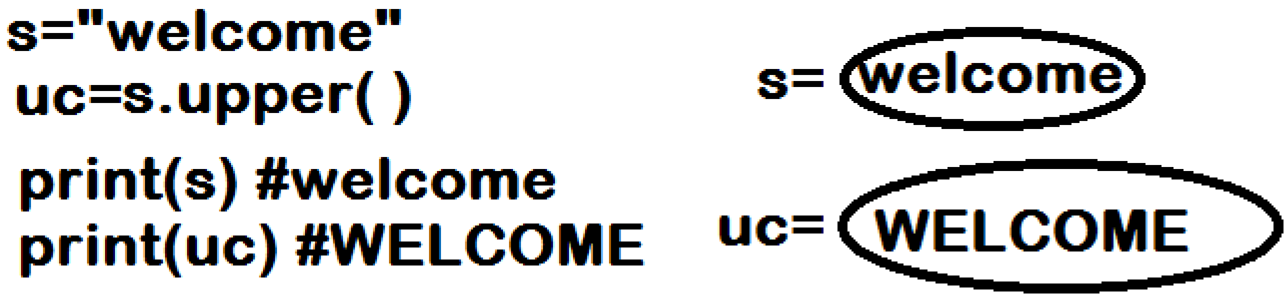
- **It will always used to return no.of. occurrences of the specified substring**

```
#S.count(sub[,start,end]]) -> int  | 0
s="welcome"
cnt=s.count("e")
print(" e found for ",cnt)

cnt=s.count("e",3,len(s))
print("e found for ",cnt)

cnt=s.count("E")
print("E found for ",cnt)
```

<u>Formatting Strings Using formatting methods</u>

Upper( ) : it will return string object by converting string into upper case letters

    Syn: S.upper() -> str



```
s="welcome"
uc=s.upper( )
print(s) #welcome
print(uc) #WELCOME
```

lower( ) : it will return string object by converting them into lowercase letters

Syn: S.lower( ) -> str

S="WELCome"
lc=s.lower()
print("Result is : ",lc)

Capitalize( ) : it will return string object by converting starting letter of statement in upper case letter

Title() : it will return string object by converting entire string into title case format(Starting letter of each word in upper case letter)

Syn: S.title()->str

Swapcase( ) : it will return string object by converting capital to small and small to capital in single operation

Syn: S.swapcase()-> str

Example:

db="SSSIT" # data is coming from DB

```
user=input("Enter Username : ") #SSsit___
udws=user.strip()  #udws=SSsit
uc=udws.upper()  #uc=SSSIT

udb=db.upper()
if uc==udb:
   print("Valid User ")
else:
   print("Invalid User...!!!")
```

Example 2:

```
db="SSSIT" # data is coming from DB
user=input("Enter Username : ") #SSsit___
   #1.user.strip()  ["SSsit__".strip() --> "SSsit"]
   #2."SSsit".upper()   ["SSSIT"]
if user.strip().upper()==db.upper():
   print("Valid User ")
else:
   print("Invalid User...!!!")
```

Note: In Python For every character | digit | symbol or any literal will have its Unicode char

A TO Z → 65 to 90 | a to z → 97 to 122 |
0 to 9 → 48 to 57

Ord() : it will return Unicode char for given char
Eg: ord('A') → 65

ord('a') → 97

chr(): it will return "character" for given Unicode

Eg: chr(65) → A
chr(97) → a

Casefold( ) : it will return string object by ignoring the case [they from any language]

```
>>> chr(223)
'ß' [Germen letter which is equal to "ss"]
  print('ß'=="ss")   #False
  print('ß'.casefold()=="ss".casefold())  #True
>>> f="shashi"+chr(223)
>>> print(f)
shashiß
>>> s="shashiss"
>>> f.lower()
'shashiß'
>>> s.lower()
'shashiss'
>>> f==s
False
>>> f.casefold()==s.casefold()  #True
>>> chr(223).casefold()  #'ss
```

Startswith() : it will returns True if main string object is starts with the given “substring”

```
 #  S.startswith(substring) -> bool
```

Endswith() : it will returns True if main String object is ends with the given “substring”

```
# S.endswith(substring) -> bool
```

```
main=input("Enter main string ") #sai is good
sub=input("Enter sub string ") #sai
if main.startswith(sub):
   print(main," starts with ",sub)
else:
   print(main," not starts with ",sub)
```

<u>Testing methods in String:</u>

isdigit()-> bool : it will returns True if content of string object is digits

Syn: S.isdigit() -> bool

```
s="123"
>>> s.isdigit()
True
```

isupper() : return True if the content of string object is uppercase letters only

```
s="SHASHI"
```

```
>>> s.isupper()
True
```

islower() | isspace()|isprintable() |isalpha()|isalnum()

Ex:

```
#counting No.of.Digits | small | capital

import time
s="1ab2CD3"
nd=ns=nc=0

if s.isalnum():
   for i in s:
      if i.isdigit():
         nd=nd+1
      elif i.islower():
         ns=ns+1
      elif i.isupper():
         nc=nc+1
else:
   print("Sorry it is not an A|N")

time.sleep(1)
print("Data is : ",s)
print("No.of.Digits : ",nd)
print("No.of.Capital : ",nc)
print("No.of.Small : ",ns)
```

Ex 2:

#counting No.of.Digits | small | capital
# Spaces | Sym | vowels | total no.lts

```
import time

data=input("Enter u r data ")  #hello
vowels=['a','A','e','E','i','I','o','O','u','U']
ns=nc=nd=nsy=nsp=nv=0

for i in data:
   if i in vowels:
      nv=nv+1

   if ord(i)>=97 and ord(i)<=122:
      ns=ns+1
   elif ord(i)>=65 and ord(i)<=90:
      nc=nc+1
   elif ord(i)>=48 and ord(i)<=57:
      nd=nd+1
   elif ord(i)==32:
      nsp=nsp+1
   else:
      nsy=nsy+1

print("-"*50)
print("No.of.Digits : ",nd)
print("No.of.Cap : ",nc)
print("No.of.Sm : ",ns)
print("No.of.Spaces : ",nsp)
print("No.of.Sym : ",nsy)
print("No.of.Vowel : ",nv)
tnl=len(data)
print("Total No.of.Char : ",tnl)
print("-"*50)
```