Strings:

- The most commonly used object in any project and in any programming language is String only. Hence we should aware complete information about String data type.
- Any sequence of characters within either single quotes or double quotes is considered as a String.

Accessing characters of the string:

- We can access characters of a string by using the following ways.

    1. By using index
    2. By using slice operator

1. Accessing characters by using index:

- Python supports both +ve and -ve index.
- +ve index means left to right(Forward direction)
- -ve index means right to left(Backward direction)

Example:

```
s="ABCDEFG"
print(s[3])
print(s[-2])
```

Output:

```
D
F
```

Example:

```
s="ABCDEFG"
print(s[300])
```

Output:

```
Traceback (most recent call last):
  File "test.py", line 2, in <module>
    print(s[300])
IndexError: string index out of range
```

Note: If we are trying to access characters of a string with out of range index then we will get error saying : IndexError

Example:

```
s="ABCDEFGH"
for x in s:
    print(x)
```

Output:

```
A
B
C
D
E
F
G
H
```

Example:

```
s="ABCDEFGH"
i=0
for x in s:
    print(x,"+ve index is ", i)
    i=i+1
```

Output:

```
A +ve index is  0
B +ve index is  1
C +ve index is  2
D +ve index is  3
E +ve index is  4
F +ve index is  5
G +ve index is  6
H +ve index is  7
```

Example:

```
s="ABCDEFGH"
l=len(s)
print(l)
l1=len(s) - 1
print(l1)
print(s[len(s)-1])
```

Output:

```
8
7
H
```

Example:

```
s="ABCDEFGH"
i=0
for x in s:
   print(x,"-ve index", i-len(s))  #2 - 8
   i=i+1
```

Output:

```
A -ve index -8
B -ve index -7
C -ve index -6
D -ve index -5
E -ve index -4
F -ve index -3
G -ve index -2
H -ve index -1
```

Example:

```
s="ABCDEFGH"
i=0
for x in s:
   print(x,"+ve index",i,"-ve index", i-len(s))   #2 - 8
   i=i+1
```

Output:

```
A +ve index 0 -ve index -8
B +ve index 1 -ve index -7
C +ve index 2 -ve index -6
D +ve index 3 -ve index -5
E +ve index 4 -ve index -4
F +ve index 5 -ve index -3
G +ve index 6 -ve index -2
H +ve index 7 -ve index -1
```

2. Accessing characters by using slice operator:

Syntax: s[beginindex:endindex:step]

beginindex: From where we have to consider slice(substring)

endindex: We have to terminate the slice(substring) at endindex-1

step: incremented value

Note: If we are not specifying begin index then it will consider from beginning of the string. If we are not specifying end index then it will consider up to end of the string, the default value for step is 1:

Example:

```
s="ABCDEFGH"
b=len(s)-1
print(b)
print(s[b])
```

Output:

```
7
H
```

Behaviour of slice operator:

- s[bEgin:end:step]
- step value can be either +ve or –ve
- if +ve then it should be forward direction(left to right) and we have to consider begin to end-1 if -ve then it should be backward direction(right to left) and we have to consider begin to end+1

Note:

In the backward direction if end value is -1 then result is always empty. In the forward direction if end value is 0 then result is always empty.

In forward direction:

- default value for begin: 0
- default value for end: length of string default value for step: +1

In backward direction:

- default value for begin: -1
- default value for end: -(length of string+1)

Note: Either forward or backward direction, we can take both +ve and -ve values for begin and end index.

Example:

```
s="abcdefghij"
print(s)
print(len(s))
```

Output:

```
abcdefghij
10
```

Example:

```
s="abcdefghij"
print(s[1:6:2])
```

Output:

```
bdf
```

Example:

```
s="abcdefghij"
print(s[::1])
```

Output:

```
abcdefghij
```

Example:

```
s="abcdefghij"
print(s[::-1])
```

Output:

```
jihgfedcba
```

Example:

```
s="abcdefghij"
print(s[3:7:-1])
```

Output:

```
        Empty
```

Example:

```
s="abcdefghij"
print(s[7:4:-1])
```

Output:

```
hgf
```

Example:

```
s="abcdefghij"
print(s[0:1000:1])
```

Output:

```
abcdefghij
```

Mathematical Operators for String:

- We can apply the following mathematical operators for Strings.

1. + operator for concatenation

2. * operator for repetition

Example:

```
print("Hello"+"World")

print("Hello"*4)
```

Output:

```
HelloWorld

HelloHelloHelloHello
```

Note:

1. To use + operator for Strings, compulsory both arguments should be str type

2. To use * operator for Strings, compulsory one argument should be str and other argument should be int

len() in-built function:

- We can use len() function to find the number of characters present in the string.

Example:

```
s='ABCDEFGH'

print(len(s))

output:

8
```

Example:

```
a="Hello"

b="How are you"

print(a+b)

output:

HelloHow are you
```

Example:

```
a="Hello"

b=4

print(a*b)
```

Output:

```
HelloHelloHelloHello
```

Checking Membership:

- We can check whether the character or string is the member of another string or not by using in and not in operators

Example:

```
a="Hello"

print("e" in a)

print("z" in a)
```

Output:

```
True

False
```

Example:

```
a="Hello"

print("e" not in a)

print("z" not in a)
```

Output:

```
False

True
```

Example:

```
s=input("Enter main string:")

subs=input("Enter sub string:")

if subs in s:

        print(subs,"is found in main string")

else:

        print(subs,"is not found in main string")
```

Output:

```
Enter main string:Hello World
Enter sub string:World
World is found in main string
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter main string:Hello World
Enter sub string:Hai
Hai is not found in main string
```

Removing spaces from the string:

We can use the following 3 methods

1. rstrip()===>To remove spaces at right hand side
2. lstrip()===>To remove spaces at left hand side
3. strip() ==>To remove spaces both sidesstrip()

Example:

```
city=input("Enter your city name:")
if city=="Hyd":
   print("Services are available")
else:
   print("No serbices")
```

output:

```
Enter your city name:Hyd
Services are available

C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter your city name:  Hyd
No serbices

C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter your city name:Hyd
No serbices
```

Example:

```
city=input("Enter your city name:")
if city.lstrip()=="Hyd":
   print("Services are available")
else:
   print("No serbices")
```

Output:

```
Enter your city name:Hyd
No serbices
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter your city name:  Hyd
Services are available
```

Example:

```
city=input("Enter your city name:")
if city.rstrip()=="Hyd":
   print("Services are available")
else:
   print("No serbices")
```

output:

```
Enter your city name:   Hyd
No services
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter your city name:Hyd
Services are available
```

Example:

```
city=input("Enter your city name:")
if city.strip()=="Hyd":
   print("Services are available")
else:
   print("No serbices")
```

Output:

```
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py

Enter your city name:Hyd

Services are available

C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py

Enter your city name:  Hyd

Services are available

C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py

Enter your city name:Hyd

Services are available

C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py

Enter your city name:    Hyd

Services are available
```

Counting substring in the given String:

We can find the number of occurrences of substring present in the given string by using count() method.

1. s.count(substring) ==> It will search through out the string
2. s.count(substring, bEgin, end) ===> It will search from begin index to end-1 index

To count characters or substrings of given string

Example:

```
s="ABCDEAB"

print(s.count("A"))

print(s.count("B"))

print(s.count("C"))

print(s.count("D"))
```

Output:

```
2

2

1

1
```

Example:

```
s="ABCDEABCDABABCD"
print(s.count("ABCD"))
print(s.count("AB"))
print(s.count("CD"))
```

output:

```
3
4
3
```

Example:

```
s="abcabcabcabcadda"
print(s.count('a'))
print(s.count('ab'))
print(s.count('a',3,7))
```

Output:

```
6
4
2
```

Replacing a string with another string:

- s.replace(oldstring,newstring) inside s, every occurrence of old string will be replaced with new string.

Example:

```
s="ABCDEABCDABABCD"
s2=s.replace("A","Z")
print(s2)
```

Output:

```
ZBCDEZBCDZBZBCD
```

Example:

```
s="ABCDEABCDABABCD"

s2=s.replace("ABCD","1234")

print(s2)
```

Output:

```
1234E1234AB1234
```

Splitting of Strings:

- We can split the given string according to specified separator by using split() method.
- l=s.split(seperator)
- The default separator is space. The return type of split() method is List

Example:

```
s="Hello How Are you"

s1=s.split()

print(s1)

print(type(s1))
```

output:

```
['Hello', 'How', 'Are', 'you']

<class 'list'>
```

Example:

```
s="Hello How Are you"

s1=s.split()

for x in s1:

   print(x)
```

Output:

```
Hello

How

Are

you
```

Example:

```
s="Hello How Are you"
s1=s.split()
for x in s1:
   for  y   in x:
       print(y)
```

Output:

```
H
e
l
l
o
H
o
w
A
r
e
y
o
u
```

Example:

```
d='05-04-2022'
l=d.split('-')
print(l)
print(type(l))
output:
['05', '04', '2022']
<class 'list'>
```

Joining of Strings:

- We can join a group of strings (list or tuple) wrt the given separator.
- s=seperator.join(group of strings)

Example:

```
l=["Sun","Mon","Tue","Wed"]
s='-'.join(l)
print(s)
print(type(s))
output:
Sun-Mon-Tue-Wed
<class 'str'>
```

Example:

```
l=["Sun","Mon","Tue","Wed"]
s=''.join(l)
print(s)
print(type(s))
output:
SunMonTueWed
<class 'str'>
```

Changing case of a String:

- We can change case of a string by using the following 4 methods.

  1.upper()===>To convert all characters to upper case

  2.lower() ===>To convert all characters to lower case

  3.swapcase()===>converts all lower case characters to upper case and all upper case characters to lower case

  4.title() ===>To convert all character to title case. i.e first character in every word should be upper case and all remaining characters should be in lower case.

  5.capitalize() ==>Only first character will be converted to upper case and all remaining characters can be converted to lower case

Example:

```
s="hello world"
print(s.upper())
s1="HELLO WORLD"
print(s1.lower())
s2="HeLlO wOrLd"
print(s2.swapcase())
s3="python programing is very easy"
print(s3.title())
print(s3.capitalize())
```

Output:

```
HELLO WORLD
hello world
hElLo WoRlD
Python Programing Is Very Easy
Python programing is very easy
```

Example:

```
username=input("enter user name:")
pwd=input("Enter pass word:")
if username=="jagan" and pwd=="mohan":
   print("Welcome",username)
else:
   print("Invalid user")
```

Output:

```
enter user name:xyz
Enter pass word:abc
Invalid user
```

```
C:\Users\jagan\OneDrive\Desktop\pythonbatch2>python test.py
enter user name:jagan
Enter pass word:xyz
Invalid user

C:\Users\jagan\OneDrive\Desktop\pythonbatch2>python test.py
enter user name:Jagan
Enter pass word:mohan
Invalid user
```

Example:

```
username=input("enter user name:")
pwd=input("Enter pass word:")
if username.lower()=="jagan" and pwd=="mohan":
   print("Welcome",username)
else:
   print("Invalid user")
```

output:

```
C:\Users\jagan\OneDrive\Desktop\pythonbatch2>python test.py
enter user name:JAGAN
Enter pass word:mohan
Welcome JAGAN
C:\Users\jagan\OneDrive\Desktop\pythonbatch2>python test.py
enter user name:Jagan
Enter pass word:mohan
Welcome Jagan
C:\Users\jagan\OneDrive\Desktop\pythonbatch2>python test.py
enter user name:JaGAn
Enter pass word:mohan
Welcome JaGAn
```

Example:

```
username=input("enter user name:")
pwd=input("Enter pass word:")
if username.lower()=="jagan" and pwd=="mohan":
   print("Welcome",username.lower())
else:
   print("Invalid user")
```

output:

```
enter user name:JAGAN
Enter pass word:mohan
Welcome jagan
```

Checking starting and ending part of the string:

- Python contains the following methods for this purpose

1. s.startswith(substring)
2. s.endswith(substring)

Example:

```
s="Python is Easy"
print(s.startswith("P"))
print(s.startswith("Z"))
print(s.startswith("Python"))
```

Output:

```
True
False
True
```

Example:

```
s="Python is Easy"
print(s.endswith("y"))
print(s.endswith("x"))
print(s.endswith("asy"))
```

output:

```
True
False
True
```

To check type of characters present in a string:

- Python contains the following methods for this purpose.

1) isalnum(): Returns True if all characters are alphanumeric(a to z , A to Z ,0 to9)
2) isalpha(): Returns True if all characters are only alphabet symbols(a to z,A to Z)
3) isdigit(): Returns True if all characters are digits only(0 to 9)
4) islower(): Returns True if all characters are lower case alphabet symbols
5) isupper(): Returns True if all characters are upper case aplhabet symbols
6) istitle(): Returns True if string is in title case
7) isspace(): Returns True if string contains only spaces1.isalnum():

1.isalnum():

- If given string contains alphanumeric data (a to z,A to Z, 0 to 9) isalnum() return True

otherwise it returns False.

Example:

```
s="12345"
print(s.isalnum())
s2="123abc"
print(s2.isalnum())
s3="123@#abc"
print(s3.isalnum())
```

Output:

```
True
True
False
```

2.isalpha():

- If given string contains only alpha data (a to z,A to Z) isalpha() returns True

Otherwise it return False

Example:

```
s="ABCD"
print(s.isalpha())
s1="ABC123"
print(s1.isalpha())
output:
True
False
```

3.islower():

- It is checking given data is lower case or not

Example:

```
s="ABCD"
print(s.islower())
s1="abcd"
print(s1.islower())
Output:
False
True
```

4.isupper():

- It is checking given data is upper case or not

Example:

```
s="ABCD"
print(s.isupper())
s1="abcd"
print(s1.isupper())
Output:
True
False
```

5.isdigit():

- It is checking given data is numeric or not

Example:

```
s="12345"
print(s.isdigit())
s1="abcd1234"
print(s1.isdigit())
```

output:

```
True
False
```

6.istitle():

- It checks given string is title case or not

Example:

```
s="Hello world "
print(s.istitle())
s1="Hello World "
print(s1.istitle())
```

output:

```
False
True
```

7.isspace():

- it is checking given string contains space or not

Example:

```
s="ABC  ABC"
print(s.isspace())
s1="   "
print(s1.isspace())
```

Output:

```
False
True
```

Example:

```
s="ABCD"
print(s.isalnum())
s1="ABC123"
print(s1.isalnum())
s2="ABC@123"
print(s2.isalnum())
```

Output:

```
True
True
False
```

Example:

```
s=input("Enter Data:")
if s.isalnum():
   if s.isalpha():
      if s.islower():
         print("It is alpha data with lower case")
      else:
         print("It is alpha data with  Upper case")
   elif s.isdigit():
      print("It is digit")
   else:
      print("It is alphanumeric")
elif s.isspace():
   print("It is space data")
else:
   print("It is special charcter or combination of alphanumeric and special charcters")
```

output:

```
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter Data:@#$
It is special charcter or combination of alphanumeric and special charcters
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter Data:
It is space data
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter Data:ABCD1234
It is alphanumeric
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter Data:12345
It is digit
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter Data:ABCD
It is alpha data with  Upper case
C:\Users\jagan\OneDrive\Desktop\Python_sessions>python test.py
Enter Data:abcd
It is alpha data with lower case
```

#Write a program to reverse programing.

Example:

```
s="Hello world"
print(s[::-1])
```

output:

```
dlrow olleH
```

Example:

```
s="Hello world"
r=reversed(s)
print(r)
print(type(r))
l=[]
for x in r:
   print(x)
   l.append(x)
print(l)
a=''.join(l)
print(a)
```

```
output:
<reversed object at 0x000001D9BFAEE748>
<class 'reversed'>
d
l
r
o
w

o
l
l
e
H
['d', 'l', 'r', 'o', 'w', ' ', 'o', 'l', 'l', 'e', 'H']
dlrow olleH
```

Example:

```
s="Hello"
output=""
i=len(s)-1#i=5-1=4
while i>=0:
   output=output+s[i]
   i=i-1
print(output)
output:
olleH
```

#Write program to remove duplicates

Example:

```
s="ABDDDC"
output=''
for ch in s:
   if ch not in output:
       output=output+ch
print(output)
output:
ABDC
```

Example:

```
s="ABDDDC"
l=[]
for ch in s:
   if ch not in l:
       l.append(ch)
out=''.join(l)
print(out)
output:
ABDC
```