Immutable data structure

1. String

Access the characters of string

1. By using index

In [3]:
```
s="Hello World"
print(s[2])
print(s[5])
print(s[20])
```

```
l

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-3-33226ec3376f> in <module>
      2 print(s[2])
      3 print(s[5])
----> 4 print(s[20])

IndexError: string index out of range
```

In [2]:
```
s="Arman"
print(s[3])
print(s[-2])
```

```
a
a
```

2. By using slicing operator

```
Syntax---> s[begin index:end index:step]
```

In [10]:
```
s="Learning Python is very easy."
print(s[1:7:1])
print(s[1:7])
print(s[:7])
print(s[5:])
print(s[1:7:2])
print(s[::2])
print(s[:])
print(s[::])
print(s[::-1]) #only for string not for number
print(s[-5::])
print(s[-5:-1:])
print(s[7:1:-1])
```

```
earnin
earnin
Learnin
ing Python is very easy.
eri
Lann yhni eyes.
Learning Python is very easy.
Learning Python is very easy.
.ysae yrev si nohtyP gninraeL
easy.
easy
gninra
```

Check whether the given string is palindrome or not

In [15]:
```
s=input("Enter string:")
str=s[::-1]
if(str==s):
    print(s,"is a palindrome string")
else:
    print(s,"not a palindrome string")
```

```
Enter string:sas
sas is a palindrome string
```

Mathematical operators for string

```
+ ---> String concatenation
* ---> String repetation
```

In [16]:
```
print("Arman"+"Arman")
print("Arman"*3)
```

```
ArmanArman
ArmanArmanArman
```

Comparison of String

In [20]:
```
s1=input("Enter string 1:")
s2=input("Enter string 2:")
if(s1==s2):
    print("Both strings are equal.")
elif(s1<s2):
    print("Second string is greater.")
else:
    print("First string is greater.")
```

```
Enter string 1:Arman\
Enter string 2:ARyan
First string is greater.
```

Joining of string

Join a group of strings wrt the given separator

Syntax---> s=separator.join(group of string)

In [21]:
```
t=("Arman","Aryan","Dhairya")
x="$".join(t)
print(x)
```

```
Arman$Aryan$Dhairya
```

Formatting of string

In [24]:
```
name="Aryan"
salary=40000
age=24
print("{}'s salary is {} and age is {}".format(name,salary,age))
print("{0}'s salary is {1} and age is {2}".format(name,salary,age))
```

```
Aryan's salary is 40000 and age is 24
Aryan's salary is 40000 and age is 24
```

Important functions of string

1. len()

In [25]:
```
s="Aryan"
print(len(s))
```

```
5
```

Removing spaces from string

1. lstrip()
2. rstrip()
3. strip()

In [27]:
```
s="banana "
print(len(s))
x=s.rstrip()
print(x)
print(len(x))
```

```
7
banana
6
```

In [29]:
```python
s="  banana"
print(s)
x=s.lstrip()
print(x)
```

```
  banana
banana
```

In [30]:
```python
s="  banana  "
print(s)
x=s.strip()
print(x)
```

```
  banana
banana
```

In [31]:
```python
s="banana"
x=s.rstrip("a")
print(x)
```

```
banan
```

In [32]:
```python
s="banana "
x=s.rstrip("a")
print(x)
```

```
banana
```

In [34]:
```python
s="banana"
x=s.rstrip("na")
print(x)
```

```
b
```

In [35]:
```python
s="bamana"
x=s.rstrip("na")
print(x)
```

```
bam
```

In [36]:
```python
s="banana"
x=s.lstrip("b")
print(x)
```

```
anana
```

Changing the case of string

1. upper()
2. lower()
3. swapcase()
4. title()
5. capitalize()

In [39]:
```python
s="Hello World"
x=s.upper()
y=s.lower()
print(x)
print(y)
z=s.swapcase()
print(z)
```

```
HELLO WORLD
hello world
hELLO wORLD
```

In [40]:
```python
s="HELLO HOW ARE YOU"
x=s.title()
print(x)
y=s.capitalize()
print(y)
```

```
Hello How Are You
Hello how are you
```

To check type of characters present in a string(check function)

---> Answer only in True or False

1. isalnum()

```
    Returns True if all characters are alphanumeric(a-z,A-Z,0-9)
```

In [41]:
```python
x="Company123"
print(x.isalnum())
```
```
True
```

In [42]:
```python
x="Company 123"
print(x.isalnum())
```
```
False
```

2. isalpha()
3. isdigit()
4. isnumeric()

In [45]:
```python
x="CompanyX"
print(x.isalpha())
y="Company 123"
print(y.isalpha())
```
```
True
False
```

In [46]:
```python
x="50525"
print(x.isdigit())
y="50525xyz"
print(y.isdigit())
```
```
True
False
```

Casing

1. islower()
2. isupper()

In [47]:
```python
t="hello world"
x=t.islower()
print(x)
```
```
True
```

In [48]:
```python
t="Hello"
x=t.isupper()
print(x)

```
```
False
```

3. istitle()

In [50]:
```python
t="Hello How Are You"
x=t.istitle()
print(x)
```
```
True
```

In [51]:
```python
a="22 Names"
b="This Is %?"
print(a.istitle())
print(b.istitle())
```
```
True
True
```

4. isidentifier()

In [54]:
```python
a="MyFolder"
b="Demo2002"
c="2bring"
d="my demo"
e="mu_demo"
print(a.isidentifier())
print(b.isidentifier())
print(c.isidentifier())
print(d.isidentifier())
print(e.isidentifier())
```
```
True
True
False
False
True
```

5. isspace()

In [55]:
```
t=" "
x=t.isspace()
print(x)
```

```
True
```

Count number of spaces

In [57]:
```
s="Hello How Are You"
count=0
for i in range(len(s)):
    if(s[i].isspace()):
        count+=1
    else:
        continue
print(count)
```

```
3
```

In [59]:
```
s="Hello How Are You"
count=0
for i in s:
    if(i.isspace()):
        count+=1
    else:
        continue
print(count)
print("No.of words:",count+1)
```

```
3
No.of words: 4
```

In [62]:
```
s="Hello How Are You"
charcount=0
lowcount=0
upcount=0
for i in s:
    if(i.isalpha()):
        charcount+=1
        if(i.islower()):
            lowcount+=1
        elif(i.isupper()):
            upcount+=1
print("Total:",charcount)
print("Lower:",lowcount)
print("Upper:",upcount)
```

```
Total: 14
Lower: 10
Upper: 4
```

In [67]:
```
s=input("enter string:")
n=len(s)
if(n%2==0):
    print(s)
else:
    mid=n//2
    print(s[0],s[mid],s[n-1])
```

```
enter string:James
J m s
```

In [69]:
```python
s="Py$t00567@23hon@_"
chcount=0
dicount=0
spcount=0
sum=0
for i in s:
    if(i.isalpha()):
        chcount+=1
    elif(i.isdigit()):
        dicount+=1
        sum=sum+int(i)
    else:
        spcount+=1
avg=sum/dicount
print(chcount)
print(dicount)
print(spcount)
print(sum)
print(avg)
```

```
6
7
4
23
3.2857142857142856
```

find()

Returns index of first occurence of the given substring if it is not available then we will get -1

Syntax---> s.find(substring)

In [9]:
```python
s="Learning Python is very easy."
print(s.find("a"))
print(s.find("s"))
print(s.find("x"))
print(s.find(" "))
print(s.find("Python"))
print(s.find("s v"))
print(s.find("a",3,))
```

```
2
17
-1
8
9
17
25
```

count()

In [14]:
```python
s="abcd abcxyz abcdefgh"
print(s.count("a"))
print(s.count("abc"))
print(s.count("abcd"))
print(s.count("i"))
print(s.count(" "))
print(s.count("a",8,15))
```

```
3
3
2
0
2
1
```

replace()

To replace old string with new string.

Syntax---> s.replace(old string,new string)

In [19]:
```python
s="Learning Java is easy."
x=s.replace("Java","Python")
y=s.replace("a","A")
print(x)
print(y)
```

```
Learning Python is easy.
LeArning JAvA is eAsy.
```

split()

```
split(separator)---> we can split the given string accordinf to specified separator by using split() method.
                ---> default separator is space
                ---> The return type of split() method is list
```

In [25]:

```python
s="Hello     World"
l=s.split()
m=s.split("l")
print(l)
print(m)
```

```
['Hello', 'World']
['He', '', 'o     Wor', 'd']
```

In [24]:

```python
s="29-10-2025"
l=s.split("-")
print(l)
```

```
['29', '10', '2025']
```

In [27]:

```python
s="abcd"
l=s.split("d")
print(l)
for i in l:
    print(i)
```

```
['abc', '']
abc
```

translate() with maketrans() function

In [29]:

```python
import string
print(string.punctuation)
print(len(string.punctuation))
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
32
```

maketrans():make translation table

```
mapping of character to their replacement or to none for deletion
```

Syntax---> maketrans(from_chars,to_chars,delete_chars)

translate()

```
- Applies to translation table created by maketrans()
- returns new string with characters replaced or deleted according to table.
```

In [36]:

```python
import string
s="Py$@tg!!on"
l=s.maketrans("","",string.punctuation)
x=s.translate(l)
y=s.maketrans("","","@$")
z=s.maketrans("n","m","@$")
print(x)
print(y)
print(z)
n=s.translate(z)
print(n)
```

```
Pytgon
{64: None, 36: None}
{110: 109, 64: None, 36: None}
Pytg!!om
```

In [38]:

```python
t="Hello Sam!!"
x="mSa"
y="eJo"
table=t.maketrans(x,y)
print(t.translate(table))
```

```
Hello Joe!!
```

Write a program to replace each special symbol with # for following string

In [50]:
```python
import string
s="/*John is @developer & musician!!"
for i in string.punctuation:
    s=s.replace(i,"#")
print(s)
```

```
##John is #developer # musician##
```

Write a program to remove ith character from the string

In [4]:
```python
s="Hello World"
i=int(input("Enter index:"))
l=s.replace(s[i],"",1)
print(l)
```

```
Enter index:2
Helo World
```

In [6]:
```python
s="Hello World"
i=int(input("Enter index:"))
x=s[:i]+s[i+1:]
print(x)
```

```
Enter index:2
Helo World
```

Write a program to find the count of all occurences of a substring in a given string by ignoring the case

In [8]:
```python
s="Welcome to USA. usa is awesome.Usa is good.Usain bolt is American."
l=s.lower()
print(l.count("usa"))
```

```
4
```

Write a program to display all positions of substring in a given string

In [12]:
```python
s="abcdabcacdab"
l=s.count("a")
sub="a"
pos=0
for i in s:
    if(i==sub):
        print(sub,"found on",pos,"position.")
    pos+=1
print("count:",l)
```

```
a found on 0 position.
a found on 4 position.
a found on 7 position.
a found on 10 position.
count: 4
```

Write a program to merge characters of two strings into a single string by taking characters alternatively

In [23]:
```python
x="abc"
y="123"
l=""
for i in range(len(x)):
    l=l+x[i]+y[i]
print(l)
```

```
a1b2c3
```

In [18]:
```python
s="abcabcdefxyzabcxyzab"
sub="abc"
pos=-1
flag=False
n=len(s)
while True:
    pos=s.find(sub,pos+1,n)
    if(pos==-1):
        break
    print("Found at position:",pos)
    flag=True
if flag==False:
    print("Not found.")
```

```
Found at position: 0
Found at position: 3
Found at position: 12
```

In [30]:
```python
s="a4b3c2"
sub=""
a=0
for i in s:
    if(i.isalpha()):
        a=i
    if(i.isdigit()):
        sub=sub+(a*int(i))
print(sub)
```

```
aaaabbbcc
```

Write a program to check the validity of a password primary conditions for password is given as below

```
minimum 8 characters
alphabets should be between [a-z]
Atleast one uppercase between[A-Z]
atleast one digit between[0-9]
atleast one character from[_,@,$]
```

In [12]:
```python
password=input("Enter password:")
upcount=0
digicount=0
sym="_@$"
symcount=0
othercount=0
if len(password)>=8:
    for i in password:
        if(i.isalpha()):
            if(i.isupper()):
                upcount+=1
        elif(i.isdigit()):
            digicount+=1
        elif(sym.find(i)>=0):
            symcount+=1
        else:
            othercount+=1
    if(upcount>=1 and digicount>=1 and symcount>=1 and othercount==0):# can use length of password too if sum of all cou
        print("Valid password")
    else:
        print("Invalid password")
else:
    print("Invalid length")
```

```
Enter password:Tejas@1234#
Invalid password
```

WAP to shift the decimal digits n places to the left wrapping the extra digits around if shift is greater than the no of digits of n then reverse the string

In [23]:
```python
n=input("Enter number:")
shift=int(input("Enter shift:"))
p=len(n)
if(p==shift):
    str=n[::-1]
    print(str)
else:
    print(n[shift:]+n[:shift])
```

```
Enter number:12345
Enter shift:3
45123
```

Tuple

- Tuple is same as list except it is immutable once we create tuple object we cant perform any changes in that object
- Tuple is read only version of list
- if our data is fixed and never changes hten we should go for tuple
- insertion order is preserved
- duplicates are allowed
- we can differentiate objects by using index.Hence index play important role in tuple
- heterogeneous objects are allowed
- Tuple supports both +ve and -ve indexing
- we can represent tuple elements within () with comma separator
- () are optional but recommended to use

Creation of tuple

In [24]:
```
t=()
print(type(t))
print(t)
```

```
<class 'tuple'>
()
```

In [27]:
```
t=(10)
print(t)
print(type(t))
```

```
10
<class 'int'>
```

In [28]:
```
t=(10,)
print(t)
print(type(t))
```

```
(10,)
<class 'tuple'>
```

In [29]:
```
t=10,20,30
print(t)
print(type(t))
```

```
(10, 20, 30)
<class 'tuple'>
```

In [31]:
```
t=tuple(range(10,20,2))
print(t)
```

```
(10, 12, 14, 16, 18)
```

Accessing elements of tuple

- By using index
- By using slicing operator

In [33]:
```
t=(10,20,30,40,50,60)
print(t[2])
print(t[4])
print(t[-2])
#print(t[100])
print(t[2:5])
print(t[:2])
print(t[4:])
print(t[2:100])
print(t[2:])
print(t[::2])
```

```
30
50
50
(30, 40, 50)
(10, 20)
(50, 60)
(30, 40, 50, 60)
(30, 40, 50, 60)
(10, 30, 50)
```

Mathematical operators of tuple

- 1.Concatenation operator(+)

In [34]:
```
t1=(10,20,30)
t2=(30,40,50)
t=t1+t2
print(t)
```

```
(10, 20, 30, 30, 40, 50)
```

- 2.Repitation operator(*)

In [35]:
```
t=(10,20,30)
t1=t*3
print(t1)
```

```
(10, 20, 30, 10, 20, 30, 10, 20, 30)
```

```
In [ ]:
```