

▼ DISCUSS STRING SLICING AND PROVIDE EXAMPLES

STRINGS :

Strings are combination of letters,symbols and numbers enclosed inside quotations:

STRING SLICING :

it is used to know a particular data which is assigned by a user to check whether it is available or not .
for example:

```
a= "MS Dhoni the untold story of the legend"  
print (a)
```

```
➦ MS Dhoni the untold story of the legend
```

if you want first 5 letters of the string that is mentioned above

```
print (a[0:5])
```

```
➦ MS Dh
```

if you want characters from 6th element to 12th element

```
print (a[6:13])
```

```
➦ ni the
```

if you want elements from the string in reverse order

```
print (a[-6:])
```

```
➦ legend
```

to execute a particular string by giving certain spans in middle:

```
print (a[0:20:2])
```

```
➦ M hn h nod
```

Start coding or [generate](#) with AI.

if you want to reverse the string totally .

```
print (a[::-1])
```

```
➦ dnegel eht fo yrots dlotnu eht inohD SM
```

Double-click (or enter) to edit

▼ EXPLAIN THE KEY FEATURES OF LISTS IN PYTHON

Lists are used to store a various kind of elements or objects inside a variable to assign when needed in a program. You can access it by giving index number. lists are mutable means it can be changed as per user convenience after creation. We use square brackets [] in order to give any input to program. In list all datatype / elements separated by comma(.). for example:

To access the second element from a given input and all elements from given input

lets take a collection of elements in a string

```
a=["spoon","Maggi","chow","bowl",]
print (a)
```

```
['spoon', 'Maggi', 'chow', 'bowl']
```

To find length of a list

```
print (len(a))
```

```
4
```

To count an occurrence of a particular element .

```
print (a.count ("chow"))
```

```
1
```

```
a
type(a[1])
#so,inorder to get banana's first 4 letter
a[1][0:4]
```

```
'bana'
```

to add an item to the list there are 2 functions that are used insert ()and append();

where insert function is used to insert anything from the given braces that user inputs and the second one append ()is used when a user wants to add a element but this function allows the user to add in end . examples:

```
a.insert(3,"maggi masala")
print (a)
a.append ("chowmein masala")
print (a)
```

```
['spoon', 'Maggi', 'chow', 'maggi masala', 'maggi masala', 'bowl']
['spoon', 'Maggi', 'chow', 'maggi masala', 'maggi masala', 'bowl', 'chowmein masala']
```

to create a copy of list

```
b=["vita marie ","top gold","jim jam","thin arrowroot "]
print (b)
```

```
['vita marie ', 'top gold', 'jim jam', 'thin arrowroot ']
```

to see a particular element what is present and to remove from that particular location

```
a=b.pop(1)
print (a)
```

```
top gold
```

to create a copy of list

```
c=b.copy()
print (c)
```

```
['vita marie ', 'jim jam', 'thin arrowroot ']
```

to swap the elements 1 with 3

```
c[1],c[2]=c[2],c[1]
print (c)
```

```
['vita marie ', 'thin arrowroot ', 'jim jam']
```

to extend the list

```
d=["oreo","bournvita"]
c.extend(d)
print(c)
```

```
['vita marie ', 'thin arrowroot ', 'jim jam', 'oreo', 'bournvita']
```

to sort the list *here this function is used to arrange the given strings in ascending order*

```
c.sort ()
print (c)
```

```
['bournvita', 'jim jam', 'oreo', 'thin arrowroot ', 'vita marie ']
```

```
#reverse index
```

```
a
a[-1]
```

```
'orange'
```

```
a[-3]
```

```
1.12
```

```
#negative indexing
a=list(a)
a
```

```
['apple', 'banana', True, 1.12, (2+3j), 'orange']
```

```
a[::-1]
```

```
['orange', (2+3j), 1.12, True, 'banana', 'apple']
```

```
a[-1:-5:-3]
```

```
['orange', True]
```

```
a[-5]
```

```
'banana'
```

DESCRIBE HOW TO ACCESS , MODIFY AND DELETE ELEMENTS IN A LIST WITH EXAMPLES

In order to access a element from a list . We need to first assign a string containing of certain names or characters in a variable string . example

```
access=["dabur red","colgate","pepsodent","sensodyne","babool"]
print (access.index('pepsodent'))
```

```
2
```

to add a new value at second position

```
access.insert (1,"vicco turmeric tooth paste")
print (access)
```

```
['dabur red', 'vicco turmeric tooth paste', 'vicco turmeric tooth paste', 'colgate', 'pepsodent', 'sensodyne', 'babool']
```

to clear all the data from the list

```
access.clear()
print (access)
```

```
[]
```

Start coding or [generate](#) with AI.

```
↳ ('apple', 'banana', True, 1.12, (2+3j), 'orange')
```

✓ COMPARE AND CONTRAST TUPLES AND LIST WITH EXAMPLES

- Tuples are the collection of ordered and unmutable data whereas as lists are unordered and mutable data .
- For tuples no brackets are mandatory but by default here the brackets used is () whereas in lists we use square brackets[].
- The value inside a list is separated by coma(,) as well as in lists.
- multiple ddatatypes can be written in tuple as well as it is also allowed in lists.
- when tuples once created caannot be changed whereas list is ulterable.
- iterations are faster in tuples in compare to lists,lists are time consuming.
- Tuples has only two functions i.e. count() and index() .

- examples of tuples:

```
a="apple","mango",1.23
print (type(a))
```

```
↳ <class 'tuple'>
```

```
a="apple","banana",True,1.12,2+3j
print(type(a))
```

```
↳ <class 'tuple'>
```

If you want to print a tuple set each and every item in a different line .

```
for i in a:
    print(i)
```

```
↳ apple
banana
True
1.12
(2+3j)
```

if you want to convert a tuple to list .

```
a
b=[]
for i in a:
    b.append(i)
print(b)
```

```
↳ ['apple', 'banana', True, 1.12, (2+3j)]
```

#to edit a tuple (insert some items in it.)

```
a
b=[]
for i in a:
    b.append(i)
b.append("orange")
print(b)
a=tuple(b)
print(a)
```


```
↳ ['apple', 'banana', True, 1.12, (2+3j), 'orange']
('apple', 'banana', True, 1.12, (2+3j), 'orange')
```

##list examples:

```
l1=["iphone","samsung","nokia","motorola"]
```


#suppose here you want to access if samsung is there then continue or else print that item is not existing .

```
for i in l1:
    if i == "samsung":
        print(i)
        continue
    else:
        print(i)
        break
```

 `iphone`

```
#to enter a list with combination of strings numbers and float values separate the output with two different list .
l=[1,2,3,1,2,46,595,9,5,323,595,"toronto","angela","daniels","augusta T"]
l1_num=[]
l2_str=[]
for i in l:
    if type(i)==int or type(i)==float:
        l1_num.append(i)

    else:
        l2_str.append(i)
print(l1_num)
print(l2_str)
```

 `[1, 2, 3, 1, 2, 46, 595, 9, 5, 323, 595]`
`['toronto', 'angela', 'daniels', 'augusta T']`


```
l1=[12,12,12,5,21,4,True,"dani","kiarra",3+6j]
l1[0]
```

 `12`


```
l1[5]
```

 `4`

```
l1[7]
```

 `'dani'`

```
l1[0:7]
```

 `[12, 12, 12, 5, 21, 4, True]`

Start coding or [generate](#) with AI.

✓ DESCRIBE THE KEY FEATURES OF SETS AND PROVIDE EXAMPLES OF THEIR USE.

- Sets are unordered.
- Set elements are unique. Duplicate elements are not allowed.
- A set itself may be modified, but the elements contained in the set must be of an immutable type.
- sets are usually get denoted by {}. if you denote it by giving values inside it then it will give output as it is set . But, if you do not give any values inside it then it will return you as an it is a dictionary because python does not understands that it is set or dictionary.

like for example:


```
s1={}
type (s1)
```

 `dict`

```
s1={1,2,3,5,45,6}
type (s1)
```

 `set`

```
s3={1,2,3,4,5,6,[23,12,5,4,6]}
s3
```



```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-4-d35f1b8779b2> in <cell line: 1>()
----> 1 s3={1,2,3,4,5,6,[23,12,5,4,6]}
      2 s3

TypeError: unhashable type: 'list'
```

```
# here, it is showing like this because set is immutable data it cannot store the data which is mutable ;it can only stores the data w
# you can easily sstiore a tuple inside it .
```

```
# example using tuple :
s4={1,2,3,5,4,23+6j,(12,65,6,323+9j)}
s4
```

```
{(12, 65, 6, (323+9j)), (23+6j), 1, 2, 3, 4, 5}
```

```
# why set is used ?
#so, lets create a set using various repetitive element.
s5={1,2,3,5,4,56,2211,3,5,4,2,21,2,3,6,5,4,0,3,65,2,2,58,5,5,23,5,4,2}
s5
```

```
{0, 1, 2, 3, 4, 5, 6, 21, 23, 56, 58, 65, 2211}
```

set will give only unique data. here , the sets use case is to get wherever the messy data or anything is given as a input in set , then it will give/produce only the unique one .

```
# for exmaple:
s6={1,25,3,154,62,"raja","Raja"}
s6
```

```
{1, 154, 25, 3, 62, 'Raja', 'raja'}
```

here, it is done so because the pyh=thon is a case sensitive language.

```
type(s6)
```

```
set
```

```
s6.add(25)
s6
```

```
{1, 154, 25, 3, 62, 'Raja', 'raja'}
```

```
s6.remove(25)
s6
```

```
{1, 154, 3, 62, 'Raja', 'raja'}
```

```
# here , in sets no indexing is provided because it is the collection of unordered data
# when there is no order then, how indexing is possible on it .here , only one works that is hashing .
```

```
#for example:
s6(1)
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-13-34ed595163bc> in <cell line: 2>()
      1 #for example:
----> 2 s6(1)

TypeError: 'set' object is not callable
```

see error has occured

✓ DISCUSS THE USE CASES OF TUPLES AND SETS IN PYTHON PROGRAMMING.

```
# see as we discussed earlier about what are tuples and
# sets and their respective use cases .
# so, in this section ii just want to give a certain overview of it that .
# tuples are the immutable datatype(means if they are once executed means they will not take any of the objects or values inside it and
# tuples allso do have functions in it. but they only have 2 functions inside it , they are count() and inside ()
# they are mainly used in making or keeping the store of passwords ,unique keys , tc.
# but, if you see in sets it is collection of unordered data .
# it can also store any type of data like other data types but here the only drawback is like tuple you cannot edit it .
# it does not allows tyou to keep the list type of data inside it.(syntax error:unhashable data .)
```

Start coding or [generate](#) with AI.

DESCRIBE HOW TO ADD,MODIFY AND DELETE ITEMS IN A DICTIONARY WITH EXAMPLES.

- BASIC RULES OF DICTIONARIES.
- dictionaries are denoted by {}
- dictionaries keys should be unique .
- it is mutable.

Start coding or [generate](#) with AI.

```
d={}
type(d)
```


 dict

```
d1={'key':'raja'}
```


```
d2={'name':'praveen','email':'gprkmr@gmail.com','phoone':917748922}
d2
```

 {'name': 'praveen', 'email': 'gprkmr@gmail.com', 'phoone': 917748922}

```
#this function is used to delete an item from last.
d2.popitem()
```

 ('phoone', 917748922)

```
#this function is used to modify a given dictionary if there is any mistake.
d2.update({'name':'raja'})
d2
```

 {'name': 'raja', 'email': 'gprkmr@gmail.com'}

```
#to add anything inside a dictionary.
d2["color"]="red"
d2
```

 {'name': 'raja', 'email': 'gprkmr@gmail.com', 'color': 'red'}

DISCUSS THE IMPORTANCE OF DICTIONARY KEYS BEING IMMUTABLE AND PROVIDE EXAMPLES

Start coding or [generate](#) with AI.

```
# see in dictionary you probably do keep values with particular keys over here to access.
# dictionaries are mainly used to keep them in a precision manner that when you want to access it you can simply go and check for function
# it is used to find the large data but we should have been kept them in key-value manner in order to access.
```


for example:

```
dict={'noodles':['maggi','yippee','topramen'],'atta':['parle-g atta ', 'aashirvaad atta'],'paste':['dabur red ', 'colgate']}
```

```
#if you want to know what are the keys that are available in that particular dictionary.
dict.keys()
```

 dict_keys(['noodles', 'atta', 'paste'])

```
# if you want to know that what are present inside that particular dictionary with that key value pair.
dict.items()
```

 dict_items([('noodles', ['maggi', 'yippee', 'topramen']), ('atta', ['parle-g atta ', 'aashirvaad atta']), ('paste', ['dabur red ', 'colgate'])])

```
# if you want to see that what particularly present in that dictionary on a particular key.  
dict.get('paste')
```

```
📄 ['dabur red ', 'colgate']
```

```
#in order to clear a particular dictionary with key-value pairs  
dict.clear()  
dict
```

```
📄 {}
```