

Unit 3. Data structures in Python

3.1 List

- The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets.
- Important thing about a list is that items in a list need not be of the same type.
- List are mutable, and hence, they can be appended and updated even after its creation.

3.1.1 Defining list

- A list can be declared by placing all elements (items) inside a square bracket ([]) separated by comma (,).
- We can either declare an empty list and append it later as required or we can directly initialize a list.
- For example,

```
>>> list1=[] #empty list
>>> list2=[10,20,30]
>>> print(list2)
[10, 20, 30]
>>> list3=[10,"Ravi",3.11,1458566]
>>> print(list3) #list is initialized with mixed datatype
[10, 'Ravi', 3.11, 1458566]
>>>
```

3.1.2 Accessing list elements using index numbers

- We can use indexing operator [] to access list elements, which starts from 0.

```
>>> list1=[] #empty list
>>> list2=[10,20,30]
>>> print(list2)
[10, 20, 30]
>>> list3=[10,"Ravi",3.11,1458566]
>>> print(list3) #list is initialized with mixed datatype
[10, 'Ravi', 3.11, 1458566]
>>> print(list3[0])
10
>>> print(list2[2])
30
>>>
```

- Index error : list index out of range.
- And one more thing, this index must be an integer. We can't use float or other types, this will result into "TypeError".
- Eg. Print(list2[1.2])
- Like other programming language, we can also iterate through index numbers by using while loop

```
list1= [10,25.66,"Ravi",145,200]
print("List elements are:")
i=0
while i<list1.__len__():
    print("At index",i,"element is:",list1[i])
    i=i+1
```

while i<list1.__len__()

operators ×

C:\Users\Swanali\PycharmProjects\python_project

List elements are:
At index 0 element is: 10
At index 1 element is: 25.66
At index 2 element is: Ravi
At index 3 element is: 145
At index 4 element is: 200

3.1.3 Negative list indexing

Python allows negative indexing to its elements.

```
list1= [10,25.66,"Ravi",145,200]
print("At index 0,we have:",list1[0])
print("At index 0,we have:",list1[-1])
print("At index 0,we have:",list1[-2])
```

operators ×

C:\Users\Swanali\PycharmProjects\python_

At index 0,we have: 10
At index 0,we have: 200
At index 0,we have: 145

3.1.4 Iterating through list using for-loop

```
list1= ["C","c++","java","python","perl"]
for i in list1:
    print("loop counter refers to element:",i)

for i in range(1,11):
    print(i)
    |
```

3.1.5 Slicing lists

```
#!/usr/bin/python
list = ['abcd', 786, 2.23, 'john', 70.2]
tinylist = [123, 'john']
print(list)          # Prints complete list
print(list[0])        # Prints first element of the list
print(list[: -2])     # prints elements upto -2 index
print(list[1:3])      # Prints elements starting from 2nd till 3rd
print(list[2:])       # Prints elements starting from 3rd element
print(list[::-1])     # printing list of elements in reverse order
print(tinylist * 2)   # Prints list two times
print(list + tinylist) # Prints concatenated lists
```

operators ×

```
C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.
['abcd', 786, 2.23, 'john', 70.2]
abcd
['abcd', 786, 2.23]
[786, 2.23]
[2.23, 'john', 70.2]
[70.2, 'john', 2.23, 786, 'abcd']
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

3.1.6 Basic List Operations

3.1.6.1 Adding element: append(), insert(), extend() functions

1. append():

It adds element at last position of list

Syntax: list_name.append(new_element)

```
list1=[]
print("Enter 5 elements for list:")
i=1
while i<=5:
    val=input()
    list1.append(val)
    i=i+1
print("List contains:",list1)
```

while i<=5

operators ×

C:\Users\Swanali\PycharmProjects\python_project\venv\
 Enter 5 elements for list:
 10
 swap
 30.9
 333
 122
 List contains: ['10', 'swap', '30.9', '333', '122']

2. insert()

This function inserts an elements at specified position.

Syntax : list_name.insert (position/insert, element)

```
list1=["Mathematics","Chemistry","Physics","Biology","English"]
print("Initially list contains:")
print(list1)
list1.insert(2,"Sanskrit")
print("\n After inserting Sanskrit at index 2,list contains:")
print(list1)
```

operators ×

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:
 Initially list contains:
 ['Mathematics', 'Chemistry', 'Physics', 'Biology', 'English']

 After inserting Sanskrit at index 2,list contains:
 ['Mathematics', 'Chemistry', 'Sanskrit', 'Physics', 'Biology', 'English']

3. extend()

This function adds elements of parameter list with invoking list.

Syntax: list1.extend(list2)

```
list1=[10,20,30]
list2=[15,35,45,55]
list1.extend(list2)
print("After extention list1 contains:",list1)
print("After extention list2 contains:",list2)
```

operators ×

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python
 After extention list1 contains: [10, 20, 30, 15, 35, 45, 55]
 After extention list2 contains: [15, 35, 45, 55]

3.1.6.2 Updating element

List is mutable entity i.e we can update and over-write its element by using assignment operator (=).

```
list1=[1,2,3,4,5,6,7,8,9]
print("initially list contains:",list1)
i=0
while i<list1.__len__():
    list1[i]=list1[i]*list1[i]
    i=i+1
print("After updating list will be:",list1)
```

operators ×

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python
 initially list contains: [1, 2, 3, 4, 5, 6, 7, 8, 9]
 After updating list will be: [1, 4, 9, 16, 25, 36, 49, 64, 81]

3.1.6.3 Delete or remove elements from list

1. `del` : keyword, used to remove one or more element from list.

```
>>> list1=[1,2,3,4,5,6,7,8]
>>> print("original list contains:")
original list contains:
>>> print(list1)
[1, 2, 3, 4, 5, 6, 7, 8]
>>> del list1[2]
>>> print("After deleting element from 2nd index,list:",list1)
After deleting element from 2nd index,list: [1, 2, 4, 5, 6, 7, 8]
>>> del list1[1:5]
>>> print("After deleting element from index 1 to 5 :",list1)
After deleting element from index 1 to 5 : [1, 7, 8]
>>>
```

2. `remove()` : removes specified item from list

Syntax: `list_name.remove(element)`

3. `pop()` : This function removes element from specified index. It's syntax is:

`list_name.pop(specified index_number)`

4. `clear()` : This function removes all elements from a List. It's syntax is:

`list_name.clear()`

```
>>>
>>> list1=[1,2,3,4,2,5,6,7,4]
>>> print(list1)
[1, 2, 3, 4, 2, 5, 6, 7, 4]
>>> list1.remove(2)
>>> print("After removing 2 from list:",list1)
After removing 2 from list: [1, 3, 4, 2, 5, 6, 7, 4]
>>> list2=['p','y','t','h','o','n','p','r','o','g','r','a','m']
>>> print(list2)
['p', 'y', 't', 'h', 'o', 'n', 'p', 'r', 'o', 'g', 'r', 'a', 'm']
>>> list2.remove('p')
>>> print("After removing p from list :",list2)
After removing p from list : ['y', 't', 'h', 'o', 'n', 'p', 'r', 'o', 'g', 'r', 'a', 'm']
>>> list2.pop(4)
'n'
>>> print(list2)
['y', 't', 'h', 'o', 'p', 'r', 'o', 'g', 'r', 'a', 'm']
>>> list1.clear()
>>> print("after clearing operation,list contains:",list1)
after clearing operation,list contains: []
>>>
```

3.1.6.4 Testing Membership Operators on List

Operator	Description
in	Results True, if specified left operand element is present in specified right operand sequence; otherwise results False.
not in	Results True, if specified left operand element is not present in specified right operand sequence; otherwise results False.

```

membership.py - C:/Users/Swanali/AppData/Local/Programs/Python/Python3
File Edit Format Run Options Window Help
a = 10
b = 20
list = [1, 2, 3, 4, 5 ];

if ( a in list ):
    print ("a is available in the given list")
else:
    print (" a is not available in the given list")

if ( b not in list ):
    print (" b is not available in the given list")
else:
    print (" b is available in the given list")

a = 2
if ( a in list ):
    print ("a is available in the given list")
else:
    print ("a is not available in the given list")

```

```

== RESTART: C:/Users/Swanali/AppData/Loc
a is not available in the given list
b is not available in the given list
a is available in the given list

```


3.1.7 Built-in List Functions

Sr. No.	Functions
1	append() - append(element) Add an element to the end of the list
2	extend() - extend(sequence) Add all elements of a list to the another list
3	insert() - insert(index,element) Insert an item at the defined index
4	remove() - remove(element) Removes an item from the list
5	pop() - pop(index) Removes and returns an element at the given index
6	clear() - clear() Removes all items from the list
7	index() - index(element_to_search) Returns the index of the first matched element
8	count() - count(element) Returns the count of number of items passed as an argument
9	sort() - sort() Sort items in a list in ascending order
10	reverse() - reverse() Reverse the order of items in the list
11	copy() - copy() Returns a copy of the list

3.1.8 Nested List (Multi-Dimensional List)

Nested List means, a List as an element of another list. For example,

```
List1= [10. 20, [30, 40, 50], 60, ["Ravi", 70, 42.11, 299], 78,"Ram", 5.57]
```

For ease of understanding, let us put above list1 in form of an array. So that we can get its exact index number .

10	20	30	40	50	60	"Ravi"	40	42.11	299	78	"Ram"	5.57
		0	1	2		0	1	2	3			
0	1	2			3		4			5	6	7

Now we can see that. we have nested list at index 2 and 4. These nested lists also have index numbers.

Now we can clearly observe that, the elements

- 20 is identified as : list1[1]
- 30 is identified as :list1[2][0] # because it is at 0th index of nested list present at 2nd index
- "Ravi" is identified as : list1[4][0]
- 42.11 is identified as : list1[4][2]
- 5.57 is identified as : listl[7]

```
>>> list1=[10,20,[30,40,50],60,['Ravi',70,42.11,299],78,"Ram",5.57]
>>> for ele in list1:<print<ele>>
...
10
<None>
20
<None>
[30, 40, 50]
<None>
60
<None>
['Ravi', 70, 42.11, 299]
<None>
78
<None>
Ram
<None>
5.57
<None>
>>>
```

```
list1=[[1,2,3],[4,5,6],[7,8,9]]
list2=[[1,2,3],[4,5,6],[7,8,9]]
list3=[]
print("list1 contains:")
i=0
while i<list1.__len__():
    j=0
    while j<list1.__len__():
        print(list1[i][j],end=" ")
        j=j+1
```

```
print()
i=i+1
print("list2 contains:")
i=0
while i<list2.__len__():
    j=0
    while j<list2.__len__():
        print(list2[i][j],end=" ")
        j=j+1
    print()
    i=i+1

i=0
while i<=2:
    j=0
    list3.append([])
    while j<=2:
        sum =list1[i][j]+list2[i][j]
        list3[i].append(sum)
        j=j+1
    print()
    i=i+1

print("list3 contains:")
i=0
while i<list3.__len__():
    j=0
    while j<list3.__len__():
        print(list3[i][j],end=" ")
        j=j+1
    print()
    i=i+1
```

3.1.9 Deleting List

After performing required operations on List, if the user feels to delete/free the entire List; then it can also be done by using keyword 'del'. It is like releasing/freeing memory of specified List, during runtime.

```
list1 = [10,20,30,"Ravi",50]
print("Initially, List contains : ", list1)
del list1                                # will delete list1 during runtime
print("After 'del' operation, List contains : ",list1)
```

Above code will show following error :

```
Traceback (most recent call last):
  File "C:/Users/Majithia/PycharmProjects/MSBTE_Py_Book_All_Progs/test54.py", line 4, in <module>
    print("After 'del' operation, List contains : ",list1)
NameError: name 'list1' is not defined
```

3.2 Tuples

- A Tuple is a collection of elements/objects separated by comma (,)
- In someway a Tuple is similar to a List in term of indexing, nested objects and repetition; but the difference between these two is that we cannot change the elements of a Tuple once it is assigned; whereas, in a List, elements can be changed

3.2.1 Declaring Tuple OR Declaring Tuple

To declare a Tuple, we simply have to apply set of multiple elements inside parenthesis (()).

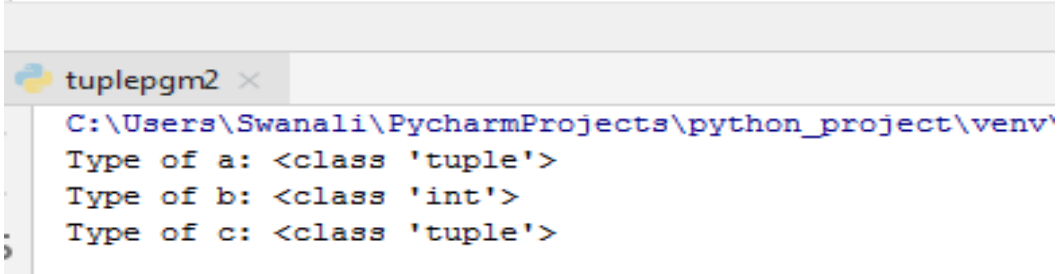
```
tuple1=(10,20,30,445.52,'Swapnali',55,'YBP')
tuple2=(2.22,4,50,6.7,56)
tuple3=(tuple1,tuple2,8,55,'ABC')
tuple4=(14,15,[89,45,23],554,('p','y','t','h','o','n'),350)
tuple5=()
print("Tuple1 contains:",tuple1)
print("Tuple2 contains:",tuple2)
print("Tuple3 contains:",tuple3)
print("Tuple4 contains:",tuple4)
print("Tuple5 contains:",tuple5)
```

```
tuplepgm x
C:/Users/Swanali/PycharmProjects/python_project/venv/Scripts/python.exe C:/Users/Swanali/PycharmProjects/
Tuple1 contains: (10, 20, 30, 445.52, 'Swapnali', 55, 'YBP')
Tuple2 contains: (2.22, 4, 50, 6.7, 56)
Tuple3 contains: ((10, 20, 30, 445.52, 'Swapnali', 55, 'YBP'), (2.22, 4, 50, 6.7, 56), 8, 55, 'ABC')
Tuple4 contains: (14, 15, [89, 45, 23], 554, ('p', 'y', 't', 'h', 'o', 'n'), 350)
Tuple5 contains: ()

Process finished with exit code 0
```

Important point is, an empty Tuple with parenthesis is valid, but it is of no use; because Tuple is immutable and we cannot change/update its element, if required.

```
a=()
b=(10)
c=(10,20)
print("Type of a:",type(a))
print("Type of b:",type(b))
print("Type of c:",type(c))
```

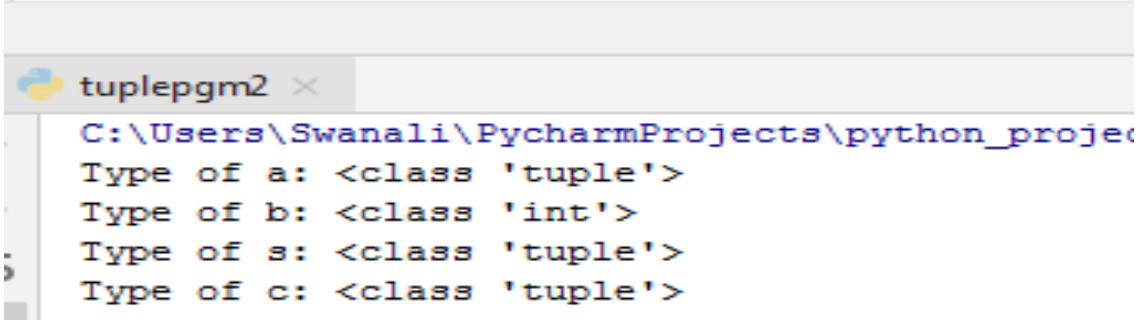


```
C:\Users\Swanali\PycharmProjects\python_project\venv
Type of a: <class 'tuple'>
Type of b: <class 'int'>
Type of c: <class 'tuple'>
```

Note that, single element in Parenthesis will not create Tuple, it either require no or minimum two elements

However creating a Tuple with one element is a bit tricky. Having one element within parentheses is not enough. We need a trailing comma to indicate that it is, in fact, a Tuple

```
a=()
b=(10)
s=(10,)
c=(10,20)
print("Type of a:",type(a))
print("Type of b:",type(b))
print("Type of s:",type(s))
print("Type of c:",type(c))
```



```
C:\Users\Swanali\PycharmProjects\python_project
Type of a: <class 'tuple'>
Type of b: <class 'int'>
Type of s: <class 'tuple'>
Type of c: <class 'tuple'>
```

3.2.2 Accessing Tuple Elements using Index Numbers

```
a=(10,20,30,452.22,'Swap',54545,15.22)
print("At index 0, we have:",a[0])
print("At index 3, we have:",a[3])
i=0
while i<a.__len__():
    print("at index",i,"element is:",a[i])
    i=i+1
```

```
while i<a.__len__()
tuplepgm2 x
C:\Users\Swanali\PycharmProjects\python_project'
At index 0, we have: 10
At index 3, we have: 452.22
at index 0 element is: 10
at index 1 element is: 20
at index 2 element is: 30
at index 3 element is: 452.22
at index 4 element is: Swap
at index 5 element is: 54545
at index 6 element is: 15.22

Process finished with exit code 0
```

3.2.3 Negative Tuple indexing

```
tuple1=(10,20,30)
print("At index -1, we have :",tuple1[-1])
print("At index -2, we have :",tuple1[-2])
print("At index -3, we have :",tuple1[-3])

At index -1, we have : 30
At index -2, we have : 20
At index -3, we have : 10
```

3.2.4 Iterating through Tuple using for-loop

```

a=(10,20,30,452.22,'Swap',54545,15.22)
print("At index 0, we have:",a[0])
print("At index 3, we have:",a[3])
i=0
while i<a.__len__():
    print("at index",i,"element is:",a[i])
    i=i+1
tuple1 = (10, 20, 30, 458.55, 'Ravi', 54441, 15.71)

for ele in tuple1 : #You can iterate through tuple by using for loop
    print(ele,end="|")

for ele in tuple1
tuplepgm2 x
C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe
At index 0, we have: 10
At index 3, we have: 452.22
at index 0 element is: 10
at index 1 element is: 20
at index 2 element is: 30
at index 3 element is: 452.22
at index 4 element is: Swap
at index 5 element is: 54545
at index 6 element is: 15.22
10,20,30,458.55,Ravi,54441,15.71,
Process finished with exit code 0

```

3.2.5 Slicing Tuple

```

tuple1=('P','Y','T','H','O','N','P','R','O','G','R','A','M')
print("All elements in tuple:",tuple1)
print("Elements slicing from index 6 upto last:",tuple1[6:])
print("Elements slicing upto index -5 :",tuple1[:-5])
print("Elements:",tuple1[:])
print("Printing all elemrnts in reverse order:",tuple1[::-1])

```

```

tuplepgm3 x
C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/Swanali/PycharmProjects/f
All elements in tuple: ('P', 'Y', 'T', 'H', 'O', 'N', 'P', 'R', 'O', 'G', 'R', 'A', 'M')
Elements slicing from index 6 upto last: ('P', 'R', 'O', 'G', 'R', 'A', 'M')
Elements slicing upto index -5 : ('P', 'Y', 'T', 'H', 'O', 'N', 'P', 'R')
Elements: ('P', 'Y', 'T', 'H', 'O', 'N', 'P', 'R', 'O', 'G', 'R', 'A', 'M')
Printing all elemrnts in reverse order: ('M', 'A', 'R', 'G', 'O', 'R', 'P', 'N', 'O', 'H', 'T', 'Y', 'P')

```

3.2.6 Basic Tuple Operations

3.2.6.1 Tuples are Immutable

- It means, we cannot modify Tuple elements, once they are initialized and created. It will show `TypeError: 'tuple' object does not support item assignment`. Run following code to get this error.

```
tuple1 = (10,20,"Ravi",5.114)
```

```
tuple1[1] = 45 # will show "TypeError"
```

- Even Tuple does not have `append()` function any other modification related function to change or update Tuple elements.
- The only thing we can do is concatenate Tuples or single element and receive it in same Tuple reference.

```
tuple1 = (10,20,"Ravi",5.114)
```

```
tuple1 = tuple1 + (50,60)
```

```
print("Tuple contain:", tuple1)
```

This will result as:

```
Tuple contains: (10,20,"Ravi",5.114, 50,60)
```

3.2.6.2 Testing Membership Operators on Tuple

```
tuple1 = (10,40,20.555,"Ravi",100)
print("Element 10 is present in tuple1 :", 10 in tuple1)
print("Element 25 is present in tuple1 :", 25 in tuple1)
print("Element 10 is not present in tuple1 :", 10 not in tuple1)
print("Element 25 is not present in tuple1 :", 25 not in tuple1)
```

It will show following output :

```
Element 10 is present in tuple1 : True
Element 25 is present in tuple1 : False
Element 10 is not present in tuple1 : False
Element 25 is not present in tuple1 : True
```


3.2.6.3 Concatenation of Tuple

```
tuple1 = (10,20,"Ravi",5.114)
```

```
tuple2= (30, " Ram", 28.6)
```

```
tuple3 = tuple1 + tuple2
```

```
print(tuple3)
```

This will display following output :

```
(10, 20, 'Ravi', 5.114, 30, 'Ram', 28.6)
```

```
tuple4 = tuple1 , tuple2
```

```
prin(tuple4)
```

This will create nested tuple4, which contains tuple1 at index 0 and tuple2 at index 1. It will create following tuple

```
((10,20, 'Ravi', 5.114), (30, 'Jaineel', 28.6))
```

3.2.6.4 Multiplying Tuple

Multiplying a Tuple by any integer x will simply return a new Tuple with all the elements from the first Tuple being repeated x number of times.

```
tuple1 = (10,20,"Ravi",5.114)
```

```
tuple2 = tuple1 * 3
```

```
print("Tuple 2 contains: ",tuple2)
```

This will result as:

```
Tuple-2 contains : (10, 20, 'Ravi', 5.114, 10, 20, 'Ravi', 5.114, 10, 20, 'Ravi', 5.114)
```

3.2.7 Built-in Tuple Functions

1	count() - count(element) Returns the count of occurrence of specified element.
2	index() - index(element) Returns the index of the first matched element
3	__add__() - __add__(tuple) Adds specified Tuple in invoking Tuple object and returns new updated Tuple object.

```

tuple1=(10,20,30)
print("Initially tuple1 contains:",tuple1)
tuple2 = tuple1.__add__((25,85,20,95,99))
print("After add operation tuple2 contains:",tuple2)
a= tuple2.count(20)
print("Element 20 appears",a,"times in tuple2")
b= tuple2.index(20)
print("First occurrence of 20 in tuple2 is at index:",b)

```

tuplepgm4 x

```

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe
Initially tuple1 contains: (10, 20, 30)
After add operation tuple2 contains: (10, 20, 30, 25, 85, 20, 95, 99)
Element 20 appears 2 times in tuple2
First occurrence of 20 in tuple2 is at index: 1

```

3.2.8 Deleting Tuple

```

tuple1=(10,20,30)
print("Initially tuple1 contains:",tuple1)
del tuple1
print("After del operation:",tuple1)

```

```

Initially tuple1 contains: (10, 20, 30)
print("After del operation:",tuple1)
NameError: name 'tuple1' is not defined

```

3.3 set

Set Python's built-in data-structure that has following characteristics.

- 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.

3.3.1 Declaring Set OR Defining Set OR Creating Set

```

set1={10,20,30}
print(set1)
set2 = {1.45, 244, "Hello", (1, 2, 3)}
print(set2)

```

This will show following output :

Set-1 contains : {10, 20, 30}

set-2 contains : {1.45, 244, (1, 2, 3), 'Hello'}

Note that, both above sets contain immutable elements — Tuple and String. But we cannot add a List as element of Set. Here is an invalid set declaration.

```
Set1= {1.45, 244, "Hello", (1, 2, 3),[10,20,30] }
```

```
Set2= {1.445,244, "hello", (1,2,3),{1: "one",2: "two"}}
```

Both above set declaration are invalid because both these contains mutable elements — List and Dictionary.

```
a = { } #by-default creates Dictionary
```

```
print("Type is :",type(a))
```

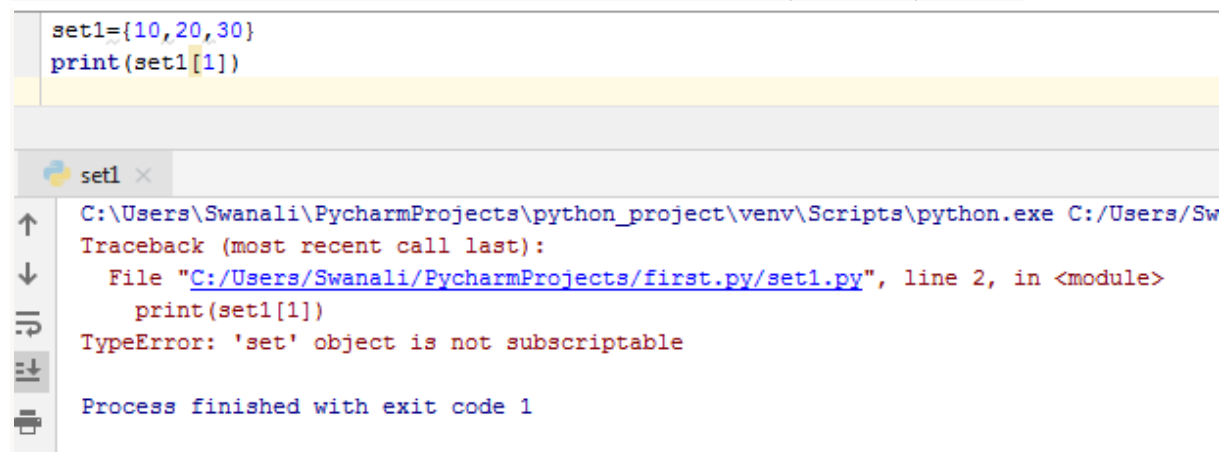
If the same is type-casted into Set, then creates Set. Here is an example code.

```
a =set({ }) # now creates Set
```

```
print(" Type is: ", type(a))
```

3.3.2 'Set' is not Subscriptable OR No indexing to 'Set' Elements OR 'Set' Object does not Support Indexing

Unlike Lists and Tuples, the Set does not support indexing i.e. its elements will not have index numbers to individually identify them.



```
set1={10,20,30}
print(set1[1])
```

set1 x

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/Sw
 Traceback (most recent call last):
 File "C:/Users/Swanali/PycharmProjects/first.py/set1.py", line 2, in <module>
 print(set1[1])
 TypeError: 'set' object is not subscriptable

Process finished with exit code 1

```
set1={10,20,30}
i=0
while i<set1.__len__():
    print(set1[i])
    i=i+1
```

set1 x

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/
 Traceback (most recent call last):
 File "C:/Users/Swanali/PycharmProjects/first.py/set1.py", line 4, in <module>
 print(set1[i])
 TypeError: 'set' object is not subscriptable

3.3.3 Iterating through set using for-loop

A set does not support indexing, but it can be iterated using for-loop.

```
set1={10,20,30}
for i in set1:
    print(i)
```

set1 x

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/Swana
 10
 20
 30

3.3.4 Basic Set Operations

3.3.4.1 Adding Element : update() and add() Functions

We can add single element using the add() function and multiple elements using the update() function.

The update() function can take Tuples, Lists, Strings or other Sets as its parameter. In all cases duplicates are avoided.

```
set1={10,20,30}
print("Initially, set contains:",set1)
set1.add(40)
print("After adding '40',set contains:",set1)
set1.update([20,30,40,50])
print("After updating,set contains:",set1)
set1.update([60,70],[70,80,90])
print("After adding list and set,set contains:",set1)
```

set1 x

C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/Swana
 Initially, set contains: {10, 20, 30}
 After adding '40',set contains: {40, 10, 20, 30}
 After updating,set contains: {40, 10, 50, 20, 30}
 After adding list and set,set contains: {70, 40, 10, 80, 50, 20, 90, 60, 30}

3.3.4.2 Finding Length of Set : using `__len__ ()` Function

The `__len__()` function returns length (numbers of elements) present in a set.

```
set1 = {10,20,54.88,"Ravi",62.14}
a = set1.__len__()
print("set1 contains",a,"elements")
```

This will result as :

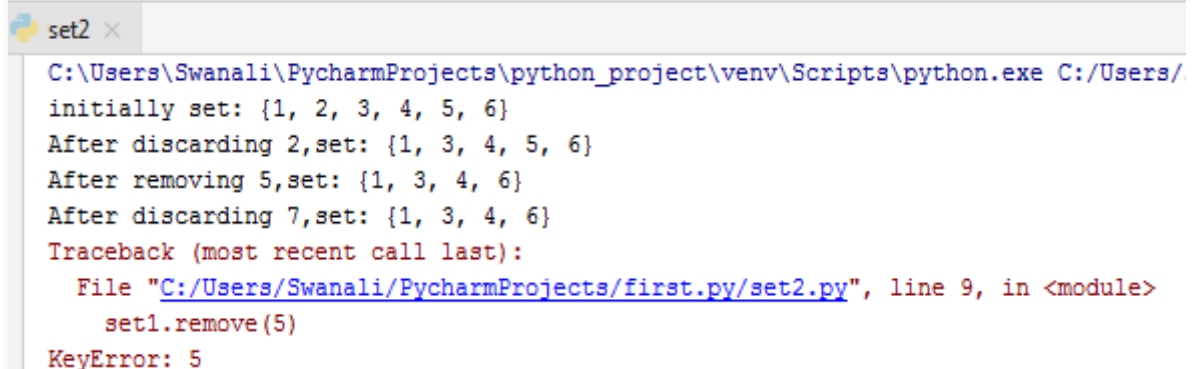
set1 contains 5 elements

3.3.4.3 Removing Element : `discard()`, `remove()`, `pop()` and `clear()` Functions

A particular item can be removed from set using functions, `discard()` and `remove()`.

The only difference between the two is that, while using `discard()` if the item does not exist in the set, it remains unchanged. But `remove()` will raise 'KeyError' in such condition.

```
set1={1,2,3,4,5,6}
print("initially set:",set1)
set1.discard(2)
print("After discarding 2,set:",set1)
set1.remove(5)
print("After removing 5,set:",set1)
set1.discard(7)
print("After discarding 7,set:",set1)
set1.remove(5)
print("After removing 5,set:",set1)
```



```
set2 x
C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/
initially set: {1, 2, 3, 4, 5, 6}
After discarding 2,set: {1, 3, 4, 5, 6}
After removing 5,set: {1, 3, 4, 6}
After discarding 7,set: {1, 3, 4, 6}
Traceback (most recent call last):
  File "C:/Users/Swanali/PycharmProjects/first.py/set2.py", line 9, in <module>
    set1.remove(5)
KeyError: 5
```

Similarly, we can remove and return an item using the `pop()` function. Set being unordered, there is no way of determining which item will be popped. It is completely arbitrary. We can also remove all items from a set using `clear()`.

```

set1=set("PYTHON")
print("Initially set:",set1)
popped_ele1=set1.pop()
print("Element",popped_ele1,"is popped from set")
print("set:",set1)
popped_ele2=set1.pop()
print("Element",popped_ele2,"is popped from set")
print("set:",set1)
popped_ele3=set1.pop()
print("Element",popped_ele3,"is popped from set")
print("set:",set1)
set1.clear()
print("After using clear method,set:",set1)

```

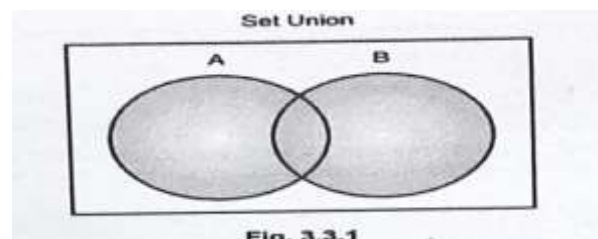
```

set3 x
C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\p
Initially set: {'O', 'T', 'H', 'Y', 'N', 'P'}
Element O is popped from set
set: {'T', 'H', 'Y', 'N', 'P'}
Element T is popped from set
set: {'H', 'Y', 'N', 'P'}
Element H is popped from set
set: {'Y', 'N', 'P'}
After using clear method,set: set()

```

3.3.4.4 Set Union

Union of two sets is a set of all elements from both sets. Union is performed using pipe operator (`|`) operator. Same can be accomplished using the function `union()`.



```

A={1,2,3,4,5}
B={6,7,8,9,10}
#C=A|B
C=A.union(B)
print("set A:",A)
print("set B:",B)
print("Union of A and B:",C)

```

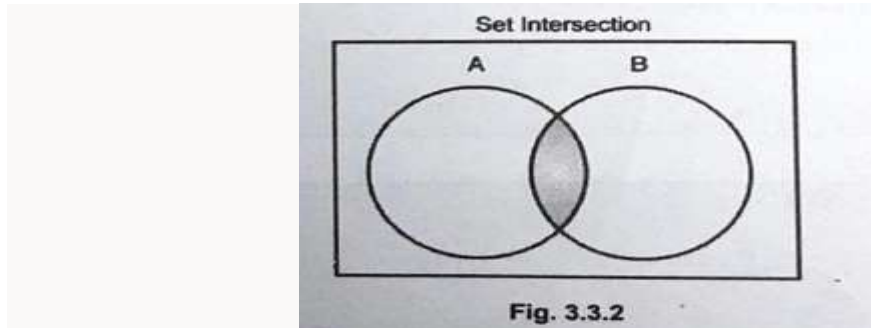
```

set4 x
C:\Users\Swanali\PycharmProjects\python_project\venv\Sc
set A: {1, 2, 3, 4, 5}
set B: {6, 7, 8, 9, 10}
Union of A and B: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

```

3.3.4.5 Set Intersection

Intersection of A and B is a set of elements that are common in both sets. The Intersection is performed using (&) operator. Same can be accomplished using the function intersection()

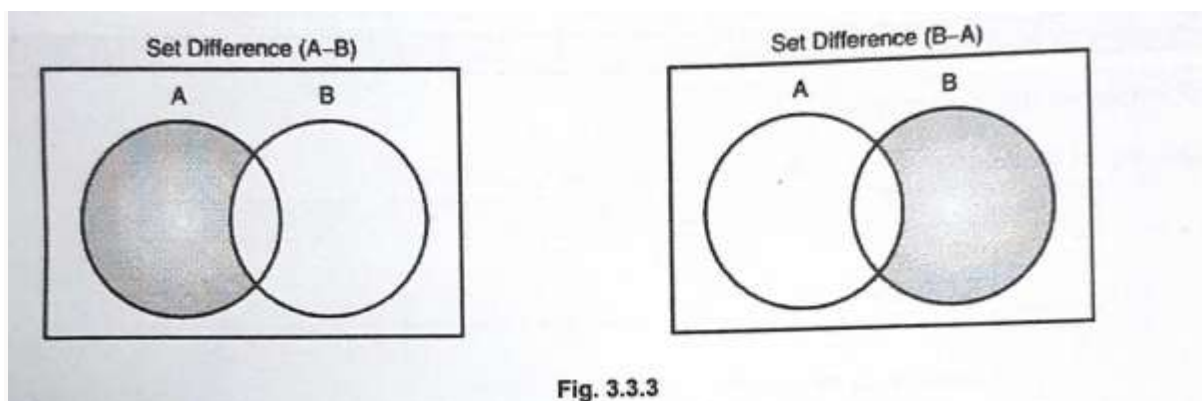


```
A={1,2,3,4,5}
B={5,7,4,9,10}
#C=A&B
C=A.intersection(B)
print("set A:",A)
print("set B:",B)
print("Intersection of A and B:",C)
```

```
set4 x
C:\Users\Swanali\PycharmProjects\python_pr
set A: {1, 2, 3, 4, 5}
set B: {4, 5, 7, 9, 10}
Intersection of A and B: {4, 5}
```

3.3.4.6 Set Difference

Difference of A and B ($A - B$) is a set of elements that are only in A but not in B. Similarly, $B - A$ is a set of element in B but not in A. The Difference is performed using (—) operator. Same can be accomplished using the function difference().




```

A={1,2,3,4,5}
B={5,7,4,9,10}
#C=A-B
C=A.difference(B)
print("set A:",A)
print("set B:",B)
print("result of A-B:",C)
#C=B-A
C=B.difference(A)
print("result of B-A:",C)

```

set4 ×

```

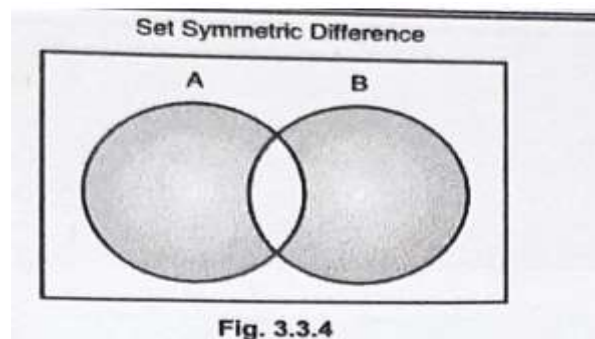
C:\Users\Swanali\PycharmProjects\
set A: {1, 2, 3, 4, 5}
set B: {4, 5, 7, 9, 10}
result of A-B: {1, 2, 3}
result of B-A: {9, 10, 7}

```

3.3.4.7 Set Symmetric_Difference

Symmetric Difference of A and B is a set of elements in both A and B except those that are common in both.

The Symmetric difference is performed using (^) operator. Same can be accomplished using the function symmetric_difference().



```

A={1,2,3,4,5}
B={5,7,4,9,10}
#C=A^B
C=A.symmetric_difference(B)
print("set A:",A)
print("set B:",B)
print("result of A^B:",C)

```

set4 ×

```

C:\Users\Swanali\PycharmProjects\python
set A: {1, 2, 3, 4, 5}
set B: {4, 5, 7, 9, 10}
result of A^B: {1, 2, 3, 7, 9, 10}

```

3.3.5 Additional Set Functions

Sr. No.	Functions
1	add() Adds an element to the set
2	clear() Removes all the elements from the set
3	copy() Returns a copy of the set
4	difference() Returns a set containing the difference between two or more sets
5	difference_update() Removes the items in this set that are also included in another, specified set
6	discard() Remove the specified item
7	intersection() Returns a set, that is the intersection of two other sets
8	intersection_update() Removes the items in this set that are not present in other, specified set(s)
9	isdisjoint() Returns whether two sets have a intersection or not
10	issubset() Returns whether another set contains this set or not

Sr. No.	Functions
11	issuperset() Returns whether this set contains another set or not
12	pop() Removes an element from the set
13	remove() Removes the specified element
14	symmetric_difference() Returns a set with the symmetric differences of two sets
15	symmetric_difference_update() inserts the symmetric differences from this set and another
16	union() Return a set containing the union of sets
17	update() Update the set with the union of this set and others

```

set1=set()
set2=set()
for i in range(1,6):
    set1.add(i)
for i in range(3,8):
    set2.add(i)
print("Set1 contains",set1)
print("Set2 contains",set2)
set3=set1|set2
print("Union of set A and set B is=",set3)
set4 =set1&set2
print("Intersection of set A and set B is=",set4)
print("set3 is superset of set4:",set3>set4)
print("set4 is superset of set3:",set4>set3)
print("set3 is subset of set4:",set3<set4)
print("set4 is subset of set3:",set4<set3)
print("set3 is same as set4:",set3==set4)
set5=set3-set4
print("Result of set3-set4:",set5)
print("set4 is disjoint of set3:",set4.isdisjoint(set5))

```

set4 ×

```

Set1 contains {1, 2, 3, 4, 5}
Set2 contains {3, 4, 5, 6, 7}
Union of set A and set B is= {1, 2, 3, 4, 5, 6, 7}
Intersection of set A and set B is= {3, 4, 5}
set3 is superset of set4: True
set4 is superset of set3: False
set3 is subset of set4: False
set4 is subset of set3: True
set3 is same as set4: False
Result of set3-set4: {1, 2, 6, 7}
set4 is disjoint of set3: True

```

3.3.6 Deleting set

Similar to Lists and Tuples, we can also delete an entire set by using keyword 'del' followed by set to be deleted.

```

set1 = {10,20,30}
print(" Initially, set1 conatains :", set1)
del set1
print("After delete operation, set1 conatains :",set1)

```

```

Traceback (most recent call last):
<class 'set'>
File "C:/Users/Majithia/PycharmProjects/MSBTE_Py_Book_All_Progs_2/test17.py", line 4, in <module>
7 print("new ",set1)
NameError: name 'set1' is not defined

```

3.4 Dictionaries

Python dictionary is an unordered collection of items. While other compound data types have only value as an element; whereas, a dictionary has a key: value (or index: element) pair. Remember that, a Dictionary is optimized to retrieve values when the key (or index) is known.

3.4.1 Creating Dictionary OR Declaring Dictionary OR Defining Dictionary

Creating a dictionary is very simple. We have to place key: value pair inside curly braces ({ }) separated by comma.

These values can be of any datatype and can repeat, but the keys must be of immutable type (String, Number or Tuple with immutable elements) and must be unique. For example,

```
dict1 = { } # creates an empty dictionary
dict2 = { 1: "Applet", 2: "Watermelon" } # dictionary with integer keys
dict3 = { "name": "Ravi", 1: [2, 4, 3] } # dictionary with mixed keys
dict4 = dict( [(1, "Apple"), (2, "Ball")] ) # from sequence having each item as a pair
```

3.4.2 Accessing Dictionary Elements using Index Numbers

Elements/values of a Dictionary are accessible only when keys/indexes are known. Similar elements to List and Tuple, elements of Dictionary are accessible by referring its index number inside index operator([]).

```
dict1 = {"Ravi": "27/09/1984", "Chandrika": "27/10/1985", "Jaineel": "23/10/2011"}
print("At index 'Ravi', element is :", dict1["Ravi"])
print("At index 'Chandrika', element is :", dict1["Chandrika"])
print("At index 'Jaineel', element is :", dict1["Jaineel"])
```

This will result as follows :

```
At index 'Ravi', element is : 27/09/1984
At index 'Chandrika', element is : 27/10/1985
At index 'Jaineel', element is : 23/10/2011
```

Remember that, the keys are case sensitive.

```
dict1 = {"Ravi": "27/09/1984", "Chandrika": "27/10/1985", "Jaineel": "23/10/2011"}
print("At index 'Ravi', element is :", dict1["ravi"])
```

This will show following error:

```
Traceback (most recent call last):
  File "C:/Users/Majithia/PycharmProjects/MSBTE_Py_Book_All_Progs_2/test16.py", line 2, in <module>
    print("At index 'Ravi', element is :", dict1["ravi"])
KeyError: 'ravi'
```


3.4.3 Iterating through dictionary using for-loop

```
dict1={"Ravi": "27/09/1984", "Chandrika": "27/10/1985", "Jaineel":  
"23/10/2011" }  
for i in dict1:  
    print(i)
```

The output of above code will be :

```
Ravi  
Chandrika  
Jaineel
```

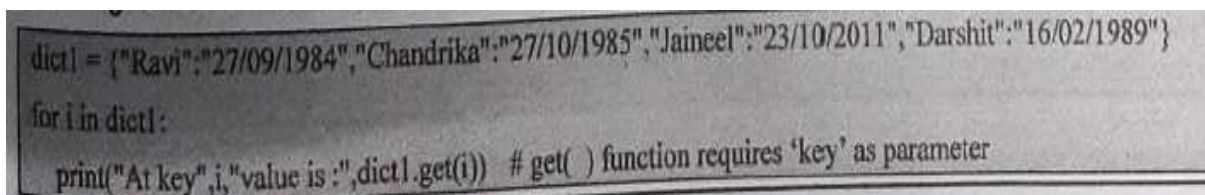
when a for-loop iterates for a Dictionary then it iterates through the keys (indexes) of Dictionary i.e. by-default it refers keys (indexes) of Dictionary and we have to use the retrieved key (index) to obtain value (element) associated with that key (index). Here is an example for that.

```
dict1={"Ravi": "27/09/1984", "Chandrika": "27/10/1985", "Jaineel":  
"23/10/2011" }  
for i in dict1:  
    print("At index",i, "value is",dict[i])
```

Here, the for-loop is iterating through keys and therefore 'i' refers to keys of dict1. And we are using that key (index) to obtain the value associated with that key. Hence, the output will be :

```
At key Ravi value is : 27/09/1984  
At key Chandrika value is : 27/10/1985  
At key Jaineel value is : 23/10/2011
```

This means, we can use indexing operator ([]) for Dictionary also, if we know the key (index). Another way of obtaining value from Dictionary is by using get() function.



```
dict1 = {"Ravi": "27/09/1984", "Chandrika": "27/10/1985", "Jaineel": "23/10/2011", "Darshit": "16/02/1989"}  
for i in dict1:  
    print("At key",i,"value is :",dict1.get(i)) # get( ) function requires 'key' as parameter
```

3.4.4 The keys() and values() Functions

Dictionary provides two built-in functions keys() and values(). The keys() function returns the keys; whereas the values() function return values.

```
dict1={"Ravi": "27/09/1984", "Chandrika": "27/10/1985", "Jaineel":  
"23/10/2011" }  
print("The keys() function returns:",dict1.keys())
```

```
print("The values() function returns:",dict1.values())
```

Now if you want to Iterate through a Dictionary even if you don't know its keys, then you can use following code.

```
dict1={"Swapnali":"7/2/1995","Shruti":"17/05/1995","Yogita":"10/06/1993"}
print("The keys() function will return:",dict1.keys())
print("The values() function will return:",dict1.values())

for i in dict1.keys():
    print("At key",i,"value is",dict1[i])
```

```
operators ×
C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/Swanali/
The keys() function will return: dict_keys(['Swapnali', 'Shruti', 'Yogita'])
The values() function will return: dict_values(['7/2/1995', '17/05/1995', '10/06/1993'])
At key Swapnali value is 7/2/1995
At key Shruti value is 17/05/1995
At key Yogita value is 10/06/1993
```

3.4.5 Basic Dictionary Operations

3.4.5.1 Change or Update Elements in Dictionary : update() Function

Dictionary allows to change or update and add an key:value pair, by using update() function. If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

```
dict1={"Swapnali":"7/2/1995","Shruti":"17/05/1995","Yogita":"10/06/1993"}
print("Initially dictionary contains:",dict1)
dict1.update({"Mina":"14/03/1990"})
print("After performing update operation:",dict1)
```

```
operators ×
C:\Users\Swanali\PycharmProjects\python_project\venv\Scripts\python.exe C:/Users/Swanali/PycharmProjects/first.py/operators.py
Initially dictionary contains: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993'}
After performing update operation: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993', 'Mina': '14/03/1990'}
```

To add new key:value pair in Dictionary, we have update() function and also assignment operation (=). Strictly remember that, to update a dictionary there must be a pair of key:value placed in curly braces { }.

We can also add an entire Dictionary to another Dictionary by using update() function.

```
dict1={"Swapnali":"7/2/1995","Shruti":"17/05/1995","Yogita":
"10/06/1993"}
print("Initially dictionary contains:",dict1)
dict1.update({"Mina":"14/03/1990"})
print("After performing update operation:",dict1)
dict2={"Ram":"1/5/1990","Shyam":"2/6/1992"}
dict1.update(dict2)
print("After performing update operation:",dict1)
```

Output :

Initially dictionary contains: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993'}
 After performing update operation: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993', 'Mina': '14/03/1990'}
 After performing update operation: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993', 'Mina': '14/03/1990', 'Ram': '1/5/1990', 'Shyam': '2/6/1992'}

3.4.5.2 Finding Length of List : using `__len__ ()` Function

The `__len__ ()` is a built-in function of Dictionary that returns numbers of elements/pairs in Dictionary.

```
dict1={"Swapnali":"7/2/1995","Shruti":"17/05/1995","Yogita":
"10/06/1993"}
a=dict1.__len__()
print("dict1 contains",a,"pairs")
```

Output:

dict1 contains 3 pairs

Remember that a Dictionary can never be iterated by using standard index numbers that starts from [0] to [length— 1]. We must use specific key (index) for Dictionary. Hence, following code is not valid.

```
dict1 = {"Ravi":"27/09/1984", "Chandrika":"27/10/1985", "Jaineel":"23/10/2011"}
i=0
while i<dict1.__len__():
    print("At index",i,"element value is :",dict1[i])
    i=i+1
```

```
Traceback (most recent call last):
  File "C:/Users/Majithia/PycharmProjects/MSBTE_Py_Book_All_Progs_2/test24.py", line 4, in <module>
    print("At index",i,"element value is :",dict1[i])
  KeyError: 0
Process finished with exit code 1
```


3.4.5.3 Updating Dictionary : Using Assignment Operator (s) to over-write Value

A Dictionary is mutable entity. We can add new items or change the value of existing items using assignment operator.

```
dict1={"Swapnali":"7/2/1995","Shruti":"17/05/1995","Yogita":  
"10/06/1993"}  
print("Initially dictionary contains:",dict1)  
dict1["Yogita"]="12/06/1993"  
print("After updating dictionary:",dict1)
```

Output:

Initially dictionary contains: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993'}

After updating dictionary: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '12/06/1993'}

Remember one thing if any invalid key is provided then also Dictionary will show any IndexError, as it was the case in List and Tuple.

```
dict1={"Swapnali":"7/2/1995","Shruti":"17/05/1995","Yogita":  
"10/06/1993"}  
print("Initially dictionary contains:",dict1)  
dict1["Yogita"]="12/06/1993"  
print("After updating dictionary:",dict1)  
dict1["Ankita"]="18/6/1993"    #ankita is not present in  
dict1  
print("After updating Ankita in dictionary:",dict1)
```

Note the assignment operation at key 'Ankita' which is actually not available in dict1. Then also it will not show any error; but instead it will add it as new key: value pair in dict1. Here is the output of above code.

Initially dictionary contains: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993'}

After updating dictionary: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '12/06/1993'}

After updating Ankita in dictionary: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '12/06/1993', 'Ankita': '18/6/1993'}

3.4.5.4 Remove Element : pop(), popitem() and clear() Functions

Dictionary allows to remove element by using four ways.

1. using keyword 'del' with specified key
2. using pop() function
3. using popitem() function
4. using clear() function

Keyword 'del' will remove key-value pair of specified key. If any invalid or unavailable key is provide, then it will show 'KeyError'.

The pop() function will remove key-value pair of specified key.

The popitem() function will remove arbitrary key:value pair.

The clear() function will clear all elements from Dictionary.

```
dict1={"Swapnali":"7/2/1995","Shruti":"17/05/1995","Yogita":"10/06/1993","Ankita":"23/4/1992"}
print("Initially dictionary contains:",dict1)
del dict1["Yogita"]
print("After deleting 'Yogita' dictionary contains:",dict1)
dict1.pop("Shruti")
print("After deleting 'Shruti' dictionary contains:",dict1)
dict1.popitem()
print("After performing popitem() method, dictionary contains:",dict1)
dict1.clear()
print("After performing clear() method, dictionary contains:",dict1)
```

Output:

Initially dictionary contains: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Yogita': '10/06/1993', 'Ankita': '23/4/1992'}

After deleting 'Yogita' dictionary contains: {'Swapnali': '7/2/1995', 'Shruti': '17/05/1995', 'Ankita': '23/4/1992'}

After deleting 'Shruti' dictionary contains: {'Swapnali': '7/2/1995', 'Ankita': '23/4/1992'}

After performing popitem() method, dictionary contains: {'Swapnali': '7/2/1995'}

After performing clear() method, dictionary contains: {}

3.4.6 Built-in Dictionary Functions

Sr. No.	Functions
1	clear() - clear() Removes all the elements from the Dictionary
2	copy() - copy() Returns a copy of the Dictionary
3	fromkeys() - fromkeys(sequence) Returns a Dictionary with the specified keys and values
4	get() - get(key) Returns the value of the specified key
5	items() - items() Returns a list containing a Tuple for each key value pair
6	keys() - keys() Returns a list containing the Dictionary's keys

	Functions
7	pop() - pop(key) Removes the element with the specified key
8	popitem() - popitem() Removes the last inserted key-value pair
9	setdefault() - setdefault(key,default_value) Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
10	update() - update(dictionary) Updates the Dictionary with the specified key-value pairs
11	values() - values() Returns a list of all the values in the Dictionary

3.4.7 Deleting Dictionary

The keyword 'del' can also delete an entire Dictionary, as we used it for Lists, Tuples and Sets.

```
dict1 = {"Ravi" : "27/09/1984", "Chandrika" : "27/10/1985", "Jaineel" : "23/10/2011"}
print("Initially, Dictionary contains :", dict1)
del dict1
print("After delete operation, Dictionary contains :", dict1)
```