Unit III

CHAPTER 9

LISTS, TUPLES, SETS AND DICTIONARY

Learning Objectives

After studying this chapter, students will be able to:

- Understand the basic concepts of various collection data types in python such as List, Tuples, sets and Dictionary.
- Work with List, Tuples, sets and Dictionaries using variety of functions.
- Writting Python programs using List, Tuples, sets and Dictionaries.
- Understand the relationship between List, Tuples and Dictionaries.

9.1 Introduction to List

Python programming language has four collections of data types such as List, Tuples, Set and Dictionary. A list in Python is known as a **"sequence data type"** like strings. It is an ordered collection of values enclosed within square brackets [ ]. Each value of a list is called as element. It can be of any type such as numbers, characters, strings and even the nested lists as well. The elements can be modified or mutable which means the elements can be replaced, added or removed. Every element rests at some position in the list. The position of an element is indexed with numbers beginning with zero which is used to locate and access a particular element. Thus, lists are similar to arrays, what you learnt in XI std.

9.1.1 Create a List in Python

In python, a list is simply created by using square bracket. The elements of list should be specified within square brackets. The following syntax explains the creation of list.

Syntax:

Variable = [element-1, element-2, element-3 …… element-n]

Example

```
Marks = [10, 23, 41, 75]
Fruits = ["Apple", "Orange", "Mango", "Banana"]
MyList = [ ]
```

In the above example, the list Marks has four integer elements; second list Fruits has four string elements; third is an empty list. The elements of a list need not be homogenous type of data. The following list contains multiple type elements.

Mylist = ["Welcome", 3.14, 10, [2, 4, 6]]

In the above example, Mylist contains another list as an element. This type of list is known as **"Nested List".**

Nested list is a list containing another list as an element.

9.1.2 Accessing List elements

Python assigns an automatic index value for each element of a list begins with zero. Index value can be used to access an element in a list. In python, index value is an integer number which can be positive or negative.

Example

Marks = [10, 23, 41, 75]

Marks	10	23	41	75
Index (Positive)	0	1	2	3
IndexNegative)	-4	-3	-2	-1

Positive value of index counts from the beginning of the list and negative value means counting backward from end of the list (i.e. in reverse order).

To access an element from a list, write the name of the list, followed by the index of the element enclosed within square brackets.

Syntax:

List_Variable = [E1, E2, E3 …… En]

print (List_Variable[index of a element])

Example (Accessing single element):

```
>>> Marks = [10, 23, 41, 75]
>>> print (Marks[0])
10
```

In the above example, print command prints 10 as output, as the index of 10 is zero.

Example: Accessing elements in revevrse order

```
>>> Marks = [10, 23, 41, 75]
>>> print (Marks[-1])
75
```

Note

A negative index can be used to access an element in reverse order.

(i) Accessing all elements of a list

Loops are used to access all elements from a list. The initial value of the loop must be zero. Zero is the beginning index value of a list.

Example

```
Marks = [10, 23, 41, 75]
i = 0
while i < 4:
        print (Marks[i])
        i = i + 1
```

Output

```
        10
        23
        41
        75
```

In the above example, Marks list contains four integer elements i.e., 10, 23, 41, 75. Each element has an index value from 0. The index value of the elements are 0, 1, 2, 3 respectively. Here, the while loop is used to read all the elements. The initial value of the loop is zero, and the test condition is i < 4, as long as the test condition is true, the loop executes and prints the corresponding output.

During the first iteration, the value of **i** is **zero**, where the condition is true. Now, the following statement **print (Marks [i])** gets executed and prints the value of Marks [0] element ie. 10.

The next statement **i = i + 1** increments the value of **i** from **0** to **1**. Now, the flow of control shifts to the while statement for checking the test condition. The process repeats to print the remaining elements of **Marks** list until the test condition of while loop becomes false.

The following table shows that the execution of loop and the value to be print.

Iteration	i	while i < 4	print (Marks[i])	i = i + 1
1	0	0 < 4 True	Marks [0] = 10	0 + 1 = 1
2	1	1 < 4 True	Marks [1] = 23	1 + 1 = 2
3	2	2 < 4 True	Marks [2] = 41	2 + 1 = 3
4	3	3 < 4 True	Marks [3] = 75	3 + 1 = 4
5	4	4 < 4 False	--	--

(ii) Reverse Indexing

Python enables reverse or negative indexing for the list elements. Thus, python lists index in opposite order. The python sets -1 as the index value for the last element in list and -2 for the preceding element and so on. This is called as **Reverse Indexing.**

Example

```
Marks = [10, 23, 41, 75]
i = -1
while i >= -4:
        print (Marks[i])
        i = i + -1
```

Output

```
        75
        41
        23
        10
```

The following table shows the working process of the above python coding

Iteration	i	while i >= -4	print (Marks[i])	i = i + -1
1	-1	-1 >= -4 True	Marks[-1] = 75	-1 + (-1) = -2
2	-2	-2 >= -4 True	Marks[-2] = 41	-2 + (-1) = -3
3	-3	-3 >= -4 True	Marks[-3] = 23	-3 + (-1) = -4
4	-4	-4 >= -4 True	Marks[-4] = 10	-4 + (-1) = -5
5	-5	-5 >= -4 False	--	--

9.1.3 List Length

The len( ) function in Python is used to find the length of a list. (i.e., the number of elements in a list). Usually, the len( ) function is used to set the upper limit in a loop to read all the elements of a list. If a list contains another list as an element, len( ) returns that inner list as a single element.

Example :Accessing single element

```
>>> MySubject = [“Tamil”, “English”, “Comp. Science”, “Maths”]
>>> len(MySubject)
    4
```

Example : Program to display elements in a list using loop

```
MySubject = ["Tamil", "English", "Comp. Science", "Maths"]
i = 0
while i < len(MySubject):
        print (MySubject[i])
        i = i + 1
```

Output

```
        Tamil
        English
        Comp. Science
        Maths
```

9.1.4 Accessing elements using for loop

In Python, the ***for*** loop is used to access all the elements in a list one by one. This is just like the ***for*** keyword in other programming language such as C++.

Syntax:

```
for index_var in list:
    print (index_var)
```

Here, ***index_var*** represents the index value of each element in the list. Python reads this "for" statement like English: *"For (every) element in (the list of) list and print (the name of the) list items"*

Example

```
Marks=[23, 45, 67, 78, 98]
for x in Marks:
          print( x )
```

Output

```
          23
          45
          67
          78
          98
```

In the above example, Marks list has 5 elements; each element is indexed from 0 to 4. The Python reads the ***for*** loop and ***print*** statements like English: *"For (every) element (represented as x) in (the list of) Marks and print (the values of the) elements"*.

9.1.5 Changing list elements

In Python, the lists are mutable, which means they can be changed. A list element or range of elements can be changed or altered by using simple assignment operator (=).

Syntax:

List_Variable [index of an element] = Value to be changed
List_Variable [index from : index to] = Values to changed

Where, ***index from*** is the beginning index of the range; ***index to*** is the upper limit of the range which is excluded in the range. For example, if you set the range [0:5] means, Python takes only 0 to 4 as element index. Thus, if you want to update the range of elements from 1 to 4, it should be specified as [1:5].

Example 9.1: Python program to update/change single value

```
MyList = [2, 4, 5, 8, 10]
print ("MyList elements before update... ")
for x in MyList:
    print (x)
MyList[2] = 6
print ("MyList elements after updation... ")
for y in MyList:
    print (y)
```

Output:

```
MyList elements before update...
            2
            4
            5
            8
            10
MyList elements after updation...
            2
            4
            6
            8
            10
```

Example 9.2: Python program to update/change range of values

```
MyList = [1, 3, 5, 7, 9]
print ("List Odd numbers... ")
for x in MyList:
    print (x)
MyList[0:5] = 2,4,6,8,10
print ("List Even numbers... ")
for y in MyList:
    print (y)
```

Output

```
List Odd numbers...
            1
            3
            5
            7
            9
List Even numbers...
            2
            4
            6
            8
            10
```

12th Computer Science_EM Chapter 9.indd 138 22-12-2021 16:45:10

9.1.6 Adding more elements in a list

In Python, append( ) function is used to add a single element and extend( ) function is used to add more than one element to an existing list.

Syntax:

List.append (element to be added)
List.extend ([elements to be added])

In extend( ) function, multiple elements should be specified within square bracket as arguments of the function.

Example

```
>>> Mylist=[34, 45, 48]
>>> Mylist.append(90)
>>> print(Mylist)
        [34, 45, 48, 90]
```

In the above example, Mylist is created with three elements. Through >>> **Mylist.append(90)** statement, an additional value 90 is included with the existing list as last element, following print statement shows all the elements within the list MyList.

Example

```
>>> Mylist.extend([71, 32, 29])
>>> print(Mylist)
        [34, 45, 48, 90, 71, 32, 29]
```

In the above code, extend( ) function is used to include multiple elements, the print statement shows all the elements of the list after the inclusion of additional elements.

9.1.7 Inserting elements in a list

As you learnt already, append( ) function in Python is used to add more elements in a list. But, it includes elements at the end of a list. If you want to include an element at your desired position, you can use insert ( ) function. The insert( ) function is used to insert an element at any position of a list.

Syntax:

List.insert (position index, element)

Example

```
>>> MyList=[34,98,47,'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan' ]
>>> print(MyList)
        [34, 98, 47, 'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan']
>>> MyList.insert(3, 'Ramakrishnan')
>>> print(MyList)
[34, 98, 47, 'Ramakrishnan', 'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan']
```

In the above example, insert( ) function inserts a new element ‘Ramakrishnan’ at the index value 3, ie. at the 4th position. While inserting a new element in between the existing elements, at a particular location, the existing elements shifts one position to the right.

9.1.8 Deleting elements from a list

There are two ways to delete an element from a list viz. **del** statement and **remove( )** function. del statement is used to delete elements whose index is known whereas remove( ) function is used to delete elements of a list if its index is unknown. The del statement can also be used to delete entire list.

Syntax:

del List [index of an element]

# to delete a particular element

del List [index from : index to]

# to delete multiple elements

del List

# to delete entire list

Example

```
>>> MySubjects = ['Tamil', 'Hindi', 'Telugu', 'Maths']
>>> print (MySubjects)
        ['Tamil', 'Hindi', 'Telugu', 'Maths']
>>> del MySubjects[1]
>>> print (MySubjects)
        ['Tamil', 'Telugu', 'Maths']
```

In the above example, the list **MySubjects** has been created with four elements. print statement shows all the elements of the list. In >>> **del MySubjects[1]** statement, deletes an element whose index value is 1 and the following print shows the remaining elements of the list.

Example

```
>>> del MySubjects[1:3]
>>> print(MySubjects)
        ['Tamil']
```

In the above codes, >>> **del MySubjects[1:3]** deletes the second and third elements from the list. The upper limit of index is specified within square brackets, will be taken as -1 by the python.

12th Computer Science_EM Chapter 9.indd 140 22-12-2021 16:45:10

Example

```
>>> del MySubjects
>>> print(MySubjects)
Traceback (most recent call last):
File "<pyshell#9>", line 1, in <module>
print(MySubjects)
NameError: name 'MySubjects' is not defined
```

Here, >>> del MySubjects, deletes the list MySubjects entirely. When you try to print the elements, Python shows an error as the list is not defined. Which means, the list MySubjects has been completely deleted.

As already stated, the remove( ) function can also be used to delete one or more elements if the index value is not known. Apart from remove( ) function, pop( ) function can also be used to delete an element using the given index value. pop( ) function deletes and returns the last element of a list if the index is not given.

The function clear( ) is used to delete all the elements in list, it deletes only the elements and retains the list. Remember that, the del statement deletes entire list.

Syntax:

```
List.remove(element)          # to delete a particular element
List.pop(index of an element)
List.clear( )
```

Example

```
>>> MyList=[12,89,34,'Kannan', 'Gowrisankar', 'Lenin']
>>> print(MyList)
        [12, 89, 34, 'Kannan', 'Gowrisankar', 'Lenin']
>>> MyList.remove(89)
>>> print(MyList)
        [12, 34, 'Kannan', 'Gowrisankar', 'Lenin']
```

In the above example, MyList has been created with three integer and three string elements, the following print statement shows all the elements available in the list. In the statement >>> **MyList.remove(89)**, deletes the element 89 from the list and the print statement shows the remaining elements.

Example

```
>>> MyList.pop(1)
      34
>>> print(MyList)
      [12, 'Kannan', 'Gowrisankar', 'Lenin']
```

In the above code, pop( ) function is used to delete a particular element using its index value, as soon as the element is deleted, the pop( ) function shows the element which is deleted. pop( ) function is used to delete only one element from a list. Remember that, del statement deletes multiple elements.

Example

```
>>> MyList.clear( )
>>> print(MyList)
            [ ]
```

In the above code, clear( ) function removes only the elements and retains the list. When you try to print the list which is already cleared, an empty square bracket is displayed without any elements, which means the list is empty.

9.1.9 List and range ( ) function

The range( ) is a function used to generate a series of values in Python. Using range( ) function, you can create list with series of values. The range( ) function has three arguments.

Syntax of range ( ) function:

range (start value, end value, step value)

where,

- **start value** – beginning value of series. Zero is the default beginning value.
- **end value** – upper limit of series. Python takes the ending value as upper limit – 1.
- **step value** – It is an optional argument, which is used to generate different interval of values.

12th Computer Science_EM Chapter 9.indd 142 22-12-2021 16:45:10

Example : Generating whole numbers upto 10

```
for x in range (1, 11):
        print(x)
```

Output

```
1
2
3
4
5
6
7
8
9
10
```

Example : Generating first 10 even numbers

```
for x in range (2, 11, 2):
        print(x)
```

Output

```
2
4
6
8
10
```

(i) Creating a list with series of values

Using the range( ) function, you can create a list with series of values. To convert the result of range( ) function into list, we need one more function called list( ). The list( )

function makes the result of range( ) as a list.

Syntax:

List_Varibale = list (range ( ))

Note

The list ( ) function is also used to create list in python.

Example

```
>>> Even_List = list(range(2,11,2))
>>> print(Even_List)
[2, 4, 6, 8, 10]
```

In the above code, list( ) function takes the result of range( ) as Even_List elements. Thus, Even_List list has the elements of first five even numbers.

Similarly, we can create any series of values using range( ) function. The following example explains how to create a list with squares of first 10 natural numbers.

Example : Generating squares of first 10 natural numbers

```
squares = [ ]
for x in range(1,11):
        s = x ** 2
        squares.append(s)
print (squares)
```

In the above program, an empty list is created named "squares". Then, the for loop generates natural numbers from 1 to 10 using range( ) function. Inside the loop, the current value of x is raised to the power 2 and stored in the variables. Each new value of square is appended to the list "squares". Finally, the program shows the following values as output.

Output

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

9.1.10 List comprehensions

List comprehension is a simplest way of creating sequence of elements that satisfy a certain condition.

Syntax:

List = [expression for variable in range]

Example : Generating squares of first 10 natural numbers using the concept of List comprehension

```
>>> squares = [ x ** 2 for x in range(1,11) ]
>>> print (squares)
```

Output:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In the above example, x ** 2 in the expression is evaluated each time it is iterated. This is the shortcut method of generating series of values.

9.1.11 Other important list funcion

Function	Description	Syntax	Example

copy ( )	Returns a copy of the list	List.copy( )	MyList=[12, 12, 36]
			x = MyList.copy()
			print(x)
			Output:
			[12, 12, 36]

count ( )	Returns the number of similar elements present in the last.	List.count(value)	MyList=[36 ,12 ,12]
			x = MyList.count(12)
			print(x)
			Output:
			2

index ( )	Returns the index value of the first recurring element	List.index(element)	MyList=[36 ,12 ,12]
			x = MyList.index(12)
			print(x)
			Output:
			1

reverse ( )	Reverses the order of the element in the list.	List.reverse( )	MyList=[36 ,23 ,12]
			MyList.reverse()
			print(MyList)
			Output:
			[12 ,23 ,36]

sort ( )	Sorts the element in list	List.sort(reverse=True|False, key=myFunc)	

Both arguments are optional
• If reverse is set as True, list sorting is in descending order.
• Ascending is default.
• Key=myFunc; “myFunc” - the name of the user defined function that specifies the sorting criteria.
Note: sort( ) will affect the original list.
		MyList=['Thilothamma', 'Tharani', 'Anitha', 'SaiSree', 'Lavanya']
		MyList.sort( )
		print(MyList)
		MyList.sort(reverse=True)
		print(MyList)
		Output:
		['Anitha', 'Lavanya', 'SaiSree', 'Tharani', 'Thilothamma']
		['Thilothamma', 'Tharani', 'SaiSree', 'Lavanya', 'Anitha']	

max( )	Returns the maximum value in a list.	max(list)	MyList=[21,76,98,23]
			print(max(MyList))
			Output:
			98

min( )	Returns the minimum value in a list.	min(list)	MyList=[21,76,98,23]
			print(min(MyList))
			Output:
			21

sum( )	Returns the sum of values in a list.	sum(list)	MyList=[21,76,98,23]
			print(sum(MyList))
			Output:
			218

9.1.12 Programs using List

Program 1: write a program that creates a list of numbers from 1 to 20 that are divisible by 4

```
divBy4=[ ]
for i in range(21):
        if (i%4==0):
                divBy4.append(i)
print(divBy4)
```

Output

```
      [0, 4, 8, 12, 16, 20]
```

Program 2: Write a program to define a list of countries that are a member of BRICS. Check whether a county is member of BRICS or not

```
country=["India", "Russia", "Srilanka", "China", "Brazil"]
is_member = input("Enter the name of the country: ")
if is_member in country:
        print(is_member, " is the member of BRICS")
else:
        print(is_member, " is not a member of BRICS")
```

Output

```
      Enter the name of the country: India
      India  is the member of BRICS
```

Output

```
      Enter the name of the country: Japan
      Japan  is not a member of BRICS
```

Program 3: Python program to read marks of six subjects and to print the marks scored in each subject and show the total marks

```
marks=[]
subjects=['Tamil', 'English', 'Physics', 'Chemistry', 'Comp. Science', 'Maths']
for i in range(6):
        m=int(input("Enter Mark = "))
        marks.append(m)
for j in range(len(marks)):
        print("{ }. { } Mark = { } ".format(j+1,subjects[j],marks[j]))
print("Total Marks = ", sum(marks))
```

Output

```
      Enter Mark = 45
      Enter Mark = 98
      Enter Mark = 76
      Enter Mark = 28
      Enter Mark = 46
      Enter Mark = 15
        1. Tamil Mark = 45
        2. English Mark = 98
        3. Physics Mark = 76
        4. Chemistry Mark = 28
        5. Comp. Science Mark = 46
        6. Maths Mark = 15
        Total Marks =  308
```

Program 4: Python program to read prices of 5 items in a list and then display sum of all the prices, product of all the prices and find the average

```
items=[]
prod=1
for i in range(5):
      print ("Enter price for item { } : ".format(i+1))
      p=int(input())
      items.append(p)
for j in range(len(items)):
      print("Price for item { } = Rs. { }".format(j+1,items[j]))
      prod = prod * items[j]
print("Sum of all prices = Rs.", sum(items))
print("Product of all prices = Rs.", prod)
print("Average of all prices = Rs.",sum(items)/len(items))
```

Output:

```
Enter price for item  1 :
5
Enter price for item  2 :
10
Enter price for item  3 :
15
Enter price for item  4 :
20
Enter price for item  5 :
25
Price for item  1 = Rs. 5
Price for item  2 = Rs. 10
Price for item  3 = Rs. 15
Price for item  4 = Rs. 20
Price for item  5 = Rs. 25
Sum of all prices = Rs. 75
Product of all prices = Rs. 375000
Average of all prices = Rs. 15.0
```

Program 5: Python program to count the number of employees earning more than 1 lakh per annum. The monthly salaries of n number of employees are given

```
count=0
n=int(input("Enter no. of employees: "))
print("No. of Employees",n)
salary=[]
for i in range(n):
        print("Enter Monthly Salary of Employee { } Rs.: ".format(i+1))
        s=int(input())
        salary.append(s)
for j in range(len(salary)):
        annual_salary = salary[j] * 12
        print ("Annual Salary of Employee { } is:Rs. { }".format(j+1,annual_salary))
        if annual_salary >= 100000:
                count = count + 1
print("{ } Employees out of { } employees are earning more than Rs. 1 Lakh per annum".
format(count, n))
```

Output:

```
Enter no. of employees: 5
No. of Employees 5
Enter Monthly Salary of Employee 1 Rs.:
3000
Enter Monthly Salary of Employee 2 Rs.:
9500
Enter Monthly Salary of Employee 3 Rs.:
12500
Enter Monthly Salary of Employee 4 Rs.:
5750
Enter Monthly Salary of Employee 5 Rs.:
8000
Annual Salary of Employee 1 is:Rs. 36000
Annual Salary of Employee 2 is:Rs. 114000
Annual Salary of Employee 3 is:Rs. 150000
Annual Salary of Employee 4 is:Rs. 69000
Annual Salary of Employee 5 is:Rs. 96000
2 Employees out of 5 employees are earning more than Rs. 1 Lakh per annum
```

Program 6: Write a program to create a list of numbers in the range 1 to 10. Then delete all the even numbers from the list and print the final list.

```
Num = []
for x in range(1,11):
    Num.append(x)
print("The list of numbers from 1 to 10 = ", Num)

for index, i in enumerate(Num):
    if(i%2==0):
        del Num[index]
print("The list after deleting even numbers = ", Num)
```

Output

```
The list of numbers from 1 to 10 =  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The list after deleting even numbers =  [1, 3, 5, 7, 9]
```

12th Computer Science_EM Chapter 9.indd 149 22-12-2021 16:45:10

Program 7: Write a program to generate in the Fibonacci series and store it in a list. Then find the sum of all values.

```
a=-1
b=1
n=int(input("Enter no. of terms: "))
i=0
sum=0
Fibo=[]
while i<n:
        s = a + b
        Fibo.append(s)
        sum+=s
        a = b
        b = s
        i+=1
print("Fibonacci series upto "+ str(n) +" terms is : " + str(Fibo))
print("The sum of Fibonacci series: ",sum)
```

Output

```
        Enter no. of terms: 10
        Fibonacci series upto 10 terms is : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
        The sum of Fibonacci series:  88
```

9.2 Tuples

Introduction to Tuples

Tuples consists of a number of values separated by comma and enclosed within parentheses. Tuple is similar to list, values in a list can be changed but not in a tuple.

DO YOU KNOW?

The term Tuple is originated from the Latin word represents an abstraction of the sequence of numbers:

single(1), double(2), triple(3), quadruple(4), quintuple(5), sextuple(6), septuple(7), octuple(8), ..., n-tuple, ...,

9.2.1 Advantages of Tuples over list

1. The elements of a list are changeable (mutable) whereas the elements of a tuple are unchangeable (immutable), this is the key difference between tuples and list.
2. The elements of a list are enclosed within square brackets. But, the elements of a tuple are enclosed by paranthesis.
3. Iterating tuples is faster than list.

9.2.2 Creating Tuples

Creating tuples is similar to list. In a list, elements are defined within square brackets, whereas in tuples, they may be enclosed by parenthesis. The elements of a tuple can be even defined without parenthesis. Whether the elements defined within parenthesis or without parenthesis, there is no differente in it's function.

Syntax:

```
# Empty tuple
Tuple_Name = ( )

# Tuple with n number elements
Tuple_Name = (E1, E2, E2 ....... En)

# Elements of a tuple without parenthesis
Tuple_Name = E1, E2, E3 ..... En
```

Example

```
>>> MyTup1 = (23, 56, 89, 'A', 'E', 'I', "Tamil")
>>> print(MyTup1)
        (23, 56, 89, 'A', 'E', 'I', 'Tamil')

>>> MyTup2 = 23, 56, 89, 'A', 'E', 'I', "Tamil"
>>> print (MyTup2)
        (23, 56, 89, 'A', 'E', 'I', 'Tamil')
```

(i) Creating tuples using tuple( ) function

The tuple( ) function is used to create Tuples from a list. When you create a tuple, from a list, the elements should be enclosed within square brackets.

Syntax:

```
Tuple_Name = tuple( [list elements] )
```

Example

```
>>> MyTup3 = tuple( [23, 45, 90] )
>>> print(MyTup3)
        (23, 45, 90)
>>> type (MyTup3)
        <class ‘tuple’>
```

12th Computer Science_EM Chapter 9.indd 151 22-12-2021 16:45:10

Note

Type ( ) function is used to know the data type of a python object.

(ii) Creating Single element tuple

While creating a tuple with a single element, add a comma at the end of the element. In the absence of a comma, Python will consider the element as an ordinary data type; not a tuple. Creating a Tuple with one element is called "Singleton" tuple.

Example

```
>>> MyTup4 = (10)
>>> type(MyTup4)
        <class 'int'>
>>> MyTup5 = (10,)
>>> type(MyTup5)
        <class 'tuple'>
```

9.2.3 Accessing values in a Tuple

Like list, each element of tuple has an index number starting from zero. The elements of a tuple can be easily accessed by using index number.

Example

```
>>> Tup1 = (12, 78, 91, "Tamil", "Telugu", 3.14, 69.48)
# to access all the elements of a tuple
>>> print(Tup1)
        (12, 78, 91, 'Tamil', 'Telugu', 3.14, 69.48)
#accessing selected elements using indices
>>> print(Tup1[2:5])
        (91, 'Tamil', 'Telugu')
#accessing from the first element up to the specified index value
>>> print(Tup1[:5])
        (12, 78, 91, 'Tamil', 'Telugu')
# accessing from the specified element up to the last element.
>>> print(Tup1[4:])
        ('Telugu', 3.14, 69.48)
# accessing from the first element to the last element
>>> print(Tup1[:])
        (12, 78, 91, 'Tamil', 'Telugu', 3.14, 69.48)
```

9.2.4 Update and Delete Tuple

As you know a tuple is immutable, the elements in a tuple cannot be changed. Instead of altering values in a tuple, joining two tuples or deleting the entire tuple is possible.

Example

```
# Program to join two tuples
Tup1 = (2,4,6,8,10)
Tup2 = (1,3,5,7,9)
Tup3 = Tup1 + Tup2
print(Tup3)
```

Output

```
(2, 4, 6, 8, 10, 1, 3, 5, 7, 9)
```

To delete an entire tuple, the del command can be used.

Syntax:

del tuple_name

Example

```
Tup1 = (2,4,6,8,10)
print("The elements of Tup1 is ", Tup1)
del Tup1
print (Tup1)
```

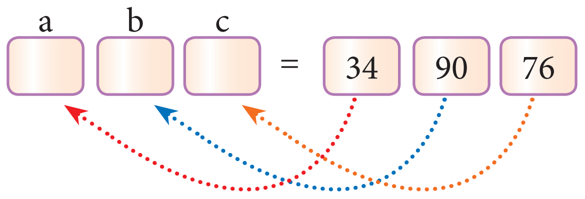
Output:

```
The elements of Tup1 is  (2, 4, 6, 8, 10)
Traceback (most recent call last):
File "D:/Python/Tuple Examp 1.py", line 4, in <module>
print (Tup1)
NameError: name 'Tup1' is not defined
```

Note that, the print statement in the above code prints the elements. Then, the del statement deletes the entire tuple. When you try to print the deleted tuple, Python shows the error.

9.2.5 Tuple Assignment

Tuple assignment is a powerful feature in Python. It allows a tuple variable on the left of the assignment operator to be assigned to the values on the right side of the assignment operator. Each value is assigned to its respective variable.



Example

```
>>> (a, b, c) = (34, 90, 76)
>>> print(a,b,c)
        34 90 76
# expression are evaluated before assignment
>>> (x, y, z, p) = (2**2, 5/3+4, 15%2, 34>65)
>>> print(x,y,z,p)
        4 5.666666666666667 1 False
```

Note that, when you assign values to a tuple, ensure that the number of values on both sides of the assignment operator are same; otherwise, an error is generated by Python.

9.2.6 Returning multiple values in Tuples

A function can return only one value at a time, but Python returns more than one value from a function. Python groups multiple values and returns them together.

Example : Program to return the maximum as well as minimum values in a list

```
def Min_Max(n):
      a = max(n)
      b = min(n)
      return(a, b)
Num = (12, 65, 84, 1, 18, 85, 99)
(Max_Num, Min_Num) = Min_Max(Num)
print("Maximum value = ", Max_Num)
print("Minimum value = ", Min_Num)
```

Output:

```
      Maximum value =  99
      Minimum value =  1
```

9.2.7 Nested Tuples

In Python, a tuple can be defined inside another tuple; called Nested tuple. In a nested tuple, each tuple is considered as an element. The for loop will be useful to access all the elements in a nested tuple.

Example

```
Toppers = (("Vinodini", "XII-F", 98.7), ("Soundarya", "XII-H", 97.5),
                ("Tharani", "XII-F", 95.3), ("Saisri", "XII-G", 93.8))
for i in Toppers:
        print(i)
```

Output:

```
('Vinodini', 'XII-F', 98.7)
('Soundarya', 'XII-H', 97.5)
('Tharani', 'XII-F', 95.3)
('Saisri', 'XII-G', 93.8)
```

Note

Some of the functions used in List can be applicable even for tuples.

9.2.8 Programs using Tuples

Program 1: Write a program to swap two values using tuple assignment

```
a = int(input("Enter value of A: "))
b = int(input("Enter value of B: "))
print("Value of A = ", a, "\n Value of B = ", b)
(a, b) = (b, a)
print("Value of A = ", a, "\n Value of B = ", b)
```

Output:

```
Enter value of A: 54
Enter value of B: 38
Value of A =  54
 Value of B =  38
Value of A =  38
 Value of B =  54
```

12th Computer Science_EM Chapter 9.indd 155 22-12-2021 16:45:10

Program 2: Write a program using a function that returns the area and circumference of a circle whose radius is passed as an argument.two values using tuple assignment

```
pi = 3.14
def Circle(r):
        return (pi*r*r, 2*pi*r)
radius = float(input("Enter the Radius: "))
(area, circum) = Circle(radius)
print ("Area of the circle = ", area)
print ("Circumference of the circle = ", circum)
```

Output:

```
        Enter the Radius: 5
        Area of the circle =  78.5
        Circumference of the circle =  31.400000000000002
```

Program 3: Write a program that has a list of positive and negative numbers. Create a new tuple that has only positive numbers from the list

```
Numbers = (5, -8, 6, 8, -4, 3, 1)
Positive = ( )
for i in Numbers:
        if i > 0:
                Positive += (i, )
print("Positive Numbers: ", Positive)
```

Output:

```
        Positive Numbers:  (5, 6, 8, 3, 1)
```

9.3 Sets

Introduction

In python, a set is another type of collection data type. A Set is a mutable and an unordered collection of elements without duplicates. That means the elements within a set cannot be repeated. This feature used to include membership testing and eliminating duplicate elements.

9.3.1 Creating a Set

A set is created by placing all the elements separated by comma within a pair of curly brackets. The set( ) function can also used to create sets in Python.

Syntax:

Set_Variable = {E1, E2, E3 …….. En}

Example

```
>>> S1={1,2,3,'A',3.14}
>>> print(S1)
        {1, 2, 3, 3.14, 'A'}

>>> S2={1,2,2,'A',3.14}
>>> print(S2)
        {1, 2, 'A', 3.14}
```

In the above examples, the set S1 is created with different types of elements without duplicate values. Whereas in the set S2 is created with duplicate values, but python accepts only one element among the duplications. Which means python removed the duplicate value, because a set in python cannot have duplicate elements.

Note

When you print the elements from a set, python shows the values in different order.

9.3.2 Creating Set using List or Tuple

A list or Tuple can be converted as set by using set( ) function. This is very simple procedure. First you have to create a list or Tuple then, substitute its variable within set( ) function as argument.

Example

```
MyList=[2,4,6,8,10]
MySet=set(MyList)
print(MySet)
```

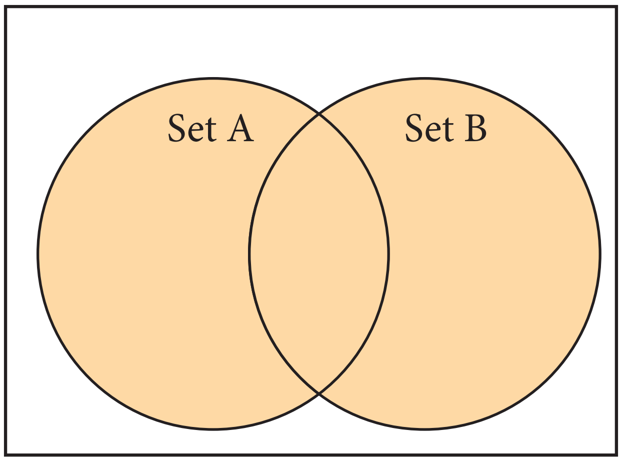
Output:

```
        {2, 4, 6, 8, 10}
```

9.3.3 Set Operations

As you learnt in mathematics, the python is also supports the set operations such as Union, Intersection, difference and Symmetric difference.

(i) Union: It includes all elements from two or more sets



In python, the operator | is used to union of two sets. The function union( ) is also used to join two sets in python.

Example: Program to Join (Union) two sets using union operator

```
set_A={2,4,6,8}
set_B={'A', 'B', 'C', 'D'}
U_set=set_A|set_B
print(U_set)
```

Output:

```
{2, 4, 6, 8, 'A', 'D', 'C', 'B'}
```

Example: Program to Join (Union) two sets using union function

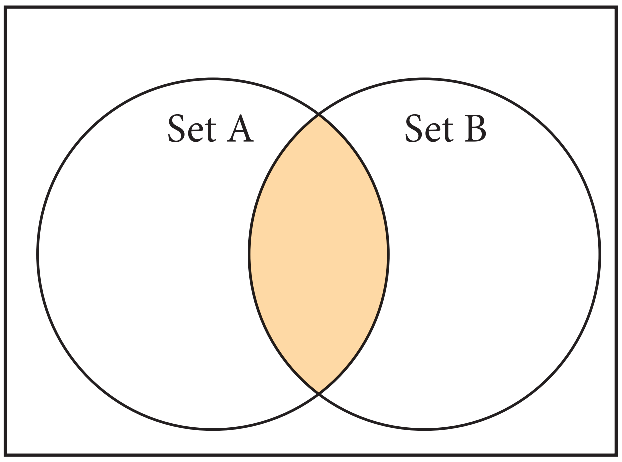
```
set_A={2,4,6,8}
set_B={'A', 'B', 'C', 'D'}
set_U=set_A.union(set_B)
print(set_U)
```

Output:

```
{'D', 2, 4, 6, 8, 'B', 'C', 'A'}
```

12th Computer Science_EM Chapter 9.indd 158 22-12-2021 16:45:11

(ii) Intersection: It includes the common elements in two sets



The operator & is used to intersect two sets in python. The function **intersection( )** is also used to intersect two sets in python.

Example: Program to insect two sets using intersection operator

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A & set_B)
```

Output:

```
{'A', 'D'}
```

Example: Program to insect two sets using intersection function

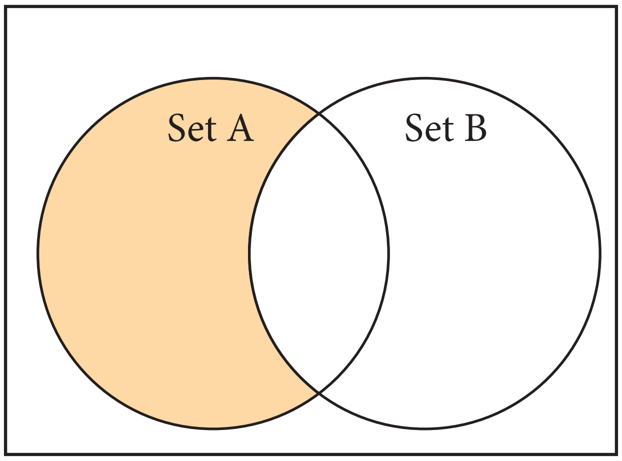
```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.intersection(set_B))
```

Output:

```
{'A', 'D'}
```

(iii) Difference

It includes all elements that are in first set (say set A) but not in the second set (say set B)



The minus (-) operator is used to difference set operation in python. The function **difference( )** is also used to difference operation.

Example: Program to difference of two sets using minus operator

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A - set_B)
```

Output:

```
{2, 4}
```

Example: Program to difference of two sets using difference function

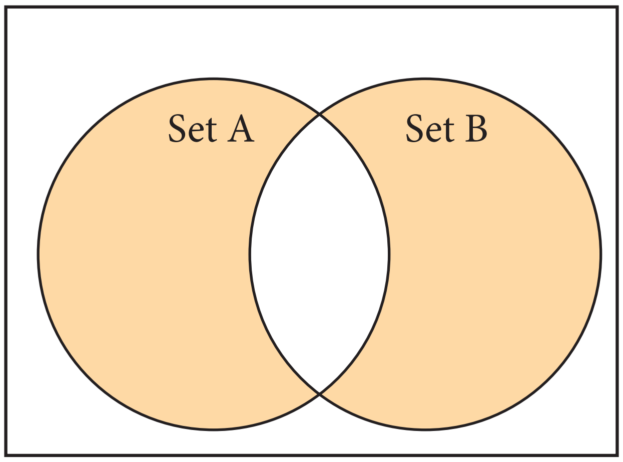
```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.difference(set_B))
```

Output:

```
{2, 4}
```

(iv) Symmetric difference

It includes all the elements that are in two sets (say sets A and B) but not the one that are common to two sets.



The caret (^) operator is used to symmetric difference set operation in python. The function **symmetric_difference( )** is also used to do the same operation.

Example: Program to symmetric difference of two sets using caret operator

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A ^ set_B)
```

Output:

```
{2, 4, 'B', 'C'}
```

Example: Program to difference of two sets using symmetric difference function

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.symmetric_difference(set_B))
```

Output:

```
{2, 4, 'B', 'C'}
```

9.3.4 Programs using Sets

Program 1: Program that generate a set of prime numbers and another set of even numbers. Demonstrate the result of union, intersection, difference and symmetirc difference operations.

Example

```
even=set([x*2 for x in range(1,11)])
primes=set()
for i in range(2,20):
        j=2
        f=0
        while j<=i/2:
                if i%j==0:
                        f=1
                j+=1
        if f==0:
                primes.add(i)
print("Even Numbers: ", even)
print("Prime Numbers: ", primes)
print("Union: ", even.union(primes))
print("Intersection: ", even.intersection(primes))
print("Difference: ", even.difference(primes))
print("Symmetric Difference: ", even.symmetric_difference(primes))
```

Output:

```
Even Numbers:  {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
Prime Numbers:  {2, 3, 5, 7, 11, 13, 17, 19}
Union:  {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20}
Intersection:  {2, 4}
Difference:  {6, 8, 10, 12, 14, 16, 18, 20}
Symmetric Difference:  {3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20}
```

9.4 Dictionaries

Introduction

In python, a dictionary is a mixed collection of elements. Unlike other collection data types such as a list or tuple, the dictionary type stores a key along with its element. The keys in a Python dictionary is separated by a colon (:) while the commas work as a separator for the elements. The key value pairs are enclosed with curly braces { }.

12th Computer Science_EM Chapter 9.indd 162 22-12-2021 16:45:11

Syntax of defining a dictionary:

```
Dictionary_Name = {        Key_1: Value_1,
                            Key_2:Value_2,
                            ……..
                            Key_n:Value_n
                            }
```

Key in the dictionary must be unique case sensitive and can be of any valid Python type.

9.4.1 Creating a Dictionary

```
# Empty dictionary
      Dict1 = { }

# Dictionary with Key
      Dict_Stud = { 'RollNo': '1234', 'Name':'Murali', 'Class':'XII', 'Marks':'451'}
```

9.4.2 Dictionary Comprehensions

In Python, comprehension is another way of creating dictionary. The following is the syntax of creating such dictionary.

Syntax

```
    Dict = { expression for variable in sequence [if condition] }
```

The if condition is optional and if specified, only those values in the sequence are evaluated using the expression which satisfy the condition.

Example

```
Dict = { x : 2 * x for x in range(1,10)}
```

Output of the above code is

```
{1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18}
```

9.4.3 Accessing, Adding, Modifying and Deleting elements from a Dictionary

Accessing all elements from a dictionary is very similar as Lists and Tuples. Simple print function is used to access all the elements. If you want to access a particular element, square brackets can be used along with key.

Example : Program to access all the values stored in a dictionary

```
MyDict = {      'Reg_No': '1221',
                'Name' : 'Tamilselvi',
                'School' : 'CGHSS',
                'Address' : 'Rotler St., Chennai 112' }
print(MyDict)
print("Register Number: ", MyDict['Reg_No'])
print("Name of the Student: ", MyDict['Name'])
print("School: ", MyDict['School'])
print("Address: ", MyDict['Address'])
```

Output:

```
    {'Reg_No': '1221', 'Name': 'Tamilselvi', 'School': 'CGHSS', 'Address': 'Rotler St., Chennai 112'}
    Register Number:  1221
    Name of the Student:  Tamilselvi
    School:  CGHSS
    Address:  Rotler St., Chennai 112
```

Note that, the first print statement prints all the values of the dictionary. Other statements are printing only the specified values which is given within square brackets.

In an existing dictionary, you can add more values by simply assigning the value along with key. The following syntax is used to understand adding more elements in a dictionary.

dictionary_name [key] = value/element

Example : Program to add a new value in the dictionary

```
MyDict = {      'Reg_No': '1221',
                'Name' : 'Tamilselvi',
                'School' : 'CGHSS', 'Address' : '
                Rotler St., Chennai 112'}
print(MyDict)
print("Register Number: ", MyDict['Reg_No'])
print("Name of the Student: ", MyDict['Name'])
MyDict['Class'] = 'XII - A'                 # Adding new value
print("Class: ", MyDict['Class'])           # Printing newly added value
print("School: ", MyDict['School'])
print("Address: ", MyDict['Address'])
```

Modification of a value in dictionary is very similar as adding elements. When you assign a value to a key, it will simply overwrite the old value.

In Python dictionary, del keyword is used to delete a particular element. The clear( ) function is used to delete all the elements in a dictionary. To remove the dictionary, you can use del keyword with dictionary name.

12th Computer Science_EM Chapter 9.indd 164 22-12-2021 16:45:11

Syntax:

```
    # To delete a particular element.
    del dictionary_name[key]
    # To delete all the elements
    dictionary_name.clear( )
    # To delete an entire dictionary
    del dictionary_name
```

Example : Program to delete elements from a dictionary and finally deletes the dictionary.

```
Dict = {'Roll No' : 12001, 'SName' : 'Meena', 'Mark1' : 98, 'Marl2' : 86}
print("Dictionary elements before deletion: \n", Dict)
del Dict['Mark1']              # Deleting a particular element
print("Dictionary elements after deletion of a element: \n", Dict)
Dict.clear()                   # Deleting all elements
print("Dictionary after deletion of all elements: \n", Dict)
del Dict
print(Dict)                    # Deleting entire dictionary
```

Output:

```
    Dictionary elements before deletion:
        {'Roll No': 12001, 'SName': 'Meena', 'Mark1': 98, 'Marl2': 86}
    Dictionary elements after deletion of a element:
        {'Roll No': 12001, 'SName': 'Meena', 'Marl2': 86}
    Dictionary after deletion of all elements:
        { }
    Traceback (most recent call last):
        File "E:/Python/Dict_Test_02.py", line 8, in <module>
        print(Dict)
NameError: name 'Dict' is not defined
```

9.4.4 Difference between List and Dictionary

(1) List is an ordered set of elements. But, a dictionary is a data structure that is used for matching one element (Key) with another (Value).

(2) The index values can be used to access a particular element. But, in dictionary key represents index. Remember that, key may be a number of a string.

(3) Lists are used to look up a value whereas a dictionary is used to take one value and look up another value.

Points to remember:

- Python programming language has four collections of data types such as List, Tuple, Set and Dictionary.
- A list is known as a "sequence data type". Each value of a list is called as element.
- The elements of list should be specified within square brackets.
- Each element has a unique value called index number begins with zero.
- Python allows positive and negative values as index.
- Loops are used access all elements from a list.
- The "for" loop is a suitable loop to access all the elements one by one.
- The append ( ), extend ( ) and insert ( ) functions are used to include more elements in a List.
- The del, remove ( ) and pop ( ) are used to delete elements from a list.
- The range ( ) function is used to generate a series of values.
- Tuples consists of a number of values separated by comma and enclosed within parentheses.
- Iterating tuples is faster than list.
- The tuple ( ) function is also used to create Tuples from a list.
- Creating a Tuple with one element is called "Singleton" tuple.
- A Set is a mutable and an unordered collection of elements without duplicates.
- A set is created by placing all the elements separated by comma within a pair of curly brackets.
- A dictionary is a mixed collection of elements.

Hands on Experience

1. Write a program to remove duplicates from a list.
2. Write a program that prints the maximum value in a Tuple.
3. Write a program that finds the sum of all the numbers in a Tuples using while loop.
4. Write a program that finds sum of all even numbers in a list.

5. Write a program that reverse a list using a loop.

6. Write a program to insert a value in a list at the specified location.

7. Write a program that creates a list of numbers from 1 to 50 that are either divisible by 3 or divisible by 6.

8. Write a program to create a list of numbers in the range 1 to 20. Then delete all the numbers from the list that are divisible by 3.

9. Write a program that counts the number of times a value appears in the list. Use a loop to do the same.

10. Write a program that prints the maximum and minimum value in a dictionary.

Evaluation

TS1FQ

Part - I

Choose the best answer (1 Marks)

1. Pick odd one in connection with collection data type

   (a) List (b) Tuple (c) Dictionary (d) Loop

2. Let list1=[2,4,6,8,10], then print(List1[-2]) will result in

   (a) 10 (b) 8 (c) 4 (d) 6

3. Which of the following function is used to count the number of elements in a list?

   (a) count() (b) find() (c)len() (d) index()

4. If List=[10,20,30,40,50] then List[2]=35 will result

   (a) [35,10,20,30,40,50] (b) [10,20,30,40,50,35]

   (c) [10,20,35,40,50] (d) [10,35,30,40,50]

5. If List=[17,23,41,10] then List.append(32) will result

   (a) [32,17,23,41,10] (b) [17,23,41,10,32]

   (c) [10,17,23,32,41] (d) [41,32,23,17,10]

6. Which of the following Python function can be used to add more than one element within an existing list?

   (a) append() (b) append_more() (c)extend() (d) more()

7. What will be the result of the following Python code?

   S=[x**2 for x in range(5)]

print(S)

(a) [0,1,2,4,5] (b) [0,1,4,9,16] (c) [0,1,4,9,16,25] (d) [1,4,9,16,25]

8. What is the use of type() function in python?

   (a) To create a Tuple

   (b) To know the type of an element in tuple.

   (c) To know the data type of python object.

   (d) To create a list.

9. Which of the following statement is not correct?

   (a) A list is mutable

   (b) A tuple is immutable.

   (c) The append() function is used to add an element.

   (d) The extend() function is used in tuple to add elements in a list.

10. Let setA={3,6,9}, setB={1,3,9}. What will be the result of the following snippet?

    print(setA|setB)

    (a) {3,6,9,1,3,9} (b) {3,9} (c) {1} (d) {1,3,6,9}

11. Which of the following set operation includes all the elements that are in two sets but not the one that are common to two sets?

    (a) Symmetric difference (b) Difference

    (c) Intersection (d) Union

12. The keys in Python, dictionary is specified by

    (a) = (b) ; (c)+ (d) :

Part - II

Answer the following questions (2 Marks)

1. What is List in Python?
2. How will you access the list elements in reverse order?
3. What will be the value of x in following python code?

   List1=[2,4,6[1,3,5]]

   x=len(List1)

4. Differentiate del with remove( ) function of List.

5. Write the syntax of creating a Tuple with n number of elements.
6. What is set in Python?

Part - III

Answer the following questions (3 Marks)

1. What are the advantages of Tuples over a list?
2. Write a shot note about sort( ).
3. What will be the output of the following code?

```
list = [2**x for x in range(5)]

print(list)
```

4. Explain the difference between del and clear( ) in dictionary with an example.
5. List out the set operations supported by python.
6. What are the difference between List and Dictionary?

Part - IV

Answer the following questions (5 Marks)

1. What the different ways to insert an element in a list. Explain with suitable example.
2. What is the purpose of range( )? Explain with an example.
3. What is nested tuple? Explain with an example.
4. Explain the different set operations supported by python with suitable example.

References

1. https://docs.python.org/3/tutorial/index.html
2. https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list
3. Python programming using problem solving approach – Reema Thareja – Oxford University press.
4. Python Crash Course – Eric Matthes – No starch press, San Francisco.