

Lists, Dictionaries and Tuples

Topic – 1 Lists

Very Short Answer Type Questions (1 marks each)

Question 1:

How many types of built-in types of sequences in Python.

Answer:

Python has six built-in types of sequences.

Question 2:

Write the output of the given Python code.

```
# !/user/bin/python
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1,2,3,4,5,6, 7];
print "list1[0]", list1[0]
```

Answer:

```
list1[0]: physics
```

Question 3:

Write the output of the given Python code.

```
# !/user/bin/python
list1 = ['physics', 'chemistry', 1997,2000];
list2 = [1,2, 3,4,5, 6, 7];
print "list2[1:5[ :", list2[1:5]
```

Answer:

```
list2[1:5[: [2,3,4,5]
```

Question 4:

How can we remove list element ?

Answer:

To remove a list element, you can use either the del statement or the remove()) method

Question 5:

Why we use `cmp(list1, list2)` in Python?

Answer:

It compares elements of both lists.

Question 6:

What is the use of `len(list)` in Python?

Answer:

It gives the total length of the list.

Question 7:

Describe `max(list)` method.

Answer:

It returns item from the list with max value.

Question 8:

Describe `min(list)` method.

Answer:

It returns item from the list with min value.

Question 9:

What is the use of `list(seq)` in Python ?

Answer:

It converts a tuple into list.

Question 10:

What do you mean by `list.append(obj)`

Answer:

Appends object `obj` to list

Question 11:

What do you mean by `list.count(obj)`

Answer:

Returns count how many times `obj` occurs in list

Question 12:

Explain `list.extend(seq)`

Answer:

Appends the contents of `seq` to list.

Question 13:

Is list.reverse() method return a value?

Answer:

This method does not return any value but reverse the given object from the list

Question 14:

Which method is used to sort objects of list.

Answer:

sort ()

Question 15:

Which function is used to reverse objects of list in place.

Answer:

list.reverse ()

Question 16:

Which function is use for removing object obj from list.

Answer:

list.remove(obj)

Question 17:

Which function is use for returning the lowest index in list that obj appears.

Answer:

List index (obj).

Short Answer Type Questions (2 marks each)

Question 1:

Define list in Python.

Answer:

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. Good thing about a list is that items in a list need not all have the same type. Creating a list is as simple as putting different comma-separated values between square brackets.

For example :

```
list1 = ['physics', 'chemistry', 1997,2000];
```

```
list2 = [1,2, 3,4, 5];
```

```
list3 = ['a", "b", "c", "d"];
```

Question 2:

Give an example to access values in lists.

Answer:

To access value in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. Following is a simple example :

```
# !/user/bin/python
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7];
print "list1 [0]:",
list1 [0] print "list2[1 :5]:", list2[1:5]
```

When the above code is executed, it produces the following result:

```
list1 [0]: physics
list2[1:5] : [2, 3, 4, 5]
```

Question 3:

Give an example to update single or multiple elements of lists

Answer:

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add elements in a list with the append() method.

Following is a simple example:

```
# !/user/bin/python
list1 = ['physics', 'chemistry', 1997, 2000];
print "Value available at index 2 :"
```

```
print list[2];
list[2] = 2001;
print "New value available at index 2 :"
```

```
print list [2];
```

Note: append()) method is discussed in subsequent section.

When the above code is executed, it produces the following result:

```
Value available at index 2 :
1997
New value available at index 2 :
2001
```

Question 4:

Give an example to remove list element.

Answer:

To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting or the remove() method if you do not know. Following is a simple example:

```
#!/user/bin/python
list1 = ['physics', 'chemistry', 1997, 2000];
print list1;
del list1 [2];
print "After deleting value at index 2 :"
```

When the above code is executed, it produces the following result:

```
['physics', 'chemistry', 1997, 2000];
After deleting value at index 2;
['physics', 'chemistry', 2000]
```

Question 5:

Describe list() with its syntax.

Answer:

The method list() takes sequence types and converts them to lists. It is used to convert a given tuple into list.

Syntax

Following is the syntax for list() method :

```
list (seq)
```

Question 6:

Write the output of the given Python code :

```
#!/user/bin/python
list1, list2 = [123, 'xyz'], [456, 'abc']
print cmp(list1, list2);
print cmp(list2, list1);
list3 = list2 + [786];
print cmp(list2, list3)
```

Answer:

This will produce the following result:

-1

1
-1

Question 7:

Write the output of the given Python code :

```
#!/user/bin/python
aList = [123, 'xyz', 'zara', 'abc', 123];
bList = [2009, 'manni'];
aList.extend (bList)
print "Extended List :", aList;
```

Answer:

This will produce the following result:

Extended List : [123, 'xyz', 'zara', 'abc', 123, 2009, 'manni']

Question 8:

Write the output of the given python code :

```
# ! 'user/bin'python
aList1 = [123, 'xvz', 'zara', 'abc'];
print "Index for xyz : " aList. index) 'xyz');
print "Index for zara :", aList. index('zara');
```

Answer:

This will produce the following result:

Index for xyz : 1 Index for xxx : 2

Question 9:

Write the output of the given python code :

```
#!/user/bin/python
aList = [123, 'xyz', 'zara', 'abc'];
aList.insert (3,2009) print "Final List:", aList
```

Answer:

Output:

Final List: [123, 'xyz', 'zara', 2009, 'abc']

Question 10:

Write the output of the given python code :

```
#!/user/bin/python
aList1 = [123, 'xyz', 'zara', 'abc'];
aList.insert (3,2009) print "Final Lista List"
```

Answer:

Output:

Final List: [123, 'xyz', 'zara1, 2009,'abc1']

Question 11:

Write the output of the given python code :

```
# !/user/bin/python
aList1 = [123, 'xyz', 'zara', 'abc'];
print "A List:", aList.pop()
print "B List:", aList.pop(2)
```

Answer:

Output:

A List: abc B List: zara

Question 12:

Write the output of the following code [CBSE Text Book]

```
A = [2, 4, 6, 8,10]
L = len (A)
S = 0
for I in range (1, L, 2):
S += A[I]
print "Sum=", S
```

Answer:

Sum = 12

Justification:

A[1] = 4 step size = 2

A[3] = 8
S = 4 + 8 = 12

Question 13:

How are lists different from strings when both are sequences ?

Answer:

The lists and strings are different in following ways :

- (a) The lists are mutable sequences while strings are immutable.
- (b) Strings store single type of elements, all characters while lists can store elements belonging to different types.
- (c) In consecutive locations, strings store the individual characters while list stores the references of its elements.

Question 14:

Write a program to calculate and display the sum of all the odd numbers in the list.

Answer:

```
pos = 0
sum = 0
while pos < len (L):
    if L[pos] %2 == 1 :
        sum = sum + L [pos]
    pos = pos + 1
print sum
```

Question 15:

Define a function overlapping () that takes two lists and returns True if they have at least one member in common, False otherwise.

Answer:

```
def overlapping (a, b):
    n1 = len (a)
    n2 = len (b)
    for i in range (n1):
        for j in range (n2):
            if a[i] == b[j] :
                return True
    else
        return False
```


Question 16:

Write a program to find all duplicates in a list. [CBSE Text Book]

Answer:

```
a=[1,2,3,2,1,5,6,5,5,5]
d={}
for elem in a:
    if elem in d:
        d[elem] += 1
    else:
        d[elem] = 1
print elem
print [x for x, y in d.items() if y > 1],
print "have duplicates"
print d
```

Output screenshot:

```
[1,2,5] have duplicates
{1:2,2: 2,3:1,5:4, 6: 1}
```

Long Answer Type Questions (4 marks each)**Question 1:**

What is string in Python and how can we access values in list.

Answer:

The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number – its position or index. The first index is zero, the second index is one, and so forth. Python has six built-in types of sequences, but the most common ones are lists and tuples.

There are certain things you can do with all sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership. In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements.

Python Lists:

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between brackets. Good thing about a list is that items in a list need not all have the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example:

```
list1 = ['physics', 'chemistry', 1997,2000];
list2 = [1, 2, 3,4,5];
```

```
list3 = ["a", "b", "c", "d"];
```

Like string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

Accessing Values in Lists :

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. Following is a simple example :

```
# !/user/bin/python
list = ['physics', 'chemistry', 1997,2000];
list2 = [1,2, 3,4,5,6, 7];
print "list1 [0]:", list1 [0]
print "list2[1:5]:", list2[1:5]
list 2[1:5]: [2, 3, 4, 5]
```

Question 2:

How can we update and delete list in python.

Answer:

Updating Lists:

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method. Following is a simple example:

```
# !/user/bin/python
list = ['physics', 'chemistry', 1997, 2000];
print "Value available at index 2 :"
```

```
print list[2];
list [2] = 2001;
print "New value available at index 2 :"
```

```
print list [2];
```

When the above code is executed it produces the following result:

Value available at index 2 :

1997

New value available at index 2 :

2001

Delete List Elements:

To remove a list element, you can use either the del statement if you know exactly which

element(s) you are deleting of the remove() method if you do not know.
Following is a simple example :

```
# !/user/bin/python
list1 = ['physics', 'chemistry', 1997, 2000];
print list1; del list1 [2];
print "After deleting value at index 2:"
print list1;
```

When the above code is executed, it produces the following result:

```
['physics', 'chemistry', 1997, 2001]
After deleting value at index 2 :
['physics', 'chemistry', 2000]
```

Question 3:

Write a program to input any two matrices and print sum of matrices.

Answer:

```
import random
m1 = input ("Enter total number of rows in the first matrix")
n1 = input ("Enter total number of columns in the first matrix")
a = [[random.random() for row in range(m1) for col in range (n1)]
for i in range (m1)
for j in range (n1):
a[i][j] = input()
m2 = input ("Enter total number of rows in the second matrix")
n2 = input ("Enter total number of columns in the second matrix")
b = [[random.random () for row in range (m2)] for col in range (n2)]
for i in range (m2):
for j in rnage (n2):
b [i][j] = input ()
c = [[random.random () for row in range (m1)] for col in range (n1)]
if ((m1 = m2) and (n1 = = b)):
print "Output is"
for i in range (m1):
for j in range (n1)]
c[i][j] = a[i][j] + b[i][j]
print c[i][j], 'it',
print
else :
print "Matrix addition is not possible"
```

Question 4:

Write a program to input any two matrices and print product of matrices.

Answer:

```
import random
m1 = input ("Enter number of rows in first matrix")
n1 = input ("Enter number of columns in first matrix")
a = [[random.random () for row in range (m1)] for col in range (n1)]
for i in range (m1):
    for j in range (n1):
        a[i][j] = input ()
m2 = input ("Enter the number of rows in the second matrix")
n2 = input ("Enter the number of columns in the second matrix")
b = [[random.random () for row in range (m2)] for cal in range (n2)]
for i in range (m2):
    for j in range (n2):
        b[i][j] = input ()
c = [[random.random () for row in range (m1)] for col in range (n2)]
if (n1 == m2):
    for i in range (m1):
        for j in range (n2):
            c[i][j] = 0 for kin range (n1):
            c[i][j] += a[i][k]*b[k][j]
            print c[i][j], '\t',
        print
    else:
        print "Multiplication is not possible"
```

Question 5:

Write a program to input any matrix and print both diagonal values of the matrix.

Answer:

```
Import random
m = input ("Enter the number of rows")
n = input ("Enter the number of columns")
a = [[random.random () for row in range (m)] for cal in mange (n)]
if i in range (m):
    for j in range (n):
        a[i][j] = input ()
    print "First diagonal"
    for i in range (m):
        print a[i][j], '\t'
    k = m - 1
```

```

print "second diagonal"
for j in range (m):
    print a[j][k], '\t'
    k- = 1
else:
    print "Diagonal values are not possible"

```

Question 6:

Write a program to input n x m matrix and find the sum of all numbers.

Answer:

```

def summat (a, m, n):
    s = 0
    for i in range (m):
        for j in range (n):
            s + = a[i][j]
    return s

```

Question 7:

Write a program to perform various list operations after displaying menu.

Answer:

```

ch = 0
list = []
while true :
    print "List Menu"
    print "1. Adding"
    print "2.Modify"
    print "3.Delete"
    print "4.Sort list"
    print "5.Display list"
    print "6.Exit"
    ch = int (raw_input ("Enter your choice :"))
    if ch == 1 :
        print "1.Add element"
        print "2.Add a List"
        ch1 = int (raw_input
            ("Enter choice 1 or 2:"))
        if ch1 == 1:
            item = int
            (raw_input ("Enter element:"))
            pos = int (raw_input
                ("Enter position to add : "))

```

```

list.insert (pos, item)
elif chi == 2 :
list2 = eval (raw_input ("Enter list:"))
list.extend (lst2)
else:
print "Valid choices are 1 or 2"
print "Successfully added" elif ch == 2 :
pos = int (raw_input
("Enter position :"))
item = int (raw_input
("Enter new value :"))
old = list[pos]
list[pos] = item
print old, 'modified with vlaue', item elif ch == 3:
print "1.Delete element by position"
print "2.Delete element by value"
ch1 = int (raw_input ("Enter you choice 1 or 2'))
if chi == 1:
pos = int (raw_input ("Enter position :"))
item = list.pop (pos)
print item, "deleted"
elif ch1 == 2:
item = int (raw_input ("Enter element:"))
pos = list.remove (item)
print "Succcessfully delted"
else :
print "Valid choices are 1 or 2"
elif ch == 4 :
print "1.Ascending"
print "2.Descending"
chi = int (raw input ("Enter choice 1 or 2"))
if chi == 1:
list. sot () elif ch1 == 2:
list.sort (reverse = True)
else :

```

print "Valid choices are 1 or 2"

```

elif ch == 5:
print list elif ch == 6:
break
else :
print "valid choice 1 to 6"

```

Question 8:

Write a program to pass any list and to arrange all numbers in descending order.

Answer:

```
def arrange (l, n)
for i in range (n - 1):
for j in range (n - i - 1):
if l[j] > l[j + 1] :
temp = l[j]
l[j] = l[j+1]
l[j+1] = temp
```

Question 9:

Write the output for the following codes. **[CBSE Text Book]**

```
A= {10:1000,20:2000,30:3000,40:4000, 50:5000}
print A.items()
print A.keys()
print A.values()
```

Answer:

```
[(40, 4000), (10, 1000), (20, 2000), (50, 5000), (30, 3000)]
[40,10, 20, 50, 30]
[4000,1000, 2000,5000,3000]
```

Topic – 2 Tuples

Very Short Answer type Questions [1 mark each]

Question 1:

What is a tuple ?

Answer:

Tuple is a sequence of immutable Python objects

Question 2:

Can we remove individual tuple elements ?

Answer:

No, we cannot remove individual tuple elements.

Question 3:

Which function is use for comparing elements of both tuples.

Answer:

`cmp(tuple1, tuple2)`

Question 4:

Which function gives the total length of the tuple.

Answer:

`len(tuple)`

Question 5:

Which function returns item from the tuple with max value.

Answer:

`max(tuple)`

Short Answer type Questions [2 mark each]

Question 1:

What is tuple ?

Answer:

Tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists.

The only difference is that tuples can't be changed i.e., tuples are immutable and tuples use parentheses and lists use square brackets. Creating a tuple is as simple as putting different comma-separated values and optionally you can put these comma-separated values between parentheses also.

For example :

`tup1 = ('physics', 'chemistry', 1997, 2000);`


```
tup2 = (1,2, 3,4, 5);  
tup3 = "a", "b", "c", "d";
```

Question 2:

Write the output of the given python code :

```
#!/user/bin/python  
tup1 = (12, 34.56);  
tup2 = ('abc', 'xyz');  
#Following action is not valid for tuples  
#tup1 [0] = 100;  
#So let's create a new tuple as follows  
tup3 = tup1 + tup2;  
print tup3;
```

Answer:

Output :

(12,34.56, 'abc', 'xyz')

Question 3:

Write the output of the given python code :

```
# !/user/bin/python  
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')  
print cmp (tuple1, tuple2) ;  
print cmp (tuple2, tuple1) ;  
tuple3 = tuple2 + (786,);  
print cmp (tuple2, tuple3)
```

Answer:

Output :

-1

1

-1

Question 4:

Write the output of the given python code :

```
#!/user/bin/python  
tuple1, tuple2 = (123, 'xyz', 'zara', 'abc'), (456, 700, 200)  
print "min value element : ", min (tuple1);  
print "min value element : ", min (tuple2);
```

Answer:

Output :

min value element : 123

min value element : 200

Long Answer type Questions [4 mark each]

Question 1:

What is tuple in Python and how can we access values in tuples ?

Answer:

Tuple is a sequence of immutable Python object. Tuples are sequences, just like lists. The only difference is that tuples can't be changed i.e., tuples are immutable and tuples use parentheses and lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values and optionally you can put these comma -separated values between parentheses also.

For example :

```
tup1 = ('physics', 'chemistry', 1997, 2000) ;
```

```
tup2 = (1, 2, 3, 4, 5);
```

```
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing :

```
tup1 = ( ) ;
```

To write a tuple containing a single value you have to include a comma, even though there is only one

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and tuples can be sliced, concatenated and so on.

Accessing Values in Tuples :

To access value in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index.

Following is a simple example :

```
# !/user/bin/python
```

```
tup1 = ('physics', 'chemistry', 1997, 2000);
```

```
tup2 = (1, 2, 3, 4, 5, 6, 7);
```

```
print "tup1[0]", tup1[0]
```

```
print "tup2[1:5]:", tup2[1:5]
```

When the above code is executed, it produces the following result :

```
tup1[0] : physics
```

```
tup2[1:5]: [2, 3, 4, 5]
```

Question 2:

How we update and delete tuples ?

Answer:

Updating Tuples :

Tuples are immutable which means you cannot update them or change values of tuple elements. But we are able to take portions of an existing tuples to create a new tuples as follows. Following is a simple example :

```
#!/user/bin/python
```

```
tup1 =(12,34,56);
```

```
tup2 = ('abc', 'xyz');
```

#Following action is not valid for tuples

```
#tup1[0] = 100;
```

#So lets create a new tuple as follows :

```
tup3 = tup1 + tup2;
```

```
print tup3;
```

When the above code is executed, it produces the following result :

```
(12,34.56, 'abc', 'xyz')
```

Delete Tuple Elements :

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. Following is a simple example :

```
#!/user/bin/python
```

```
tup = ('physics', 'chemistry', 1997,2000);
```

```
print tup;
```

```
del tup;
```

```
print "After deleting tup:"
```

```
print tup;
```

This will produce following result. Note an exception raised, there is because after del tup tuple does not exist any more :

```
('physics', 'chemistry', 1997,2000)
```

After deleting tup :

Traceback (most recent call last) :

File "test.py", line 9, in < modulo>

```
print tup;
```

Name Error : name 'tup' is not defined

Question 3:

Explain cmp(tuple1, tuple2) with example.

Answer:

```
cmp(tuple1, tuple2)
```

Description :

The method cmp() compares elements of two tuples.

Syntax :

Following is the syntax for cmp() method :

```
cmp (tuple1, tuple2)
```

Parameters :

- tuple 1 – This is the first tuple to be compared
- tuple 2 – This is the second tuple to be compared

Return Value :

If elements are of the same type, perform the compare and return the result. If elements are different types, check to see if they are numbers.

- If numbers, perform numeric conversion if necessary and compare.

- If either element is a number, then the other element is “larger” (numbers are “smallest”).
- Otherwise, types are sorted alphabetically by name.

If we reached the end of one of the tuples, the longer tuple is “larger”. If we exhaust both tuples and share the same data, the result is a tie, meaning that 0 is returned.

Example :

The following example shows the usage of cmp() method

```
#!/user/bin/python
```

```
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')
```

```
print cmp(tuple1, tuple2);
```

```
print cmp(tuple1, tuple1);
```

```
tuple3 = tuple2 + (786,);
```

```
print cmp (tuple2, tuple3)
```

Let us compile and run the above program, this will produce the following result :

```
-1
```

```
1
```

```
-1
```

Question 4:

Explain tuple(seq) function of the python also give example.

Answer:

```
tuple(seq)
```

Description :

The method tuple() is used to convert list into a tuple.

Syntax :

Following is the syntax for tuple() method :

```
tuple (seq)
```

Parameters :

- seq – This is a list to be converted into tuple. Return Value :

This method returns the tuple.

Example :

The following example shows the usage of tupe() method.

```
# !/user/bin/python
```

```
aList = [123, 'xyz', 'zara', 'abc'];
```

```
aTuple = tuple (aList)
```

```
print “Tuple elements :”,aTuple
```

Let us compile and run the above program, this will produce the followin’, result :

```
Tuple elements : (123, 'xyz', 'zara', 'abc')
```

Question 5:

Write a program to input ‘n’ numbers and store it in tuple.

Answer:

```

t = tuple ( )
n = input ("Enter any number")
print "Enter all numbers one after other"
for i in range (n) :
a = input ("Enter number")
t = t+(a, )
print "Output is"
print t

```

Question 6:

Write a program to input any two tuples and interchange the tuple values.

Answer:

```

t1 = tuple( )
n = input("Total number of values in first tuple")
for i in range (n):
a = input("Enter elements")
t2 = t2 + (a, )
print "First tuple"
print t1
print "Second tuple"
print t2
t1, t2 = t2, t1
print "After swapping"
print "First tuple"
print t1
print "Second tuple"
print t2

```

Question 7:

Write a program to input 'n' numbers and separate the tuple in the following manner.

Example

T=(10,20,30,40,50,60)

T1=(10,30,50)

T2=(20,40,60)

Answer:

```

t=tuple( )
n=input(" Enter number of values:")
for i in range(n):
a=input("enter elements")
t=t+(a,)
t1=t2=tuple( )
for i in t:
if i%2 == 0:
t2= t2 + (i,)

```

```
else:
    tl=tl+(i,)
    print "First Tuple"
    print t1
    print "Second Tuple"
    print t2
```

Output:

Enter number of values: 10

enter elements 1

enter elements 9

enter elements 8

enter elements 5

enter elements 2

enter elements 3

enter elements 6

enter elements 4

enter elements 7

enter elements 10

First Tuple (1, 9, 5, 3, 7)

Second Tuple (8,2,6,4,10)

Question 8:

Write a program to input 'n' employees' salary and find minimum & maximum salary among 'n' employees.

Answer:

```
t=tuple( )
n=input("Enter total number of employees")
for i in range(n):
    a=input("enter salary of employee:")
    t=t+(a,)
    print "maximum salary=",max(t)
    print "minimum salary=",min(t)
```

Output :

Enter total number of employees 3

enter salary of employee: 7800

enter salary of employee: 8934

enter salary of employee:

6544 maximum salary = 8934

minimum salary = 6544

Topic – 3

Dictionaries

Short Answer type Questions [2 mark each]

Question 1:

How are dictionaries different from lists ?

Answer:

The dictionary is similar to lists in the sense that it is also a collection of data-items just like lists but it is different from lists in the sense that lists are sequential collections and dictionaries are non-sequential collections. In lists, where there is an order associated with the data items because they act as storage units for other objects or variables created. Dictionaries are different from lists and tuples because the group of objects they hold are not in any particular order, but rather each object has its own unique name, commonly known as a key.

Question 2:

When are dictionaries more useful than lists ?

Answer:

Dictionaries can be much more useful than lists. For example : suppose we wanted to store all our friend's cell phone numbers. We could create a list of pairs (phone number, name), but once this list becomes long enough searching this list for a specific phone number will get time consuming. Better would be if we could index the list by our friend's name. This is precisely what a dictionary does.

Question 3:

Write a program to print the worker's informations (Name age, salary) in records format.

Answer:

```
Employees = {'Rohan' : {'age' : 20, 'salary' : 10000}, 'Mohan' : {'age' : 30, 'salary' : 15000}}
for key in Employees :
    print "Employee", key + ':'
    print 'Age :' + str (Employees [key] ['age'])
    print 'Salary :' + str (Employees [key] ['salary'])
```

Question 4:

What is a key-value pair with reference to Python dictionary ?

Answer:

A dictionary is a mapping between a set of indices (which are called keys) and a set of values. Each key maps a value. The association of a key and a value is called a key-value pair.

Question 5:

What are the characteristics of Python Dictionaries ?

Answer:

The 3 main characteristics of a dictionary are :

1. Dictionaries are Unordered : The dictionary elements (key-value pairs) are not in ordered form.
2. Dictionary Keys are Case Sensitive : The same key name but with different case are treated as different keys in Python dictionaries.
3. No duplicate key is allowed : When duplicate keys encountered during assignment, the last assignment wins.
4. Keys must be immutable : We can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed.

Long Answer type Questions [4 mark each]**Question 1:**

Explain `cmp(dict1, dict2)` with example.

Answer:

Description

The method `cmp()` compares two dictionaries based on key and values.

Syntax

Following is the syntax for `cmp()` method :

`cmp(dict1, dict2)`

Parameters

- `dict1` – This is the first dictionary to be compared with `dict2`.
- `dict2` – This is the second dictionary to be compared with `dict1`.

Return value :

This method returns 0 if both dictionaries are equal, - 1 if `dict1 < dict2` and 1 if `dict1 > dict2`.

Example :

The following example shows the usage of `cmp()` method.

```
# !/user/bin/python
```

```
dict1 = {'Name' : 'Zara', 'Age' : 7};
```

```
dict2 = {'Name1' : 'Mahnaz', 'Age' : 27};
```

```
dict3 = {'Name' : 'Abid', 'Age' : 27};
```

```
dict4 = {'Name' : 'Zara', 'Age' : 7};
```

```
print "Return Value : %d", % cmp (dict1, dict2)
```

```
print "Return Value : %d", % cmp (dict2, dict3)
```

```
print "Return Value : %d", % cmp (dict1, dict4)
```

Let us compile and run the above program, this will produce the following result :

Return Value : - 1 .

Return Value : 1

Return Value : 0

Question 2:

Explain str(dict) with example.

Answer:

Description

The method str() produces a printable string representation of a dictionary.

Syntax

Following is the syntax for str() method :

str(dict)

Parameters

- dict – This is the dictionary.

Return Value

This method returns string representation.

Example :

The following examples shows the usage of str() method.

```
#!/user/bin/python
```

```
diet = {'Name' : 'Zara', 'Age' : 7};
```

```
print "Equivalent String : %s" %str (dict)
```

Let us compile and run the above program, this will produce the following result :

```
Equivalent String : {'Age' : 7, 'Name' : 'Zara'}
```

Question 3:

Describe dict.fromkeys() with example.

Answer:

dict.fromkeys()

The method fromkeys() creates a new dictionary with keys from seq and values set to value.

Syntax

Following is the syntax for fromkeys() method :

dict.fromkeys{seq[,value]}

Parameters :

- seq – This is the list of values which would be used be used for dictionary keys preparation.
- value – This is optional, if provided then value would be set to this value.

Return Value

This method returns the list.

Example :

The following example shows the usage of fromkeys() method.

```
#!/user/bin/python
```

```
seq = {'name', 'age', 'sex'}
```

```
diet = dict.from keys(seq)
```

```
print "New Dictionary : %s" % str(dict)
```

```
dict = dict.from keys(seq, 10)
```

```
print "New Dictionary : %s" % str(dict)
```

Let us compile and run the above program, this will produce the following result :

New Dictionary : {'age' : None, 'name' : None, 'sex': None}
New Dictionary : {'age' : 10, 'name' : 10, 'sex' : 10}

Question 4:

Write a program to input total number of sections and class teacher's name in 11th class and display all information on the output screen.

Answer:

```
class xi = diet ( )
n = input ("Enter total number of section in xi class")
i = 1
while i <= n :
a = raw_input ("enter section")
b = raw input ("enter class teacher name")
classxi[a] = b
i = i + 1
print "Class", "\t", "Section",
\t,, "Teacher Name"
for i in classxi :
print "XI", "\t", i, "\t", class xi[i]
```

Question 5:

Write a Python program to input 'n' names and phone numbers to store it in a dictionary and to input any name and to print the phone number of the particular name.

Answer:

```
phonebook = dict( )
n = input ("Enter total number of friends")
i = 1
while i<=n :
a = raw_input ("Enter name")
b = raw_input ("Enter phone number")
phonebook [a] = b
i = i + 1
name = raw_input ("Enter name")
f = 0
l = phonebook.keys( )
for i in l :
if (comp (i, name) == 0) :
print "Phone number =", phonebook[i]
f = 1
if (f == 0):
print "Given name not exist"
```

Question 6:

Write a Python program to input 'n' names and phone numbers to store it in a dictionary and to search and print the phone number of that particular name.

Answer:

```
phonebook=dict( )
n=input(" Enter total number of friends to create a telephone book")
i=1
while i<=n:
a=raw_input("enter name")
b=raw_input("enter phone number")
phonebook[a]=b
i=i+1
name=raw_input("enter friend's name to search for")
f=0
l=phonebook.keys( )
for i in l:
if(cmp(i,name) == 0):
print "Phone number= ",phonebook[i]
f=1
if (f==0):
print "Given name not exist"
```

Output:

Enter total number of friends to create a telephone book 5

enter name Ramu

enter phone number 9842311111

enter name Raju

enter phone number 9842309564

enter name Sarita

enter phone number 9842398765

enter name Sowmya

enter phone number 9842399922

enter name Karan

enter phone number 9842366554

enter friend's name to search for Sarita

Phone number = 9842398765

Question 7:

Write the code to input any 5 years and the population of any city and print it on the screen.

Answer:

```
city=dict( )
n=5
i=1
```

```

while i<=n:
a=raw_input(" enter city name")
b=raw_input("enter population")
city[a] =b
i=i+1
print city
Output :
enter city name Chennai
enter population 3400000
enter city name Mumbai
enter population 8900000
enter city name Bangalore
enter population 98700
enter city name Calicut
enter population 560000
enter city name Hyderabad
enter population 900000
{'Bangalore': ' 98700'/ Hyderabad': ' 900000', ' Chennai': ' 3400000', ' Mumbai': ' 8900000', '
Calicut': ' 560000'}

```

Question 8:

Write a code to create customer's list with their number & name and delete any particular customer using his /her number.

Answer:

```

customer=dict( )
n=input(" Enter total number of customers")
i=1
while i<=n:
a=raw_input("enter customer name")
b=raw_input("enter customer number")
customer[a]=b
i=i+1
name=raw_input(" enter customer name to delete")
del customer[name]
l=customer.keys( )
print "Customer Information"
print "Name"/\t/'number"
for i in 1:
print i/\t',customer[i]
Output :
Enter total number of customers 5
enter customer name Rohan
enter customer number 123678

```

```
enter customer name Ranjish
enter customer number 123677
enter customer name Suraj
enter customer number 123899
enter customer name Damu
enter customer number 123777
enter customer name Dhariya
enter customer number .123885
enter customer name to delete Suraj
Customer Information
Name number
Rohan 123678
Dhariya 123885
Ranjish 123677
Damu 123777
```

Question 9:

Find errors from the following codes:

```
c=dict( )
n=input(Enter total number) i=l
while i<=n
a=raw_input("enter place")
b=raw_input(" enter number")
c(a)=b
i=i+l
print "place","\\t"/"number"
for i in c:
print i,"\\t",cla[i]
```

Answer:

```
c=dict()
n=input("Enter total number" )
i=l
while i<=n:
a=raw_input("enter place")
b=raw_input("enter number")
c[a]=b
i=i+l
print "place","\\t"/"number"
for i in c:
print i,"\\t".c[i]
```