CHAPTER 02

Tuples

In this Chapter...

- Tuple *vs* List
- Creating a Tuple in Python
- Accessing Tuples
- Traversing a Tuple
- Comparing Tuples
- Membership Operators
- Common Tuple Operations
- Deleting a Tuple

A tuple is a collection of Python objects separated by commas (,) and put the elements in parentheses. Tuples are immutable by design which means they cannot be changed after creation. Tuple holds a sequence of heterogeneous elements. Tuples store a fixed set of elements and do not allow changes.

Tuple *vs* List

- Elements of a tuple are immutable whereas elements of a list are mutable.
- Tuples are declared in parentheses ( ), while lists are declared in square brackets [ ].
- Iterating over the elements of a tuple is faster compared to iterating over a list.

Creating a Tuple in Python

To create a tuple in Python, put all the elements in a parentheses ( ), separated by commas. We can have tuple of same type of data items as well as mixed type of data items.

```
a = (34, 76, 12, 90)
b = ('s', 3, 6, 'a')
c = (34, 0.5, 75)
d = ()
```

We can create different types of tuple in Python, which are as follows:

Empty Tuple

Empty tuple can be created in Python using ( ). It takes the truth value as false or equivalent of 0.

Here is the two ways to create empty tuple as

```
>>>t = ()
>>>print (t)
()
>>>t= tuple ()
>>>print(t)
()
```

Single Element Tuple

Creating a single element tuple is very complicated because it considered as integer value, if you give single element in a tuple.

```
>>>t=(8)
>>>t
8
>>>t=("a")
>>>t
'a'
```

For creating a single element tuple, you have to put comma (,) after the element.

```
>>>t = 8,
>>>t
(8,)
>>>t1 = (5,)
>>>t1
(5,)
>>>t2=("a",)
>>>t2
('a',)
```

Nested Tuple

Nested tuples are tuple objects where the elements in the tuples can be tuples themselves.

For example,

```
>>>t=(9, 6, 9, (1, 4, 8), 7)
>>>print(t)
(9, 6, 9, (1, 4, 8), 7)
```

It contains 5 elements while inner tuple contains 3 elements as (1, 4, 8). Tuple will considered (1, 4, 8) as one element.

Mixed Data Types Tuple

It can be created to place different data types such as integers, strings, double etc, into one tuple.

For example,

```
>>>t1=('Maths', 90, 89, 'English', 78.5)
>>>t1
('Maths', 90, 89, 'English', 78.5)
```

Creating Tuple from an Existing Sequence

In Python, tuple( ) method is used to create tuple from an existing sequence.

Syntax new_tuple_name = tuple (sequence)

Here is the sequence includes list, string, tuple etc.

For example,

```
>>>t="PROGRAM"
>>>t1=tuple(t)
>>>t1
('P', 'R', 'O', 'G', 'R', 'A', 'M')
>>>t=tuple("PROGRAM")
>>>t
('P', 'R', 'O', 'G', 'R', 'A', 'M')
>>>t1=['P', 'R', 'O', 'G', 'R', 'A', 'M']
>>>t2=tuple(t1)
>>>t2
('P', 'R', 'O', 'G', 'R', 'A', 'M')
```

tuple ( ) method is also used to create tuple of characters and integers which entered through keyboard.

For example,

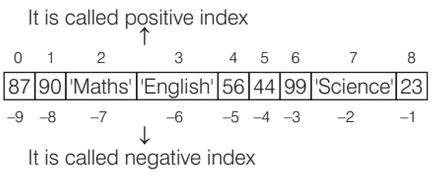
```
>>>t=tuple (input("Enter the elements:"))
Enter the elements : 24567
>>>t
('2', '4', '5', '6', '7')
>>>b=tuple(input("Enter string :"))
Enter string : PUBLICATION
>>>b
('P', 'U', 'B', 'L', 'I', 'C', 'A', 'T', 'I', 'O', 'N')
```

Accessing Tuples

To access the tuple's elements, index number is used. Tuples are accessed similar of list except for the mutability. Use the index operator [ ], elements of a tuple can be access. The index should be an integer. Index of 0 refers to first element, 1 refers to second element and so on. While the index of **– 1** refers to the last element, **– 2** refers to the second last element and so on.

For example,

```
t1 = (87, 90, 'Maths', 'English', 56, 44, 99, 'Science', 23)
>>>t1[1]
90
>>>t1[–2]
'Science'
>>>t1[7]
'Science'
>>>t1[4]
56
>>>t1[–3]
99
```



If we give index value out of range, then it will give error

```
>>>t1[9]
Trackback (most recent call last) :
File "<pyshell#8>", line 1, in <module>
    t1[9]
IndexError : tuple index out of range.
```

If you give index number with decimal point, it will also give error

```
>>>t1[4.5]
Trackback(most recent call last):
File"<pyshell#9>", line1, in <module>
    t1[4.5]
TypeError: tuple indices must be integers or slices, not float.
```

Traversing a Tuple

Traversing a tuple is a technique to access an individual element of that tuple. It is also called iterate over a tuple.

There are multiple ways to iterate over a tuple in Python.

These are as follows:

Using for loop

The most common and easy way to traverse a tuple is with for loop. This method is used when you want to iterate all elements of a tuple.

Syntax for variable in tuple_name :

For example,

```
t=('P', 'Y', 'T', 'H', 'O', 'N')
for i in t :
    print(i)
```

Output

P
Y
T
H
O
N

Using for loop with range ( )

There is another method to traverse a tuple using for loop with range ( ). This is also used len() function with range.

This method is used when you want to iterate specified elements of a tuple.

Syntax for index in range(len(tuple_name)):

For example,

```
t=('P', 'Y', 'T', 'H', 'O', 'N')
for i in range (len(t)):
    print(t[i])
```

Output

P
Y
T
H
O
N

e.g. Program to display the elements of tuple ('A', 'R', 'I', 'H', 'A', 'N', 'T') in separate line with their index number.

```
tuple1=('A', 'R', 'I', 'H', 'A', 'N', 'T')
l=len(tuple1)
for i in range (l):
    print("Character:", tuple1[i],
    "at index number:", i)
```

Output

Character : A at index number : 0
Character : R at index number : 1
Character : I at index number : 2
Character : H at index number : 3
Character : A at index number : 4
Character : N at index number : 5
Character : T at index number : 6

Comparing Tuples

A comparison operator in Python is also called Python relational operator (<, >, = = , ! =, > =, < =) that compares the values of two operands and returns True or False based on whether the condition is met.

Comparison operators for comparing tuples are as follows

Operators	Description	Example
Less than (<)	It checks if the left value is lesser than that on the right.	>>>a=(3, 5, 2, 7) >>>b=(3, 7, 0, 2) >>>a<b True
Greater than (>)	It checks if the left value is greater than that on the right.	>>>a=(4, 7, 2, 8) >>>b=(3, 7, (2), 8) >>>a>b True
Less than or Equal to (<=)	It returns True only if the value on the left is either less than or equal to that on the right of the operator.	>>>a=(0, 5, 1, 2) >>>b=(3, 4, 2, 5) >>>b<=a False
Greater than or Equal to (>=)	It returns True only if the value on the left is greater than or equal to that on the right of the operator.	>>>a=(4, 7, 2) >>>b=(2,(6,7), 4) >>>a>=b True
Equal to (= =)	It returns True if the values on either side of the operator are equal.	>>>a=(1,(2,3), 4) >>>b=(1,2,3,4) >>>a==b False
Not equal to (!=)	It returns True if the values on either side of the operator are unequal.	>>>a=(6, 7, 5) >>>b=(6, 4, 2) >>>a!=b True

Membership Operators

These operators are used to check whether a value/variable exists in the tuples. These operators return True or False as per the conditions met.

Membership operators in Python are two types as follows

in Operator

This operator is used to check if a value exists in a sequence or not. If it exists in the sequence, then it will return True else it will return False.

For example,

```
tuple1=(1,2,3,4,5)
tuple2=(6,7,8,9)
for item in tuple1:
    if item in tuple2:
        print("overlapping")
    else:
        print("not overlapping")
```

Output

not overlapping

not in Operator

This operator is the opposite of 'in' operator. So, if a value does not exist in the sequence then it will return a True else it will return a False.

For example,

```
x=44
y=30
tuple = (45, 65, 30, 78, 512)
if(x not in tuple):
    print("x is NOT present in given tuple")
else :
    print ("x is present in given tuple")
if(y in tuple):
    print("y is present in given tuple")
else:
    print("y is NOT present in given tuple")
```

Output

x is NOT present in given tuple

y is present in given tuple

Common Tuple Operations

We can perform various operations on tuple in Python. Some of them are describe below:

Concatenate Tuples

To concatenate tuples, (+) operator is used in Python. This operator can easily add the whole of one tuple to other tuple and perform concatenation.

For example,

```
>>>t1=(45, 65, 23, 9)
>>>t2=(34, 23, 65)
>>>t=t1+t2
>>>t
(45, 65, 23, 9, 34, 23, 65)
```

Concatenate operator (+) cannot add one tuple with other type as number or string. It will give error in such conditions.

For example,

```
>>>t1=(5, 3, 8, 3)
>>>t=t1+4
Trackback (most recent call last):
File "<pyshell # 5>", line1, in <module>
    t=t1+4
TypeError : can only concatenate tuple (not "int") to tuple
>>>t1=(4,5,3)
>>>t = t1 + "Hello"
Trackback (most recent call last):
File "<pyshell#6>", line1, in <module>
    t = t1 + "Hello"
TypeError: can only concatenate tuple (not "str") to tuple
```

Replicate Tuple

You can repeat the elements of the tuple using (*) operator. This operator is used to replicate the tuple.

For example,

```
>>>t1=(4, 6, 2, 8)
>>>t=t1*3
>>>t
(4, 6, 2, 8, 4, 6, 2, 8, 4, 6, 2, 8)
```

This operator cannot multiply two tuples.

```
>>>t1=(4, 5, 3)
>>>t2=(5, 0, 8)
>>>t=t1*t2
Trackback (most recent call last):
File "<pyshell#11>", line 1 , in <module>
    t=t1*t2
TypeError : can't multiply sequence by non-int of type
'tuple'.
```

Tuple Slicing

In Python, there are multiple ways to display the tuple with all elements, but to display a specific range of elements from the tuple, we use slicing operation. This operation is performed on tuples with the use of colon (:).

Syntax tuple_name [Start:Stop]

For example,

```
>>>tuple1=(4, 7, 3, "This", 6, "That", "These", 8, 9,
"Those")
>>>t1=tuple1 [3:6]
>>>t1
('This', 6, 'That')
```

To display elements from beginning to a range use [: index], to display elements from end use [:-index] and to display elements from specific index till the end use [Index :]

For example,

```
>>>tuple1=(4, 7, 3, "This", 6,"That","These", 8,9,"Those")
>>>t2=tuple1[:6]
>>>t2
(4, 7, 3, 'This', 6, 'That')
>>>t3=tuple1[:-4]
>>>t3
(4, 7, 3, 'This', 6, 'That')
>>>t4=tuple1[-4:]
>>>t4
('These', 8, 9, 'Those')
>>>t5=tuple1[7:]
>>>t5
(8, 9, 'Those')
>>>t=(4,5,(3,7,5),9,2)
>>>t1=t[2:5]
>>>t1
((3, 7, 5), 9, 2)
>>>t2=t[4:-2]
>>>t2
()
```

We can also print all elements of tuple in reverse order using [::-1]

```
>>>t=tuple1[::-1]
>>>t
('Those', 9, 8, 'These', 'That', 6, 'This', 3, 7, 4)
>>>t=tuple1[2:-4]
>>>t
(3, 'This', 6, 'That')
```

Tuples are also provide slice steps which used to extract elements from tuple that are not consecutive.

Syntax t=tuple_name[Start : Stop : Step]

Here,

- **Start** integer where the slicing of the object starts.
- **Stop** integer until which the slicing takes place. The slicing stops at index stop-1.
- **Step** integer value which determines the increment between each index for slicing.

For example,

```
>>>t=(5, 3, “He”,“She”, (4, 3, “It”),3,“They”, 8,9 “We”)
>>>t1=t[2:8:2]
>>>t1
(‘He’, (4, 3, ‘It’), ‘They’)
>>>t2=t[::3]
>>>t2
(5, ‘She’, ‘They’, ‘We’)
>>>t3=t[6::]
>>>t3
(“They”, 8, 9, ‘We’)
```

Packing and Unpacking Tuples

In Python, tuples are collections of elements which are separated by commas. It packs elements or value together so, this is called packing. In other way, it is called unpacking of a tuple of values into a variable.

In packing, we put values together into a new tuple while in unpacking we extract those values into a single variables.

Syntax variable1, variable2, ..., variableN = tuple_name

For example,

```
>>>tuple1=(34, “Hello”, 54, 89, “world”)
```

To unpack the tuple, take variables equivalent to number of elements in that tuple and write this

```
>>>a, b, c, d, e = tuple1
```

Now, print individual variable which display the element of tuple.

```
>>>print(a)
34
>>>print(b)
Hello
>>>print(c)
54
>>>print(d)
89
>>>print(e)
world
```

Built-in Functions

In Python, tuple has large number of built-in functions which perform various operations and make the task easier. Some of them are describe below

(i) len()

This function is used to count the number of elements that present in the tuple.

Syntax len(tuple_name)

For example,

```
>>>t1=(45, “The”, 67, 54, “That”, 90)
>>>len(t1)
6
>>>t2=(45, “The”, (67, 64), “That”, 90)
>>>len(t2)
5
```

(ii) count()

This function is used to calculate total occurrence of given element of tuple.

Syntax tuple_name.count(element)

For example,

```
>>>t1=(10, 20, 30, 40, 10, 50, 20, 60, 10)
>>>t1.count(10)
3
>>>t1.count(20)
2
>>>t1.count(50)
1
>>>t1.count(90)
0
```

(iii) any( )

This function returns True if atleast one element is present in the tuple, otherwise returns False.

Syntax any(tuple_name)

For example,

```
>>>t1=(3, 6, 4)
>>>any(t1)
True
>>>t2=(2,)
>>>any (t2)
True
>>>t3=()
>>>any (t3)
False
```

(iv) max()

This function is used to return the element with maximum value out of elements present in tuple.

Syntax max (tuple_name)

For example,

```
>>>t1=(34, 65, 77, 45, 87, 99, 90)
>>>max (t1)
99
>>>t=(34, 65, “The”)
>>>max(t)
Trackback (most recent call last):
File “<pyshell#5>”, line 1, in <module>
    max(t)
TypeError : unorderable types : str( )>int( )
>>>t=(‘a’, ‘f’, ‘F’, ‘u’)
>>>max(t)
‘u’
```

It will return max value of character using ASCII value.

(v) min()

This function is used to return with minimum value out of elements present in tuple.

Syntax min(tuple_name)

For example,

```
>>>t1=(34, 65, 77, 45, 87, 99, 90)
>>>min(t1)
34
>>>t=('a', 'f','F', 'u')
>>>min(t)
'F'
>>>t2=("Ansh", "Yash", "Sahil")
>>>min(t2)
'Ansh'
```

(vi) sorted()

This function is used to sort the given tuple in ascending order. But this method returns the elements in square brackets.

Syntax sorted(tuple_name)

For example,

```
>>>t=(32, 45, 25, 33, 55, 89, 47, 78)
>>>sorted(t)
[25, 32, 33, 45, 47, 55, 78, 89]
>>>t1=('a', 'r', 'T', 'R', 'e', 'E')
>>>sorted(t1)
['E', 'R', 'T', 'a', 'e' , 'r']
```

(vii) index()

It returns the index of first occurrence of element in the tuple.

Syntax tuple_name.index(element)

For example,

```
>>>t=(45, 89, 9, "The", 23, "That", "This", 25)
>>>t.index('The')
3
>>>t.index(9)
2
>>>t.index(25)
7
>>>t.index("Those")
Trackback (most recent call last):
File "<pyshell#8>", line 1, in <module>
    t.index("Those")
ValueError : tuple.index(x) : x not in tuple.
```

(viii) tuple()

This function is used to convert string and list into tuple.

Syntax tuple(list/string)

For example,

```
>>>name="NIHARIKA"
>>>tuple(name)
('N', 'I', 'H', 'A', 'R', 'I', 'K', 'A')
>>>num = [4, 6, 3, 7, 2, 4, 0, 7]
>>>tuple (num)
(4, 6, 3, 7, 2, 4, 0, 7)
```

(ix) sum()

This method is used to calculate the sum of elements of tuple. The elements of tuple must be integer.

Syntax sum(tuple_name)

For example,

```
>>>marks=(78, 98, 80, 65, 75, 80)
>>>print ("Total marks :", sum(marks))
Total marks : 476
```

(x) reversed()

This method allows us to process the items in a sequence in reverse order. It accepts a sequence and returns an iterator.

Syntax iterator = reversed (sequence)

Here, sequence is a tuple.

For example,

```
>>>marks=(78, 98, 80, 65, 75, 80)
>>>t=reversed(marks)
>>>t
<reversed object at 0X00F49F70>
>>>for i in t:
    print(i)
80
75
65
80
98
78
```

or

```
>>>tuple(reversed (marks))
(80, 75, 65, 80, 98, 78)
```

Deleting a Tuple

Tuples are immutable and cannot be deleted individual element from it but deleting tuple entirely is possible by using the keyword "del".

For example,

```
t1=(4, 6, 3, 7, 6, 5, 0)
print("Tuple is :", t1)
del(t1)
print("Tuple after deletion")
print(t1)
```

Output

```
Tuple is : (4, 6, 3, 7, 6, 5, 0)
Tuple after deletion
Trackback (most recent call last) :
File "<pyshell#6>", line 1, in <module>
    print(t1)
NameError: name 't1' is not defined
```

Chapter Practice

PART 1
Objective Questions

• Multiple Choice Questions

1. Which of the following is a collection of Python objects separated by commas and represent as (,)?
(a) List (b) Tuple
(c) Dictionary (d) String

Ans. (*b*) A tuple is a collection of Python objects separated by commas and represent as (,). Tuples are immutable by design which means they cannot be changed after creation. It holds a sequence of heterogeneous elements.
e.g. T = (3, 4, 7, 6)

2. What will be the output of the following Python code?
```
>>> a=(1,2,(4,5))
>>> b=(1,2,(3,4))
>>> a<b
```
(a) False
(b) True
(c) Error, < operator is not valid for tuples.
(d) Error, < operator is valid for tuples but not if there are sub-tuples.

Ans. (*a*) Since the first element in the sub-tuple of 'a' is larger that the first element in the sub-tuple of 'b', hence False is printed.

3. What is the output of the following code?
```
t1=(70, 56, 'Hello', 22, 2, 'Hi', 'The', 'World', 3)
print(t1 [2:4])
```
(a) (56, 'Hello')
(b) ('Hello', 22)
(c) ('Hello', 22,2)
(d) (56, 'Hello', 22)

Ans. (*b*) (:) is a slice operator, which returns the sub-part of any data type as string, list, tuple etc. Index number is started from 0, so the value of index number 2 is 'Hello' and this will display the elements till last index number – 1, i.e. (4 – 1=)3.
So, the correct output is ('Hello', 22).

4. Is the following Python code valid?
```
>>>tup1=(56, 25,36, 15)
>>> result=tup1.update(4,)
```
(a) Yes, tup1=(56, 25, 36, 15,4) and result=(56, 25, 36, 15,4)
(b) Yes, tup1=(56, 25, 36,15) and result=(56, 25, 36, 15,4)
(c) No, because tuples are immutable
(d) No, because wrong syntax for update() method

Ans. (*c*) Tuple does not have any update() attribute because it is immutable and cannot be changed after creation.

5. What is the output of following code?
```
t=(4,0, 'Hello', 90, 'Two', ('One', 45), 34, 2)
t1=t[1]+t[–2]
print(t1)
```
(a) 34 (b) 38
(c) Hello34 (d) 45

Ans. (*a*) Value of t[1] is 0 because index number is 1 and value of t[–2] is 34 because index number is started from –1 at the end point. t1 will store the sum of both values, i.e. 0+34=34.

6. What is the output of following code?
```
t=(1, 2, 'Hello', 'The', 3, 4)
print(max(t))
```
(a) 'Hello' (b) 4
(c) 'The' (d) Error

Ans. (*d*) This code will give an error because '>' (max) not supported between instances of 'str' and 'int'.

7. To create a tuple in Python, put all the elements in a
(a) () (b) []
(c) {} (d) <>

Ans. (*a*) To create a tuple in Python, put all the elements in a parentheses (), separated by commas. We can have tuple of same type of data items as well as mixed type of data items.
```
>>> t=()
>>> print(t)
()
```

8. Suppose t1 = (3, 4, 5,8, 2, 1)
Find the value of t1[3.5].
(a) 8 (b) 2
(c) 5 (d) Error

And. (*d*) It will give TypeError because tuple's index must be integers or slices, not float.

9. Suppose tuple t1 = (3, 4, 5, 6, 7, 8)
Choose the correct option for t1[6].
(a) 1 (b) 8
(c) None (d) Error

Ans. (*d*) It will give IndexError because tuple index is out of range. Its maximum index is 5 because index is started from 0 but in t1[6] asked about index number 6, so it will give an error.

10. Tuple packs elements or value together, so this is called
(a) pickling (b) unpacking
(c) packing (d) unpickling

Ans. (*c*) Tuple packs elements or value together, so this is called packing. In packing, we put values together into a new tuple while in unpacking we extract those values into a single variable.

11. Choose the correct output.

```
a = (2, 4, 3, 4)
b = (5, 8, 9)
t = a + b
print(t)
```

(a) (2, 4, 3, 4, 5, 8, 9) (b) (7, 12, 12, 4)
(c) (2, 10, 13, 4, 9) (d) Error

Ans. (*a*) To concatenate tuples, (+) operator is used in Python. This operator can easily add the whole of one tuple to other tuple and perform concatenation. This operator cannot add one tuple with other type as number or string, otherwise it will give error in such conditions.

12. Suppose tuple t1 = (4, 7, 3, 6, 8, 9)
Choose the correct option for t1[: 4].
(a) (3, 6, 8, 9) (b) (4, 7, 3, 6)
(c) (6) (d) (8, 9)

Ans. (*b*) To display a specific range of elements from the tuple, we use slicing operation. This operation in performed on tuples with the use of colon (:).
To display elements from beginning to a range, use [: index].
So, t1[: 4] will print the element from starting to index–1.

13. Given a tuple t1= (1, 2, 3, 4, 5, 6, 7, 8, 9). What will be the output of print (t1 [3 : 7 : 2])?
(a) (4, 5, 6, 7, 8) (b) (4, 5, 6, 7)
(c) (4, 5, 6) (d) (4, 6)

Ans. (*d*) t1[3 : 7 : 2] starts from index number 3 to index number 7 with gap of 2 elements.
In t1= (1, 2, 3, 4, 5, 6, 7, 8, 9), element of index number 3 is 4 and after two elements of gap, element is 6. So, output is (4, 6).

14. Given a tuple t1= (1, 2, 3, 4, 5). Identify the statement that will display an error.
(a) print (t1[3])
(b) t1[4] = 7
(c) print (len (t1))
(d) print (max (t1))

Ans. (*b*) t1[4] = 7 means updation which is not possible in tuple because tuple is immutable which cannot be changed after creation.

15. What is the output of following code?

```
T = (100)
print (T * 2)
```

(a) Syntax error (b) (200,)
(c) 200 (d) (100, 100)

Ans. (*c*) Tuple T contains a single element, so * is used as multiplication operator.
So, T * 2 = 100 * 2 = 200

16. What will be the output of the following Python code?

```
>>> a=(5,6)
>>> b=(2,6)
>>> c=a+b
>>> c
```

(a) (7,12)
(b) (5,6,2,6)
(c) Error as tuples are immutable
(d) None

Ans. (*b*) In the above piece of code, the values of the tuples are not being changed. Both the tuples are simply concatenated.

17. Choose the correct option.
(a) In Python, a tuple can contain only integers as its elements.
(b) In Python, a tuple can contain only strings as its elements.
(c) In Python, a tuple can contain both integers and strings as its elements.
(d) In Python, a tuple can contain either string or integer but not both at a time.

Ans. (*c*) In Python, a tuple can contain both integers and strings as its elements is the correct option.

• Case Based MCQs

18. Suppose that tuple
t1 = (“Hello”, (“am”, “an”), (“that”, “the”, “this”), “you”, “we”, “those”, “these”)
Based on the above information, answer the following questions.

(i) Choose the correct option for len(t1).
(a) 7 (b) 10
(c) None (d) Error

(ii) What is the output of following code?

```
print(t1[3 :5])
```

(a) (‘the’, ‘this’) (b) (‘am’, ‘an’)
(c) (‘you’, ‘we’) (d) (‘you’, ‘we’, ‘those’)

(iii) Identify the output of t1[5 :] + t1[2].
(a) (‘those’, ‘these’, ‘that’)
(b) (‘those’, ‘these’, ‘that’, ‘the’, ‘this’)
(c) (‘the’, ‘this’)
(d) Error

(iv) Identify the output of print (t1[6:]).
(a) ('these',)
(b) ('those')
(c) 'these'
(d) Error

(v) Find the correct output of print (t1[–3]*2).
(a) wewe
(b) youyou
(c) None
(d) IndexError

Ans. (i) (*a*) len() is used to count the number of elements that present in the tuple. Given tuple is a nested tuple, so ("am", "an") will considered as one element and ("that", "the", "this") will considered as one element. Then, this will give 7 as output.

(ii) (*c*) To display a specific range of elements from the tuple, we use slicing operation. This operation is performed on tuples with the use of colon (:).
t1[3 : 5] displays the element from index number 3, i.e. 'you' to index number **(5 – 1 =) 4** i.e. we. So, output will be ('you', 'we').

(iii) (*b*) To display elements from specific index till the end, use [index :], so t1[5:] will display the elements from index number 5 to till end i.e. ('those', 'these').
To access a particular element, use [index], so t1[2] will display the element of index number 2, i.e. ('that', 'the', 'this').
+ operator is used to concatenate the tuples.

(iv) (*a*) To display elements from specific index till the end, use [Index :].
So, t1[6:] will display the element from index number 6, i.e. 'these' till the end.

(v) (*a*) Index number – 3 represents the third element from end i.e. 'we'. * is the replication operator that can repeat the elements of the tuple.
So, 'we' will be repeat two times because * 2 is given.

PART 2
Subjective Questions

• Short Answer Type Questions

1. Distinguish between tuple and list.

Ans. Differences between tuple and list are as follows

Tuple	List
Elements of a tuple are immutable.	Elements of a list are mutable.
Tuple is declared in parenthesis ().	List is declared in square brackets [ ].
Tuples cannot be changed after creation.	Lists can be changed after creation.
Iterating over the elements of a tuple is fast.	Iterating over the elements of a list is slow.

2. Explain the mixed data types tuple with an example.

Ans. Mixed data types can be created to place different data types such as integers, strings, double etc into one tuple. *For example,*

```
tuple1 = ('English', 90, 'Rahul','Meerut', '99.5')
```

3. Observe the following tuple and answer the questions that follow.

```
t1 = (76, 56, 'Harish', 'Ansh', 98, (45, 34), 'Muskan')
```

(i) len(t1) (ii) t1[–6]
(iii) t1[3] (iv) t1[: 2]

Ans. (i) 7
(ii) 56
(iii) 'Ansh'
(iv) (76, 56)

4. Explain sum() method of tuple with an example.

Ans. sum() method is used to calculate the sum of elements of tuple. The elements of tuple must be integer.

Syntax sum(tuple_name)

For example,

```
>>>price=(100, 150, 95, 120, 80)
>>>sum(price)
```

Output

```
545
```

5. Observe the following tuples and answer the questions that follow.

```
t1 = (4, 7, 8, 9)
t2 = (0, 4, 3)
```

(i)
```
>>>t=t1+t2
>>>print(t)
```

(ii)
```
>>>t=t1*t2
>>>print(t)
```

Ans. (i) (4, 7, 8, 9, 0, 4, 3)
(ii) It gives TypeError because cannot multiply sequence by non-int of type 'tuple'.

6. Write a Python program to find maximum and minimum elements in a tuple.

Ans.
```
tuple1 = (23,45,–65,–45,20,45,65,– 24)
print("The tuple is:",tuple1)
min1 = tuple1.index (min(tuple1))
max1 = tuple1.index (max(tuple1))
print("Maximum element in the tuple is :", max(tuple1)," at index number ",max1)
print("Minimum element in the tuple is :", min(tuple1)," at index number ",min1)
```

7. What do you mean by membership operators in Python?

Ans. Membership operators are used to check whether a value/variable exists in the sequence like string, list, tuple etc. These operators return True or False as per conditions met.

Membership operators are of two types as:

(i) in operator

(ii) not in operator

8. Explain tuple slicing syntax with its parameters.

Ans. **Syntax** t=tuple_name[start : stop : step]

Here,

- **start** integer where the slicing of the object starts.
- **stop** integer until which the slicing takes place. The slicing stops at index stop–1.
- **step** integer value which determines the increment between each index for slicing.

9. Find the output of the given questions

```
t=(45, 76, 23, 'The', 89, ('This', 56), (23, 'That'),34)
```

(i) (t[4])

(ii) t[2:10:3]

(iii) t[2] + t [–1]

Ans. (i) 89

(ii) (23,('This', 56))

(iii) 57

10. Find the output of following code?

```
t = ('A', 'R', 'I', 'H', 'A', 'N', 'T')
for i in range (len(t)):
    print (t[i])
```

Ans. **Output**

```
A
R
I
H
A
N
T
```

11. What will be the output of following code?

```
tuple1 = (1,2,3,4,5)
tuple2 = (6,7,8,9)
for item in tuple1:
  if item in tuple2:
    print("overlapping")
  else:
    print("not overlapping")
```

Ans. **Output**

not overlapping

12. What will be the output of following code?

```
x=44
y=30
tuple = (45, 65, 30, 78, 512)
if(x not in tuple):
  print("x is NOT present in given tuple")
else :
  print ("x is present in given tuple")
if(y in tuple):
  print("y is present in given tuple")
else:
  print("y is NOT present in given tuple")
```

Ans. x is NOT present in given tuple

y is present in given tuple

13. What is the output of following code?

```
t1=(1, 2, 3, 4, 5)
print("Tuple is :", t1)
del(t1)
print("Tuple after deleting")
print(t1)
```

Ans. **Output**

```
Tuple is : (1, 2, 3, 4, 5)
Tuple after deleting
Trackback (most recent call last) :
File "<pyshell#6>", line 1, in <module>
    print(t1)
NameError: name 't1' is not defined
```

14. Find and write the output of the following Python code.

```
t=(4, (8, 0, 7))
t1=(4, 7, (2, 8))
print(t.count(0))
print(t[1][2])
print(t*2)
print(len(t1))
print(t1[2])
print(t+t1)
```

Ans. **Output**

```
0
7
(4, (8, 0, 7), 4, (8, 0, 7))
3
(2, 8)
(4, (8, 0, 7), 4, 7, (2, 8))
```

15. Identify the error, if any in the following code.

```
t1=(2, 3, 4, 'Hello', 6,9)
print (min(t1))
```

Ans. min() function is used in tuple to return with minimum value out of elements in tuple.

Given code has an error because min() will work only if elements in a tuple are of same data type, i.e. (2, 3, 4, 7, 6, 9).

16. Write a Python code to remove an element '2' from the following tuple.

```
tuple1 = (2, 5, 6, 9, 4)
```

Ans.

```
tuple1 = (2, 5, 6, 9, 4)
list1 = list (tuple1)
list1. remove (2)
tuple1 = tuple (list1)
print (tuple1)
```

17. Write a Python code to display all the elements of the following tuple except 'H'.

```
t = ('A', 'R', 'I', 'H', 'A', 'N', 'T')
```

Ans.
```
t = ('A', 'R', 'I', 'H', 'A', 'N', 'T')
t = t[0 : 3] + t[–3 : ]
print (t)
```
Output
```
('A', 'R', 'I', 'A', 'N', 'T')
```

18. TypeError occurs while statement 2 is running. Give reason. How can it be corrected?

```
>>> tuple1 = (5) #statement 1
>>> len(tuple1) #statement 2
```
[NCERT]

Ans. The 'statement 1' is creating a variable, tuple1 which is of 'int' data type. The 'statement 2' is checking for the length of the variable, but the argument passed is an 'int' data type. The len() function can return the length only when the object is a sequence or a collection. This is the reason for the type error.

The error can be corrected by adding one comma after '5' in statement 1, as this will create a tuple and as a tuple is a collection, len() function will not return an error.

The correct statement will be

```
>>> tuple1 = (5,)
>>> len(tuple1)
```

19. Prove with the help of an example that the variable is rebuilt in case of immutable data types. **[NCERT]**

Ans. When a variable is assigned to the immutable data type, the value of the variable cannot be changed in place.

Therefore, if we assign any other value of the variable, the interpreter creates a new memory location for that value and then points the variable to the new memory location. This is the same process in which we create a new variable. Thus, it can be said that the variable is rebuilt in case of immutable data types on every assignment.

Program to represent the same:

```
v = 20
print("Before: ",id(v))
v = 21
print("After: ",id(v))
```

Output

Before: 140705582623120

After: 140705582623152

It can be seen that the memory location a variable is pointing after the assignment is different. The variable is entirely new and it can be said that the variable is rebuilt.

• Long Answer Type Questions

20. Write the short note on following terms.

(i) Tuple

(ii) in operator

(iii) Equal to (= =) operator

(iv) Packing

Ans. (i) Tuple is a collection of Python objects separated by commas (,) and put the elements in parentheses ( ).

(ii) in operator is used to check, if a value exists in a sequence.

(iii) Equal to (= =) operator returns True if the values on either side of the operator are equal.

(iv) Tuples put all the elements or values together in a parentheses, is called packing.

21. Answer the following questions;

(i)
```
t1 = (25, 78, (45, (65, 89)), 90, (34, 8))
len (t1)
```

(ii)
```
t2 = (45,('The',78,('This'),67),'The',67, 67)
t2.count(67)
```

(iii)
```
t1 = (3,)
t2 = ()
t = t1 + t2
any(t)
```

(iv)
```
t3 = (87, 89, 56, 99, 75, 45, 100)
max(t3)
```

Ans. (i) 5 (ii) 2 (iii) True (iv) 100

22. Write a Python program to count the number of elements in a given range using traversal. Also, display its output.

Ans.
```
c = 0
l = 40
r = 80
tuple1 = (10, 20, 30, 40, 50, 40, 40, 60, 70)
for x in tuple1:
    if x >= l and x <= r:
        c += 1
print("Tuple:", tuple1)
print("Elements in a tuple:",c)
```

Output

Tuple : (10, 20, 30, 40, 50, 40, 40, 60, 70)

Elements in a tuple : 6

23. Write a Python program to find the common elements in two tuples.

Ans.
```
tuple1 = (45,87,56,–78,36,–12)
tuple2 = (65,32,45,–78,36,–75)
a_set = set(tuple1)
b_set = set(tuple2)
if (a_set & b_set):
    print("Common elements are:",a_set &
    b_set)
else:
    print("No common elements")
```

Output

Common elements are: {–78, 36, 45}

24. Write a Python program to calculate the sum and mean of the elements in a tuple.

Ans.
```
tuple1 = (23,45,20,45,65,24)
print("The tuple is:",tuple1)
sm=0
for i in range(len(tuple1)):
        sm=sm+tuple1[i]
mean=sm/num
print("SUM = ",sm)
print("MEAN = ",mean)
```
Output

The tuple is: (23, 45, 20, 45, 65, 24)
SUM = 222
MEAN = 44.4

25. Write a program to find the occurrence of a given element.

Ans.
```
t = (23,45,20,–45,65,24,–45,–23)
print("The tuple is:",t)
k=0
num=int(input("Enter the number to be
                    counted:"))
for j in t:
  if(j==num):
    k=k+1
print("Number",num,"is appear",k, "times.")
```
Output

The tuple is: (23, 45, 20, – 45, 65, 24, – **45**, – **23**)
Enter the number to be counted:45
Number 45 is appear 1 times.

26. Write a Python program to search an element with its index number.

Ans.
```
tuple1=(12,65,78,–63,–2,3,78,–12)
sm=0
x = int(input("Enter number to be searched:"))
found = False
for i in range(len(tuple1)):
        if(tuple1[i] == x):
            found = True
            print("%d found at %drd
                                        position"%(x,i))
            break
if(found == False):
        print("%d is not in tuple"%x)
```
Output

Enter number to be searched: – 63
– 63 found at 3rd position

27. Consider the following tuples, tuple1 and tuple2.
```
tuple1 = (23,1,45,67,45,9,55,45)
tuple2 = (100,200)
```
Find the output of the following statements.

(i) print(tuple1.index(45))
(ii) print(tuple1.count(45))
(iii) print(tuple1 + tuple2)
(iv) print(len(tuple2))
(v) print(max(tuple1))
(vi) print(min(tuple1))
(vii) print(sum(tuple2))
(viii) print(sorted(tuple1))
print(tuple1) **[NCERT]**

Ans.
(i) 2
(ii) 3
(iii) (23, 1,45, 67,45, 9, 55, 45, 100, 200)
(iv) 2
(v) 67
(vi) 1
(vii) 300
(viii) [1, 9, 23, 45, 45, 45, 55, 67]
(23,1,45,67,45,9,55,45)

28. Write a program to read email IDs of n number of students and store them in a tuple. Create two new tuples, one to store only the usernames from the email IDs and second to store domain names from the email IDs. Print all three tuples at the end of the program.

[**Hint** You may use the function split()] **[NCERT]**

Ans.
```
num = int(input("Enter number of students: "))
list1=[]
for i in range(num):
  email=input("Enter email: ")
  list1.append(email)
tuple1=tuple(list1)
username=[]
domain=[]
for i in tuple1:
  n,d = i.split("@")
  username.append(n)
  domain.append(d)
username = tuple(username)
domain = tuple(domain)
print("Names = ",username)
print("Domains = ",domain)
print("Tuple = ",tuple1)
```

29. A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. We can use the index operator [] to access an item in a tuple, where the index starts from 0.

So, a tuple having 6 elements will have indices from 0 to 5. Trying to access an index outside of the tuple index range(6,7,... in this example) will raise an IndexError.

(i) Which types of elements are stored in tuple?

(ii) What do you mean by nested tuples?

(iii) Write the syntax to create tuple from an existing sequence.

(iv) Observe the output of giving code.

```
>>>t=["T", "U", "P", "L", "E"]
>>>t2 = tuple(t)
>>>t2
```

(v) What is traversing a tuple in Python?

Ans. (i) Tuples hold a sequence of heterogeneous elements.

(ii) Nested tuples are tuple objects where the elements in the tuples can be tuples themselves.

(iii) `new_tuple_name=tuple(sequence)`

(iv) ('T', 'U', 'P', 'L', 'E')

(v) Traversing a tuple is a technique to access an individual element of that tuple.

Chapter Test

Multiple Choice Questons

1. Consider the declaration obj = (2, 'Hello', 3, 4). What will be the data type of obj?

(a) List (b) Tuple

(c) Dictionary (d) String

2. What is the output of followig code?

```
t1=(1, 2, 3, 4, 5, 6, 7, 8)
print (t1[2 : 4])
```

(a) (3, 4) (b) (2, 3)

(c) (4, 5, 6) (d) (3, 4, 5)

3. Choose the correct option with respect to Python.

(a) Both tuples and lists are immutable.

(b) Tuples are immutable while lists are mutable.

(c) Both tuples and lists are mutable.

(d) Tuples are mutable while lists are immutable.

4. Which of the following options will not result in an error when performed on tuples in Python where tupl=(5,2,7,0,3)?

(a) tupl[1]=2

(b) tupl.append(2)

(c) tupl1=tupl+tupl

(d) tupl.sort()

5. What will be the output of the following Python code?

```
>>>my_tuple = (10, 20, 30, 40)
>>>my_tuple.append((50, 60))
>>>print (len(my_tuple))
```

(a) 1 (b) 6

(c) 4 (d) Error

6. Is the following Python code valid?

```
>>> a,b=1,2,3
```

(a) Yes, this is an example of tuple unpacking, where a=1 and b=2.

(b) Yes, this is an example of tuple unpacking, where a=(1,2) and b=3.

(c) No, too many values to unpack.

(d) Yes, this is an example of tuple unpacking, where a=1 and b=(2,3).

Short Answer Type Questions

7. Observe the following tuple and answer the questions that follow

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

(i) t[–3]

(ii) t[: 2]

8. Explain the any () method of tuple with an example.

9. Suppose the tuple t1 = (2, 3, 4, 7, 1, 6). Find

(i) t1.index(4)

(ii) t1.count(4)

10. Suppose the tuple t1 = (2, 3, 2, 2, 3, 4, 6, 7).

(i) count (t1) (ii) len(t1)

11. Observe the given tuples and answers the questions

```
t1 = (1, 2, 3, 4)
t2 = (5, 6, 7)
```

(i)
```
>>> t = t1 + t2
>>> print(t)
```

(ii)
```
>>> t = t1 * t2
>>> print(t)
```

12. Consider the tuple t = (2,3, 'Hello', 2, 5, 9) and find out the error if any in following code

```
tuple1 = t + 5
print(tuple1)
```

13. Compare the tuple and write the output.

(i)
```
t1 = (4, 5, 6, 9)
t2 = (6, 9, 5, 6)
print(t1 < t2)
```

(ii)
```
t1 = (4, 5, 6, 9)
t2 = (4.0, 5.0, 6.0, 9.0)
print(t1 == t2)
```

Long Answer Type Questions

14. Write a Python program to count the positive numbers and negative numbers in a tuple.

15. Write the most appropriate tuple methods for the following conditions.

(i) To count the number of elements in a tuple.

(ii) Calculate total occurence of given element.

(iii) Returns the element with maximum value.

(iv) Returns the element with minimum value.

(v) To sort the given tuple in ascending order.

(vi) Returns true if atleast one element is present in the tuple.

(vii) Returns the index of first occurrence of element.

(viii) Converts string and list into tuple.

16. Write a Python program to test if a variable is a list or tuple.

17. Write a Python program to sort a list of tuples by the second Item.

18. Write a Python program to sort a list of tuples alphabetically.

Answers

Multiple Choice Questions

1. *(b)* ***2.*** *(a)* ***3.*** *(b)* ***4.*** *(c)* ***5.*** *(d)* ***6.*** *(c)*