Chapter 11

A.

1. B
2. B
3. C
4. B
5. C
6. C
7. D
8. D
9. C
10. B

B.

1. False
2. True
3. False
4. True
5. True
6. True
7. True
8. True
9. True
10. True
11. True
12. True
13. False
14. True
15. True

C.

1. Tuples are defined in the same way as that of the list. Except elements of the tuples are enclosed in parenthesis instead of square bracket. The value stored in a tuple can be of any type and they are indexed by integers. Once the tuple is created it cannot be changed i.e. tuples are immutable.

Example: Defining a Tuple

```
T1 = ()    #Creates an Empty Tuple
T2 = (12,34,56,90)  #Create Tuple with 4 elements
T3 = ('a','b','c','d','e') #Create Tuple of 5 characters
```

2. The basic in build functions supported by tuples are len(), max(), min(), sum(), index(x), count(x), len().

3. Difference between List and tuples is as follows

List	Tuples
Elements of list are enclosed within square brackets separated by commas.	Elements of tuples are enclosed within parenthesis separated by commas.
List are mutable.	Tuples are immutable.
Example: List are created as follows L1 = [1,2,3,4]	**Example:** Tuples are created as follows T1=(1,2,3,4)

4. Single element using tuple is created is as follows
   T = (1,)

5. The in build functions supported by tuple are as follows
   len() – Returns the numbers of elements in tuple
   max() – Return the element with the greatest value
   min() – Returns the element with minimum value
   sum() – Returns the sum of all the elements of tuple
   index(x) – Returns the index of element x
   count(x) – Returns the number of occurrences of x

6. The value stored in a tuple can be of any type and they are indexed by integers. As tuples are just like lists, therefore the indexing and slicing of tuples is similar to lists. As usual index operator is used to access the element of tuple.

Example:

```
>>> t=('P','Y','T','H','O','N')      #Create Tuple
>>>t                                  #Print Tuple
>>>('P', 'Y', 'T', 'H', 'O', 'N')
>>> t[0]
'P'
>>> t[5]
'N'
>>> t[-1]
'N'
>>> t[-6]
```

```
'P'

#Slicing Example
>>> t[0:3]
('p', 'y', 't')
>>> t[::-1]
('n', 'o', 'h', 't', 'y', 'p')

>>> t[:-1]
('p', 'y', 't', 'h', 'o')
```

7. The index [] operator is used to access the elements of tuple.

8. Yes, we can pass variable as well as variable number of parameter to a function. Following program demonstrate passing variable number of parameter to a function.

```
def create_tup(*args):
    print(args)
create_tup(1,2,4,23)

Output
(1,2,4,23)
```

9.

a. 11
b. (13, 'Sachin', 14)
c. 'Hii'
d. 14
e. 14
f. 'i'
g. 'i'
h. (11,)
i. (11, 12, (13, 'Sachin', 14))
j. 4
k. False
l. False
m. Error

10. The zip() function take items in sequence from a number of collections to make a list of tuples.

Example:

```
>>> A1=[1,2,3]
>>> A2="XYZ"
```

```
>>> list(zip(A1,A2))              #Zip List A1 and A2
[(1, 'X'), (2, 'Y'), (3, 'Z')]
```

11. The ***** operator with zip function is used to unpacks the sequence into positional arguments.

Example:

```
X=[("APPLE",50000),("DELL",30000)] #List of tuples
Laptop,Prize=zip(*X)  # Unpacking Values
print(Laptop)

Output
('APPLE', 'DELL')
```

12. In python a dictionary is a collection that stores the values along with the keys. The sequence of key and value pairs is separated by commas. All entries are enclosed in curly braces **{** and **}**. A colon separates a key and its value.

Example:

```
>>> d={'1':'0001','2':'0010','3':'0010','4':'0100'}
 >>> d
 {'3': '0010', '2': '0010', '4': '0100', '1': '0001'}
```

13.

a. dict_keys(['Sahil', 'Abhijeet'])
b. dict_values([90, 65])
c. 2
d.
e. One new element is added to the dictionary.

```
>>> Grades
{'Sahil': 90, 'Kuruss': 99, 'Abhijeet': 65}
 >>> Grades
```

f. Key having name 'Abhijeet' gets deleted from the dictionary.

```
>>> Grades
{'Sahil': 90, 'Kuruss': 99}
 {'Sahil': 90, 'Kuruss': 99}
```

g. dict_items([('Sahil', 90), ('Kuruss', 99)])

14.

a. True
b. True
c. True
d. False
e. 4
f. 40
g. 100

15.

a. {'E', 'D', 'B', 'C', 'A'}
b. {'C'}
c. {'B', 'A'}
d. {'B', 'E', 'D', 'A'}
e. {'B', 'E', 'D', 'A'}
f. {'E', 'D', 'B', 'C', 'A'}
g. {'C'}

16.

a. (10, 34, 22, 87, 90)
b. 10
c. (10, 34, 22, 87)
d. (10, 34, 22, 87)

Programming Assignments

1.

```
def oddTuples(aTup):  #Function with tuple  as a argument
    rTup = ()         #Initially the output tuple rTup is empty
    index = 0
    while index < len(aTup):
        rTup += (aTup[index],)
        index += 2        #index increased by 2
    return rTup
Output_Tuple=('Hello', 'Are', 'You', 'Loving', 'Python?')
print(oddTuples(Output_Tuple))
```

Output

```
('Hello', 'You', 'Python?')
```

2.

```
def how_many(aDict):
```

```
    result = 0
    for value in aDict.values():
   #Since all the values of aDict are lists, aDict.values() will
        #  be a list of lists
        print(len(value))
        result += len(value)
    return result
animals = {'L':['Lion'],'D':['Donkey'],'E':['Elephant']}
print(how_many(animals))
```

Output

```
3
```

3.

```
def how_many(aDict):
    result = None
    biggestValue = 0
    for key in aDict.keys():
        if len(aDict[key]) >= biggestValue:
            result = key
            biggestValue = len(aDict[key])
    return result
animals = {'L':['Lion'],'D':['Donkey','Deer'],'E':['Elephant']}
print(how_many(animals))
```

Output

```
D
```

4.

```
def Count_Each_vowel(S):
    V = 'AEIOU'
    S= S.upper()
    D = {}.fromkeys(V,0)# make a dictionary with each vowel a key
and value 0
    for c in S:
        if c in D:
            D[c] += 1
    return D
H=Count_Each_vowel("AAPPLE")
print(H)
```

Output

```
{'A': 2, 'O': 0, 'U': 0, 'E': 1, 'I': 0}
```