Assignment/Task 2:

1. **Is a list mutable?**

Ans.

Yes, list is mutable data type.

2. **Does a list need to be homogeneous?**

Ans.

No, list do not need to be homogeneous because list can be a collection of heterogeneous or homogeneous.

3. **What is the difference between a list and a tuple.**

Ans.

List	Tuple
✓ The literal syntax of lists is shown by square braces **"[]"**.	✓ The literal syntax of tuple is shown by small braces or parentheses **"()"**.
✓ List has variable length.	✓ Tuple has fixed length.
✓ List is mutable data type.	✓ Tuple is immutable data type.
✓ List has more functionality or in-built functions than tuple.	✓ Tuple has less functionality or in-built functions than list.

4. **How to find the number of elements in the list?**

Ans.

To get the number of elements in a list is to use the Python built-in function **`len()`**.

len() returns the length of the list, regardless of the types of elements in it.

5. **How to check whether the list is empty or not?**

Ans.

Using len() function:

Example:

```
py_list = []

if len(py_list) :
        print("List is not empty")
else:
        print("List is empty")
```

Output:

List is empty

6. **How to find the first and last element of the list?**

Ans.

lst = [1,2,3,4,5]
To find First element : lst[0].
To find Last element : lst[-1].

7. **How to find the largest and lowest value in the list?**

Ans.

Use **max()** and **min()** to find the maximum and minimum of a list. Call **max (obj)** and **min (obj)** with a list as **obj** to return the maximum and minimum of the list.

Example:

```
a_list = [5, 2, 7, 6, 3, 1, 9]
print (max(a_list))
```

OUTPUT: 9

```
print(min(a_list))
```

OUTPUT: 1

8. **How to access elements of the list?**

Ans.

Using Index value we can access the elements in list.

9. **Remove elements in a list before a specific index**

Ans.

```
a=[1,2,3,4,5,5]

del a[0:3]
print(a)
```

Output:
[4, 5, 5]

10. **Remove elements in a list between 2 indices**

Ans.

```
a =[1,2,3,4,5,6,7,8,9]
del a[ : : 2]
print (a)
```

11. **Return every 2nd element in a list between 2 indices**

Ans.

```
a =[1,2,3,4,5,6,7,8,9]
a[ : : 2]
print (a)
```

12. **Get the first element from each nested list in a list**

Ans.

```
def Extract(lst):
        return [item[0] for item in lst]

lst = [[1, 2], [3, 4, 5], [6, 7, 8, 9]]
print(Extract(lst))
```

13. **How to modify elements of the list?**

Ans.

We use following syntax to modify a list item in python programming. To change a list element we use, list name followed by the index position inside the square brackets ([]), and then provides the new value using the Equal sign.

list_name[index_position] = "New Value"

Example:

```
languages = ["Perl", "PHP", "Java", "C", "JavaScript", "C++"]
print(languages)

languages[0] = "Python 3"
print(languages)
```

In the above example we have a python list called languages which contains the list of some programming languages. Next, we replace the index 0 of the list(First element) which is Perl with the new value “Python 3” (So the “Perl” will replace by “Python 3”). The above code will output.

['Perl', 'PHP', 'Java', 'C', 'JavaScript', 'C++']

['Python 3', 'PHP', 'Java', 'C', 'JavaScript', 'C++']

14. **How to concatenate two lists?**

Ans.

```
test_list3 = [1, 4, 5, 6, 5]
test_list4 = [3, 5, 7, 2, 5]
```

 # using + operator to concat

```
test_list3 = test_list3 + test_list4
```

 # Printing concatenated list

```
print ("Concatenated list using + : " + str(test_list3))
```

Output:

Concatenated list using + : [1, 4, 5, 6, 5, 3, 5, 7, 2, 5]

15. **How to add two lists element-wise in python?**

Ans.

```
lst1 = [1, 2, 3]
lst2 = [4, 5, 6]
sum = []

for ( a, b ) in zip [ lst1, lst2 ] :

        sum.append( a+b )

print (sum)
```

16. **Difference between del and clear?**

Ans.

i) **del()** will delete the entire list and you cannot access that list again whereas **clear()** will only clear the entire list but not you can still access the list.

ii) To remove items by index or slice we can use **del method** in python but it is not possible in **clear method.**

17. **Difference between remove and pop?**

Ans.

i) **remove()** is an inbuilt function in Python programming language that removes a given object from the list. It removes the object from the list. It does not return any value. **list.remove(index)**

ii) **pop()** method: The pop() method removes the item at the given index from the list and returns the removed item. **list.pop(index)**

18. **Difference between append and extend?**

Ans.

i) **append()** adds one new element to the end of the list. If you append a list to another list that list you append becomes a single element at the end of the first list. list1.append(list2).

ii) **extend (enumerable)** : It extends the list by appending elements from another enumerable. list1.extend(list2).

19. **Difference between indexing and Slicing?**

Ans.

i) Indexing is used to obtain individual elements whereas slicing is used to obtain sequence of elements.
ii) Attempting to use an index that is too large will result in an IndexError whereas in slicing Out of range indexes are handled gracefully.
iii) Indexing returns one item whereas slicing returns new list.
iv) Indexing starts from 0. Negative indexing starts from -1. Whereas in slicing we can specify range of indexes.

20. **Difference between sort and sorted?**

Ans.

i) The sort method will sort and modify the original list itself whereas the sorted method will sort the list and return a new list. It will not modify original list.
ii) The return type of sort method is None whereas the return type of sorted method is new sorted list.
iii) The sort method is supported only by lists whereas sorted function is supported by other iterables (string, tuples, dictionary).

21. **Difference between reverse and reversed?**

Ans.

i) The reverse method is used to reverse the list. It will reverse the original list whereas in reversed() method returns an iterator that accesses the given sequence in the reverse order.
ii) The return type of reverse method is None whereas the return type of reversed method is reversed iterator object.
iv) The reverse method is supported only by lists whereas reversed function is supported by other iterables (string, tuples, dictionary).

22. **Difference between copy and deepcopy?**

Ans.

A shallow copy constructs a new compound object and then inserts references into it to the objects found in the original. A deep copy constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original.

23. **How to remove duplicate elements in the list?**

Ans.

Set don't contain duplicate elements in python. We can convert the list to set by using the set() constructor and converts back to the list using the list() constructor.

Example:

```
num1=[1,2,3,1,2,3,4,3,2,5,4,3]
num2=set(num1)
print (list(num2))
```

Output : **[1, 2, 3, 4, 5]**

24. **How to find an index of an element in the python list?**

Ans.

index(i) method will return the index of the first item whose value is equal to i.

Example:

```
n=[1,2,3,4,5]
print (n.index(2))
```

Output: 1

25. **How to find the occurrences of an element in the python list?**

Ans.

count() method will return the number of occurrences of a particular element mentioned.

Example:

```
a1=[1,2,3,4,1,5,6,1]
print (a1.count(1))
```

Output: 3

26. **How to insert an item at a given position?**

Ans.

We can insert an item at a given position using the **insert method.**

list.insert(i,x)

i-index(at which position we have to insert)
x-element(what element needs to be inserted)

Example:

```
n=[1,2,3,4,5]

#inserting element 99 at index 1

n.insert(1,99)

print (n)
```

Output:

[1, 99, 2, 3, 4, 5]

27. **How to check if an item is in the list?**

Ans.

Can be done by **membership operator.**

in and **not in** operations are used only for simple containment testing**.**

Example:

```
n=[1,2,3,4,5]

if 1 in n:
        print ("yes")

if 2 not in n:
        print ("no")
```

Output : **yes**

28. **How to flatten a list in python?**

Ans.

Flattening a list of lists entails converting a 2D list into a 1D list by un-nesting each list item stored in the list of lists - i.e., converting [[1, 2, 3], [4, 5, 6], [7, 8, 9]] into [1, 2, 3, 4, 5, 6, 7, 8, 9].

Example:

1st Method:

```
lst1= [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
n2 = []

for num1 in lst1:

        for num2 in num1:

                n2.append(num2)
print(n2)
```

2nd Method: Using List Comprehension

```
lst1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

n = [num2 for num1 in lst1 for num2 in num1]

print(n)
```

29. **How to convert python list to other data structures like set, tuple, dictionary?**

Ans.

i) **Converting a list to set using set() constructor.**
Set don't contain duplicate elements. So if we convert a list to set, duplicates will be removed.

Example:

```
n1=[1,2,3,4,2,3]
s1=set(n1)
print (s1)
```

Output: {1, 2, 3, 4}

ii) **Converting a list to a tuple using a tuple() constructor.**

Example:

```
n1= [1,2,3,4,2,3]
t1=tuple(n1)
print (t1)
```

Output: (1, 2, 3, 4, 2, 3)

iii) **Converting a list to a dictionary.**

Converting a list of tuples to a dictionary using dict() constructor.

Example:

```
n1 = [(0,1),(2,3),(4,5)]
d1 = dict(n1)
print (d1)
```

Output: {0: 1, 2: 3, 4: 5}

30. **How to apply a function to all items in the list?**

Ans.

We can use a **map()** function to apply a function to all items in the list.

Syntax:

map(*function*, *iterable*, ...)

Example: Applying a square function to list using a map() function.

```
n1=[1,2,3,4,5]

def square(n):
return n*n

n2=map(square,n1)
print (n2)

#output:      <map object at 0x00D6EC10>
#converting map object to list using list() constructor.

print (list(n2))

#Output:      [1, 4, 9, 16, 25]
```

31. **How to filter the elements based on a function in a python list?**

Ans.

We can use **filter()** to filter the elements in the list based on some function.

Syntax:

filter(*function*, *iterable*)

Example: Filter the odd numbers in the given list.

```
n1=[1,2,3,4,5]

def odd(n):
        if n%2!=0:
                return True

n2=filter(odd,n1)
print (n2)

#output:<filter object at 0x00F3EC10>
#converting filter object to list using list() constructor.

print (list(n2))

#Output:      [1,3,5]
```

32. **How python lists are stored in memory?**

Ans.

Instead of storing values in memory space, Python lists store references/pointers to the values (object).