```
# how to take multiple values input in list.
n = int(input('How many items? '))
items = []
for i in range(n):
    item = int(input('Enter values: '))
    items.append(item)
print(items)
```

```
How many items?  6
Enter values:  34
Enter values:  67
Enter values:  89
Enter values:  23
Enter values:  78
Enter values:  88
```

```
[34, 67, 89, 23, 78, 88]
```

```
items = input("Enter items separated by space: ").split()
print(items)
```

```
Enter items separated by space:  apple banana mango
```

```
['apple', 'banana', 'mango']
```

```
numbers = list(map(int, input("Enter numbers separated by space:
").split()))
print(numbers)
```

```
Enter numbers separated by space:  1 2 3 4 5 7 8 9 10
```

```
[1, 2, 3, 4, 5, 7, 8, 9, 10]
```

```
# remove duplicate values in list
# i/p: [12,34,12,67,89,23,34,34,34,78,22,65]
# o/p: [12,34,67,89,23,78,22,65]

li = [12,34,12,67,89,23,34,34,34,78,22,65]
new = []
for i in li:  # i=12
    if i not in new:
        new.append(i)
print(new)
```

```
[12, 34, 67, 89, 23, 78, 22, 65]
```

Task -01

print unique values in list

i/p: [12,45,67,77,88,99,23,56,78,12,45,67,78]

o/p:- [77,88,99,23,56]

Tuple

```
- A tuple in python is an ordered,immutable collection of items
* Tuple is immutable (i.e we can't the values once we assigned)
- We can store hetrogeneous data
- Index starts with '0'
- It can allow the duplicates
- Represented by (),values can be separated by ,
```

```
# how to create a empty tuple
t = ()
print(t,type(t))
```

```
() <class 'tuple'>
```

```
t1 = (56,78,22,"a","i","p",6.7,5.8)
print(t1,type(t1))
```

```
(56, 78, 22, 'a', 'i', 'p', 6.7, 5.8) <class 'tuple'>
```

- Tuples are generally faster and more memory-efficient than lists because of their immutability

```
# builit-in types in tuple
t = (23,56,78,8.9,6,8.9)
print(len(t),sum(t))
print(min(t),max(t))
print(sorted(t))
```

```
6 180.8
6 78
[6, 8.9, 8.9, 23, 56, 78]
```

```
print(t[0])
print(t[1],t[2])
```

```
print(t[::-1])
print(t[0:3])
```

```
23
56 78
(8.9, 6, 8.9, 78, 56, 23)
(23, 56, 78)
```

```
t[1]=65
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 t[1]=65

TypeError: 'tuple' object does not support item assignment
```

```
print(dir(tuple),end=' ')
```

```
['__add__', '__class__', '__class_getitem__', '__contains__',
'__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__getnewargs__', '__getstate__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__',
'__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

```
# count()
t2 = (56,78,12,78,90,56,78,23,23,2,3,44)
print(t2.count(23))
print(t2.count(44))
```

```
2
1
```

```
# index()
print(t2.index(56))
print(t2.index(78))
```

```
0
1
```

```
t2 = (56,78,12,78,90,56,78,23,23,2,3,44)
for i in t2:
    print(i,end=' ')
```

```
56 78 12 78 90 56 78 23 23 2 3 44
```

```
t2 = (56,78,12,78,90,56,78,23,23,2,3,44)
for i in t2:
```

```
    if(i%3==0):
        print(i,end=' ')
```

```
78 12 78 90 78 3
```

Packing

```
- Packing means grouping multiple values together into a single
variable, typically as a tuple (or list).

You are "packing" three values into one tuple named person.

Note: Parentheses are optional during packing —
    Python treats the values as a tuple automatically.
```

```
# Packing values into a tuple
person = "Alice", 30, "Engineer"
print(person)         # Output: ('Alice', 30, 'Engineer')
print(type(person))   # Output: <class 'tuple'>
```

```
('Alice', 30, 'Engineer')
<class 'tuple'>
```

Unpacking

```
Unpacking means extracting values from a tuple, list,
or other iterable and assigning them to individual variables.
```

```
# A tuple with packed values
person = ("Alice", 30, "Engineer")
# Unpacking into separate variables
name, age, profession = person

print(name)       # Alice
print(age)        # 30
print(profession) # Engineer
```

```
Alice
30
Engineer
```

```
# Here, each variable (name, age, profession) is assigned a value from
the person tuple.
```

Dictionary

```
- A dictionary is a collection which is ordered,changeable
- Dictionary is a key value pair
- Keys doesn't allow the duplicates
- values can allow the duplicates
- Dictionary is mutable
- represented by {},keys and values can be separated by :
```

```
# how to create a empty dictionary
d = {}
print(d,type(d))
```

```
{} <class 'dict'>
```

```
d1 = {8:90,6:89,5:44,"a":"n","k":8.6,9:8.4}
print(d1,type(d1))
```

```
{8: 90, 6: 89, 5: 44, 'a': 'n', 'k': 8.6, 9: 8.4} <class 'dict'>
```

```
# Built in methods in dictionary
d2 = {8:90,6:89,5:44,9.8:8.6,9:8.4}
print(len(d2))
print(min(d2),max(d2))
print(sum(d2),sorted(d2))
```

```
5
5 9.8
37.8 [5, 6, 8, 9, 9.8]
```

```
print(d2[5])
print(d2[9.8],d2[9])
```

```
44
8.6 8.4
```

```
# nested dictionary
d2 = {8:90,6:89,5:44,7:{6:8,5:89},9.8:8.6,9:8.4}
print(d2)
```

```
{8: 90, 6: 89, 5: 44, 7: {6: 8, 5: 89}, 9.8: 8.6, 9: 8.4}
```

```
print(d2[7])
```

```
{6: 8, 5: 89}
```

```
# keys doesn't allow the duplicates
# nested dictionary
d3 = {5:78,3:67,5:89,9:45,5:78,5:46,5:900}
print(d3)
```

```
{5: 900, 3: 67, 9: 45}
```

```
# values can allow the duplicates
f = {5:78,3:67,15:78,5:900,7:67}
print(f)
```

```
{5: 900, 3: 67, 15: 78, 7: 67}
```

```
s = {}
print(s,type(s))
```

```
{} <class 'dict'>
```

```
s = {1,2,3,4}
print(s,type(s))
```

```
{1, 2, 3, 4} <class 'set'>
```

```
A={9,4,5,6,10,23,45,1,9,8,7,10,34,56,23,12,4,5,10}
A
```

```
{1, 4, 5, 6, 7, 8, 9, 10, 12, 23, 34, 45, 56}
```

```
A=[9,4,5,6,10,23,45,1,9,8,7,10,34,56,23,12,4,5,10]
print(A)
print(type(A))
print(set(A))
```

```
[9, 4, 5, 6, 10, 23, 45, 1, 9, 8, 7, 10, 34, 56, 23, 12, 4, 5, 10]
<class 'list'>
{1, 34, 4, 5, 6, 7, 8, 9, 10, 12, 45, 23, 56}
```

```
A.pop()
```

```
10
```

```
A
```

```
[9, 4, 5, 6, 10, 23, 45, 1, 9, 8, 7, 10, 34, 56, 23, 12, 4, 5]
```

```
A.pop()
```

```
5
```

```
A
```

```
[9, 4, 5, 6, 10, 23, 45, 1, 9, 8, 7, 10, 34, 56, 23, 12, 4]
```

```
print(A)
```

```
[9, 4, 5, 6, 10, 23, 45, 1, 9, 8, 7, 10, 34, 56, 23, 12, 4]
```

```
A.pop()
```

```
4
```

```
print(A)
[9, 4, 5, 6, 10, 23, 45, 1, 9, 8, 7, 10, 34, 56, 23, 12]
A.pop()
12
print(A)
[9, 4, 5, 6, 10, 23, 45, 1, 9, 8, 7, 10, 34, 56, 23]
A.pop()
23
A
[9, 4, 5, 6, 10, 23, 45, 1, 9, 8, 7, 10, 34, 56]
```

TASK -02

I/P: [4,5,6,8,9,2,7,1]

O/P: {25,81,49,1}