19 SEPTEMBER PYTHON TUTORIAL SOLUTION

(Adarsh Tiwari)

Problem 1: Remove Duplicates from a List

Approach:

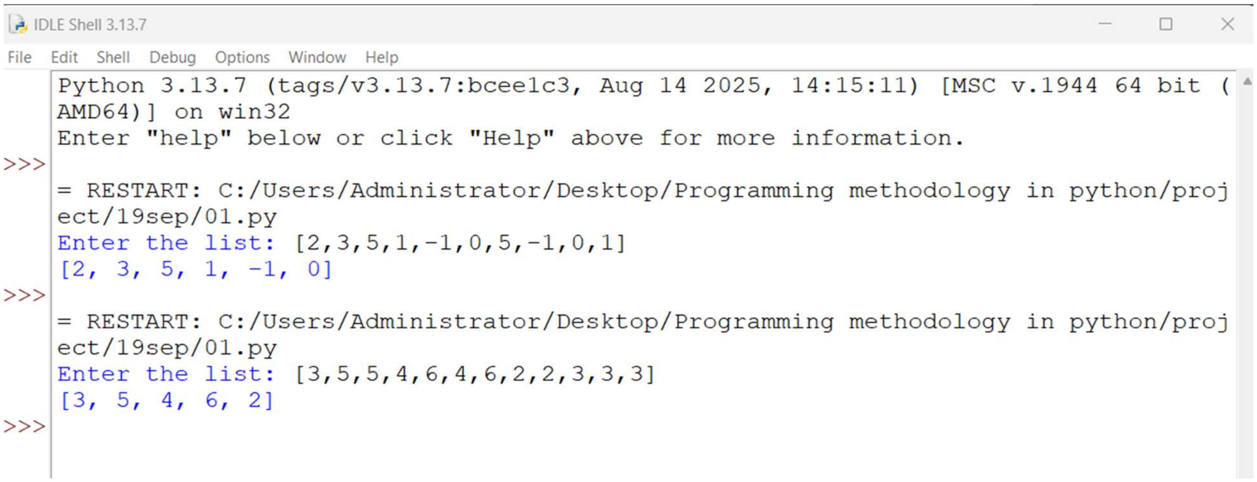
1. Take the input list from the user.
2. Create a new empty list to store unique elements.
3. Loop through the original list and add elements to the new list only if they are not already present.
4. Print the new list containing only unique elements in the same order.

Code:

```
l = eval(input('Enter the list: '))
nl = []
for i in l:
    if i not in nl:
        nl.append(i)

print(nl)
```

Sample Run:



```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
ect/19sep/01.py
Enter the list: [2,3,5,1,-1,0,5,-1,0,1]
[2, 3, 5, 1, -1, 0]
>>>
= RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
ect/19sep/01.py
Enter the list: [3,5,5,4,6,4,6,2,2,3,3,3]
[3, 5, 4, 6, 2]
>>>
```

Problem 2: Count Word Frequency in a List

Approach:

1. Take the input list from the user.
2. Create an empty dictionary to store word counts.
3. Traverse the list, and for each word increase its count in the dictionary.
4. Finally, print the dictionary with words and their frequencies.

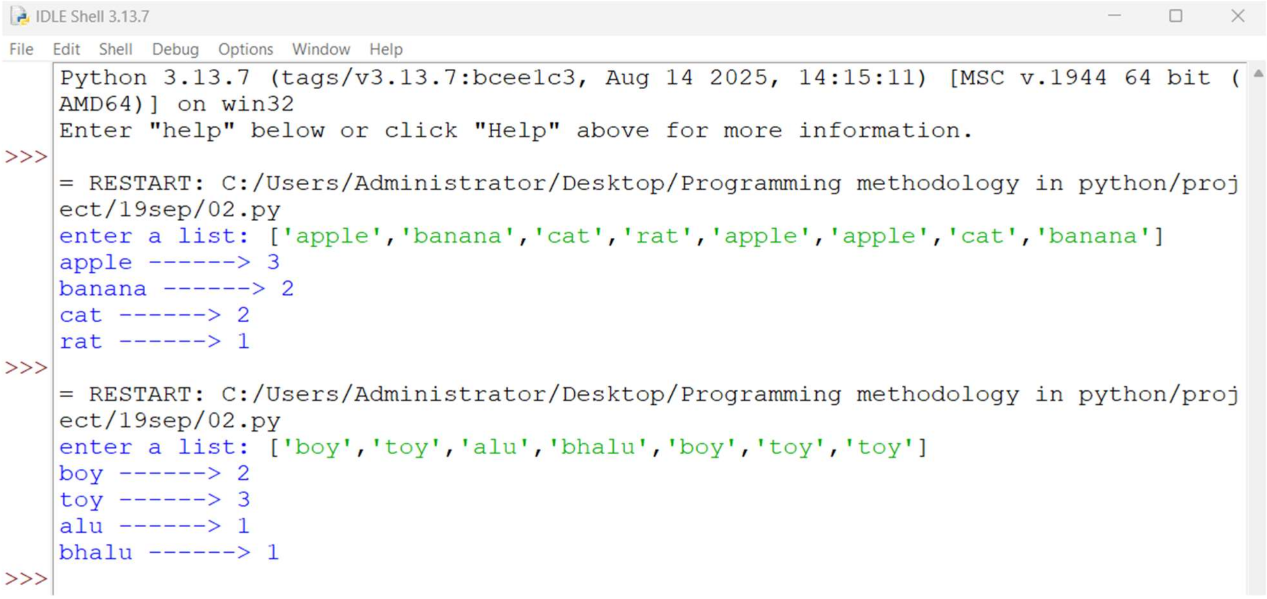
Code:

```
l = eval(input('enter a list: '))
d = {}

for i in l:
    if i in d:
        d[i] += 1
    else:
        d[i] = 1

for k,v in d.items():
    print(k,'------>',v)
```

Sample Run:



```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
ect/19sep/02.py
enter a list: ['apple','banana','cat','rat','apple','apple','cat','banana']
apple ------> 3
banana ------> 2
cat ------> 2
rat ------> 1
>>>
= RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
ect/19sep/02.py
enter a list: ['boy','toy','alu','bhalu','boy','toy','toy']
boy ------> 2
toy ------> 3
alu ------> 1
bhalu ------> 1
>>>
```

Problem 3: Find Maximum Number Without Using max()

Approach:

1. Take the input list from the user.
2. Assume the first element as the maximum.
3. Traverse through the list, compare each element with the current maximum.
4. If a larger element is found, update the maximum.
5. Print the final maximum value.

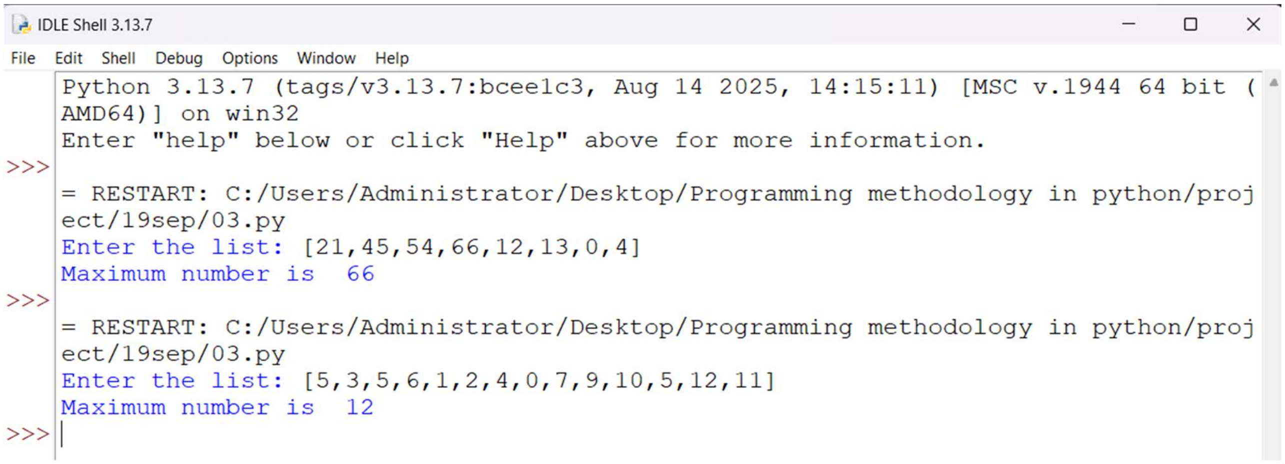
Code:

```
l = eval(input('Enter the list: '))

n_max = l[0]
for i in l:
    if i > n_max:
        n_max = i

print('Maximum number is ', n_max)
```

Sample Run:



```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
ect/19sep/03.py
Enter the list: [21,45,54,66,12,13,0,4]
Maximum number is  66
>>>
= RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
ect/19sep/03.py
Enter the list: [5,3,5,6,1,2,4,0,7,9,10,5,12,11]
Maximum number is  12
>>>
```

Problem 4: Print Lines Containing 'error' from a File

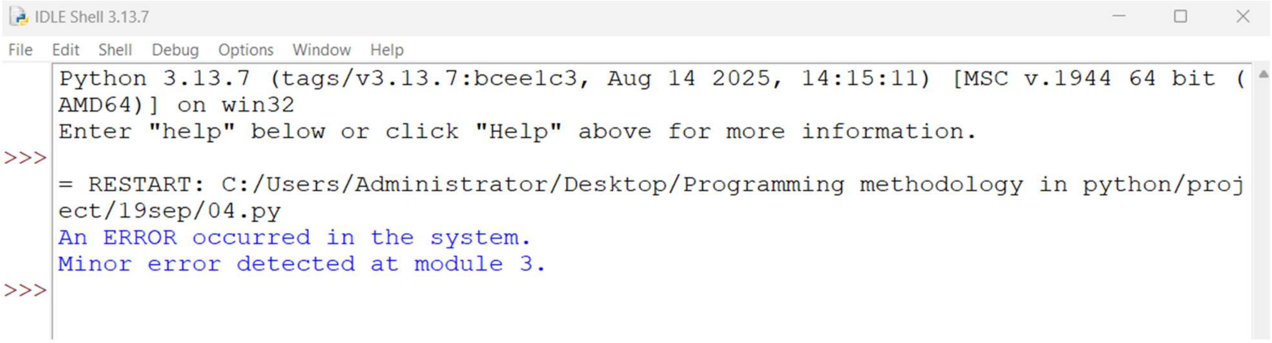
Approach:

1. Open the given text file in read mode.
2. Loop through each line of the file.
3. Convert the line to lowercase and check if the word 'error' exists.
4. If yes, print that line after removing leading/trailing spaces.

Code:

```
with open('C://Users//Administrator//Desktop/Programming
methodology in python//project//19sep//error.txt') as f:
    for line in f:
        if 'error' in line.lower():
            print(line.strip())
```

Sample Run:



```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
ect/19sep/04.py
An ERROR occurred in the system.
Minor error detected at module 3.
>>>
```

Problem 5: Group Words by First Letter

Approach:

1. Take the list of words as input.
2. Create an empty dictionary.
3. For each word, check its first letter.
4. If the first letter already exists in dictionary, append the word to its list.
5. Otherwise, create a new key with that letter and start a new list.
6. Finally, print the dictionary containing words grouped by first letter.

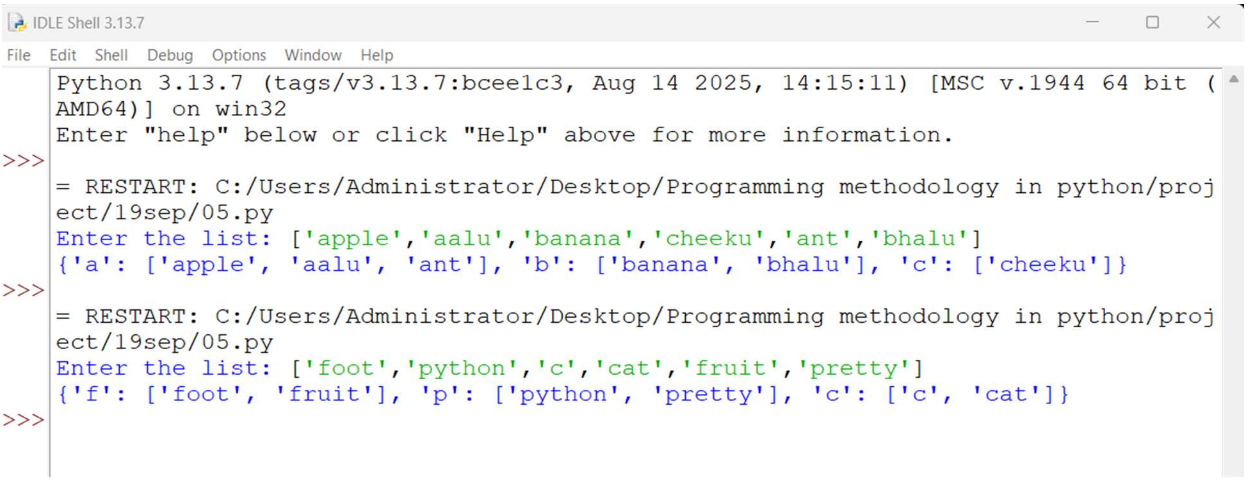
Code:

```
l = eval(input('Enter the list: '))
d = {}

for i in l:
    if i[0] in d:
        d[i[0]].append(i)
    else:
        d[i[0]] = [i]

print(d)
```

Sample Run:



```
    Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
    AMD64)] on win32
    Enter "help" below or click "Help" above for more information.
>>>
    = RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
    ect/19sep/05.py
    Enter the list: ['apple','aalu','banana','cheeku','ant','bhalu']
    {'a': ['apple', 'aalu', 'ant'], 'b': ['banana', 'bhalu'], 'c': ['cheeku']}
>>>
    = RESTART: C:/Users/Administrator/Desktop/Programming methodology in python/proj
    ect/19sep/05.py
    Enter the list: ['foot','python','c','cat','fruit','pretty']
    {'f': ['foot', 'fruit'], 'p': ['python', 'pretty'], 'c': ['c', 'cat']}
>>>
```