List

More on Sorting Lists

```
>>> L = ['abc', 'ABD', 'aBe']
>>> L.sort()                          # Sort with mixed case
>>> L
['ABD', 'aBe', 'abc']
>>> L = ['abc', 'ABD', 'aBe']
>>> L.sort(key=str.lower)             # Normalize to lowercase
>>> L
['abc', 'ABD', 'aBe']
```

More on Sorting Lists

```
>>> L

['abc', 'ABD', 'aBe']

>>> L.sort(key=str.lower, reverse=True)


# Change sort order

>>> L ['aBe', 'ABD', 'abc']
```

Other common list methods

```
>>> L = [1, 2]

>>> L.extend([3, 4, 5])  # Add many items at end (like in-place +)

>>> L

[1, 2, 3, 4, 5]

>>>k=[11,12]

>>>k.extend(L)
```

Other common list methods

```
>>> L

[1, 2, 3, 4]

>>> L.reverse()                    # In-place reversal method

>>> L

 [4, 3, 2, 1]

>>> list(reversed(L))          # Reversal built-in with a result

 [1, 2, 3, 4]

>>>x= list(reversed(L))

>>>x

 [1, 2, 3, 4]
```

Other common list methods

```
>>> L = ['spam', 'eggs', 'ham']     # Index of an object (search/find)
>>> L.index('eggs')
1


>>> L.insert(1, 'toast')            # Insert at position
>>> L
['spam', 'toast', 'eggs', 'ham']

>>> L.remove('eggs')                # Delete by value
>>> L
['spam', 'toast', 'ham']
```

Other common list methods

```
>>> L.pop()                    # Delete and return last item
 5                              # L=[1,2,3,4,5]
>>>L=['spam', 'toast', 'ham']
>>> L.pop(1)                    # Delete by position 'toast'
>>> L
['spam', 'ham‘]
>>> L.count('spam')             # Number of occurrences
1
```

Other common list methods

```
>>> L = ['spam', 'eggs', 'ham', 'toast']
>>> del L[0]                    # Delete one item
>>> L
['eggs', 'ham', 'toast']
>>> del L[1:]                   # Delete an entire section
>>> L                           # Same as L[1:] = []
['eggs‘]
```

Other common list methods

```
>>> L = ['Already', 'got', 'one']
>>> L[1:] = []                    #equivalent to del L[1:]
>>> L
['Already']
>>> L[0] = []                     #empty the element at 0
>>> L
[[]]
```

Strings and Lists

```
>>> S = ‘interest'
>>> L = list(S)
>>> L
[‘i', ‘n', ‘t', ‘e', ‘r', ‘e‘, ’s’, ‘t’]
>>> L[3] = ‘x'                    # Works for lists, not strings
>>> L[4] = ‘x‘
>>> L
[‘i', ‘n', ‘t', ‘x', ‘x', ‘e‘, ’s’, ‘t’]
```

Strings and Lists

>>> S = “”**.join(L)** #uses ‘” for joining elements of list

>>> S

‘intxxest‘

Strings and Lists

```
>>>s= ‘#'.join(['eggs', 'sausage', 'ham', 'toast'])

>>s

'eggs#sausage#ham#toast‘

>>> line = 'aaa bbb ccc'

>>> cols = line.split()    #default separator is “ “ (space)

>>> cols

['aaa', 'bbb', 'ccc']
```

Strings and Lists

Syntax

string.split(*separator, maxsplit*)

```
>>>txt = "hello+ my name is Ramu+ I am 17 years old"
>>>x = txt.split(“+")
>>>x
['hello', ' my name is Ramu', ' I am 17 years old']
```

Strings and Lists

```
>>>txt = "apple#banana#cherry#orange"
>>>x = txt.split("#")
>>>x
['apple', 'banana', 'cherry', 'orange']
>>>txt = "apple#banana#cherry#orange"
# setting the maxsplit parameter to 1, will return a list with 2 elements!
#Specifies how many splits to do.
>>>x = txt.split("#", 2)
>>>x
['apple', 'banana', '#cherry#orange']
```

Get a list as input from user

```
# creating an empty list
lst = []
# number of element's as input
n = int(input("Enter number of elements : "))
# iterating till the range
for i in range(0, n):
    x = input()
    lst.append(x)  # adding the element
print(lst)
```

Matrixes

a basic 3 × 3 two-dimensional list-based array:

>>> **matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]**

>>>matrix

1 2 3

4 5 6

5 7 8

- With **one index**, you get an **entire row** (really, a nested sublist), and with two, you get an item within the row:

Matrixes

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

```
>>> mymatrix[1]
[4, 5, 6]
>>> mymatrix[1][1]
5
>>> mymatrix[2][0]
7
>>> mymatrix = [[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]]

>>> mymatrix[1][2]
6
```

```
>>>mymatrix[0]
[1,2,3]
>>>mymatrix[0][0]
1
>>>mymatrix[0][2]
3
```

```
R = int(input("Enter the number of rows:"))
C = int(input("Enter the number of columns:"))
matrix = []
print("Enter the entries rowwise:")
for i in range(R):
        a =[]
        for j in range(C):
                a.append(int(input()))
        matrix.append(a)
# For printing the matrix
for i in range(R):
        for j in range(C):
                print(matrix[i][j], end = " ")
        print()
```

Hence…

- A list is a **sequence of items** stored as a **single object**.
- Items in a list can be **accessed by indexing**, and sub lists can be accessed by slicing.
- Lists are **mutable**; individual items or entire slices can be replaced through assignment statements.
- Lists support a number of convenient and frequently used **methods**.
- Lists will **grow and shrink** as needed.

Problem

A farmer with a fox, a goose, and a sack of corn needs to cross a river. Now he is on the east side of the river and wants to go to west side. The farmer has a rowboat, but there is room for only the farmer and one of his three items. Unfortunately, both the fox and the goose are hungry. The fox cannot be left alone with the goose, or the fox will eat the goose. Likewise, the goose cannot be left alone with the sack of corn, or the goose will eat the corn. Given a sequence of moves find if all the three items fox, goose and corn are safe. The input sequence indicate the item carried by the farmer along with him in the boat. 'F' – Fox, 'C' – Corn, 'G' – Goose, N-Nothing. As he is now on the eastern side the first move is to west and direction alternates for each step.

```
Pseudocode
READ items_carried
SET east as G, C, F
SET west as empty
SET from_Dir = east and to_Dir = west
FOR each item in items_carried
        IF from_Dir == east THEN
                remove item from east and add to west
                IF east or west contains ‘C’ and ‘G’ or ‘G’ and ‘F’ THEN
                        PRINT ‘NOT SAFE’
                        BREAK
        ELSE
                remove item from west and add to east
                IF east or west contains ‘C’ and ‘G’ or ‘G’ and ‘F’ THEN
                        PRINT ‘NOT SAFE’
                        BREAK
        END IF
IF from_Dir == east THEN
        SET from_Dir = west
        SET to_Dir = east
ELSE
        SET from_Dir = east
        SET to_Dir = west
END IF
END FOR
PRINT ‘SAFE’
```