Module 9

Data Structures

A data structure is a specialized format for organizing, processing, retrieving and storing data.

Example Scenario: In the Super market, various products are stored / organized very nicely, so that the business could be run very effectively.

In Python, we have the following Data structures

- Lists
- Tuples
- Dictionary
- Sets

Lists []

- Ordered collection of data → So, indexing is possible
- Lists are mutable type of data structure
- It can contain multiple type of data
- Lists are changeable (Mutable)
- Lists are iterable

```
l = list()
print(type(l))
<class 'list'>
```

①

```
l1 = [1, 3, 5, 2]
type(l1)
list
```

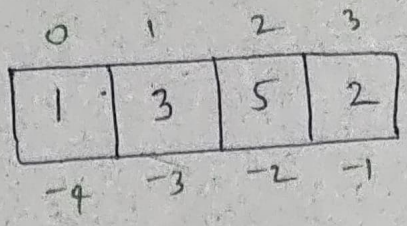
②

2 ways of creating a List.

```
len(l1)
4
```

Accessing List

```
l1
[1, 3, 5, 2]
l1[0]
1
l1[-4]
1
l1[-len(l1)]
1
l1[-1]
2
```



Mutability

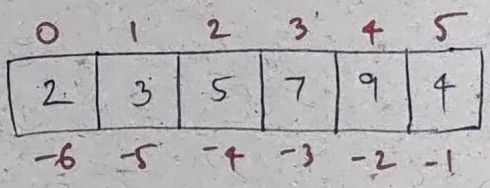
```
id(l1)
140592400639488
l1[0] = 9
l1
[9, 3, 5, 2]
id(l1)
140592400639488
```

Iterable

```
for i in l1:
    print(i, end=" ")
9 3 5 2
```

List Slicing

```
l = [2, 3, 5, 7, 9, 4]
len(l)
6
l[0:4]
[2, 3, 5, 7]
```



```
l[::]
[2, 3, 5, 7, 9, 4]
l[::-1]
[4, 9, 7, 5, 3, 2]
l[5:0]
[]
l[5:0:-1]
[4, 9, 7, 5, 3]
```

List Inbuilt methods

- count
- index
- pop
- remove
- sort
- insert
- append
- extend

count : Returns the count of an object

```
l = [100, 200, 150, 100]
l.count(100)
2
l.count(500)
0
```

index : Returns the index of 1st occurence of an object

```
l.index(200)
1
l.index(600)
ValueError : 600 is not in list
name = ["Uvaraj", "Emma", "Harry", "Ronnie"]
name.index("Harry")
2
```

pop : Remove and returns the last element of a list.

```
drop = name.pop()
drop
"Ronnie"
name
['Uvaraj', 'Emma', 'Harry']
```

remove : Removes the given object from the list.

```
name .remove ("Emma")
name
['Uvaraj', 'Harry']
name .remove ("Thor")
ValueError : list.remove(x) : x not in list
```

sort : Sorting the list.

```
l .sort ?
Signature : l.sort(*, key=None, reverse=False)
Docstring : Sort the list in ascending order and return None.
The sort is in place (i.e. the list itself is modified) and
stable (i.e. the order of two equal elements is maintained).

If a key function is given, apply it once to each list item
and sort them ascending or descending, according to their
function values.
The reverse flag can be set to sort in descending order.
l.sort()
[100, 100, 150, 200]
```

insert : Helps to add an element at a given index.

```
name .insert ?
Signature : name.insert (index, object, /)
Docstring : Insert object before index.
Type : builtin_function_or_method
name .insert (1, "Emma")
name
['Uvaraj', 'Emma', 'Harry']
name .insert (0, "Ronnie")
name
['Ronnie', 'Uvaraj', 'Emma', 'Harry']
```

append : Helps to add elements at the end of the list

```
l = [1, 2, 3, 4, 5]
l.append("Uvaraj")
l
[1, 2, 3, 4, 5, 'Uvaraj']
l.append(2)
l
[1, 2, 3, 4, 5, 'Uvaraj', 2]
l1 = [6, 7, 8]
l.append(l1)
l
[1, 2, 3, 4, 5, 'Uvaraj', 2, [6, 7, 8]]
```

extend

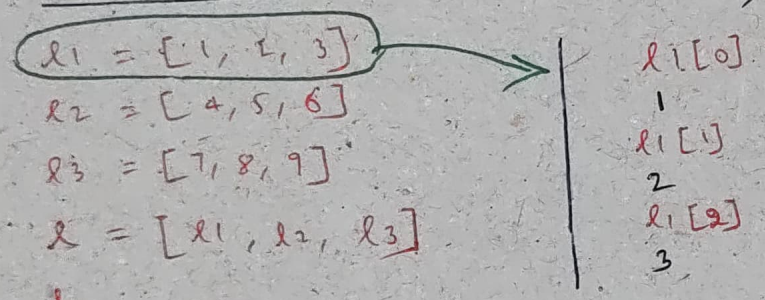
```
l = [1, 2, 3, 4, 5]
l1
[6, 7, 8]
l.extend(l1)
l[1, 2, 3, 4, 5, 6, 7, 8]
s = "Uvaraj"
for i in s:
    print(i, end = " ")
U v a r a j
l.extend(s)
l
[1, 2, 3, 4, 5, 6, 7, 8, 'U', 'v', 'a', 'r', 'a', 'j']
```

Heterogenous Lists

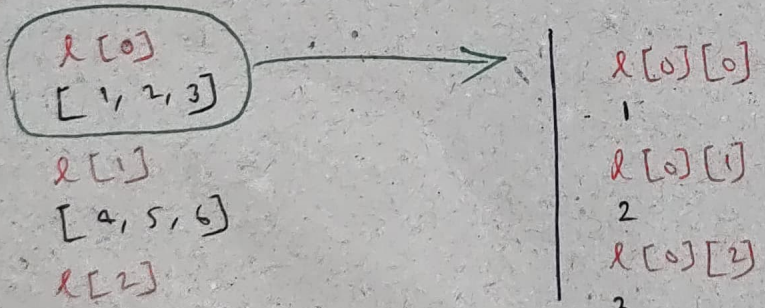
```
# List with different data types
l = [2, "Uvaraj", 1.3, True]
type(l)
list
type(l[0])
int
type(l[1])
str
type(l[2])
float
type(l[3])
bool
for i in l:
    print(i, end=" ")
2 Uvaraj 1.3 True
```

2D - Lists (Lists inside a List)

```
l1 = [1, 2, 3]
l2 = [4, 5, 6]
l3 = [7, 8, 9]
l = [l1, l2, l3]
l
[ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]
```




```
l[0]
[1, 2, 3]
l[1]
[4, 5, 6]
l[2]
[7, 8, 9]
```



Iteration in 2D list

```
for i in l:
    print(i)
```

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

```
for i in l1:
    print(i)
```

```
1
2
3
```

```
for i in l:
    for j in i:
        print(j, end=" ")
```

```
1 2 3 4 5 6 7 8 9
```

List comprehension

```
for i in range(10):
    print(i, end=" ")
```

i ** 2

```
0 1 2 3 4 5 6 7 8 9
```

```
l = []
for i in range(10):
    l.append(i)
print(l)
```

i ** 2

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]   [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
l = [i for i in range(10)]
l
```

i ** 2

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]   [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Challenge

① Given a list of population census done by Govt. of India. You need to find the total population.
Each element in the list represents total members in a family.

l = [4, 6, 5, 8, 9, 3, 2, 4, 5, 4, 3, 2, 3, 4]

[Hint] Access all the elements of the list, and sum them up.

```
l = [4, 6, 5, 8, 9, 3, 2, 4, 5, 4, 3, 2, 3, 4]
population = 0
for i in l:
    population += 1
print (population)
```

62

② Second Largest

You are given an integer array A. You have to find the second largest element/value in the array or report that no such element exists.

Problem constraints:

$1 \leq |A| \leq 10^5$

$0 \leq A[i] \leq 10^9$

Input format:

The first argument is an integer array A.

Output format:

Return the second largest element. If no such element exist, then return -1.

Sample input/output:

A = [2, 1, 2]

1

A = [2]

-1

```
class Solution:
    def solve(self, A):
        if len(A) < 2:
            return -1
        first = second = -1
        for num in A:
            if num > first:
                second = first
                first = num
            elif first > num > second:
                second = num
        return second if second != -1 else -1

solution = Solution()
int_array = [2, 3, 4, 7, 5, 9]
print(solution.solve(int_array))
```

7

③ Is it Identity Matrix ?

You are given a N×N square integer matrix A. You have to tell whether A is an identity matrix or not.

Identity matrix is a special square matrix whose main diagonal elements are equal to 1 and all other elements are 0.

Problem Constraints :

$2 \leq N \leq 10^3$

A[i][j] equals 0 or 1.

Input format :

The first argument is a 2D integer array denoting the matrix A.

Output format :

Return 1 if A is an identity matrix, else return 0.

Sample input / output :

[[1, 0], [0, 1]]

1

[[0, 0, 1], [0, 1, 0], [1, 0, 0]]

0

```
class Solution:
    def solve(self, A):
        n = len(A) #Dimension of the square matrix

        #check each element in the matrix
        for i in range(n):
            for j in range(n):
                if i == j: #Diagonal elements
                    if A[i][j] != 1:
                        return 0
                else: #Non-diagonal elements
                    if A[i][j] != 0:
                        return 0
        return 1 # if all conditions are satisfied

Solution = Solution()
identity = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
print(Solution.solve(identity))
```

1

④ Which of the options adds the elements 'Dhoni' and 'Virat' to the end of the given list ?

players = ['Jadeja', 'Rahul', 'Rohit']

So, the resultant list will be

players = ['Jadeja', 'Rahul', 'Rohit', 'Dhoni', 'Virat']

a) players += 'Dhoni Virat'

b) players [len (players) :] = ['Dhoni', 'Virat'] ✓

c) players [-1 :] = ['Dhoni', 'Virat']

d) players += ['Dhoni', 'Virat']

⑤ After executing the code below, the given word gets converted into a list of individual characters of the word called word_list, indexing on which elements of the list word_list would give us a sublist that forms the word "Data Science" ?

```
word = " Scaler Data Science Course"
word_list = list (word)
```

a) print (word_list [1 : 3])

b) print (word_list [7 : 19])

c) print (word_list [7 : 18])

d) print (word_list [1 : 4])

['D', 'a', 't', 'a', ' ', 'S', 'c', 'i', 'e', 'n', 'c', 'e']