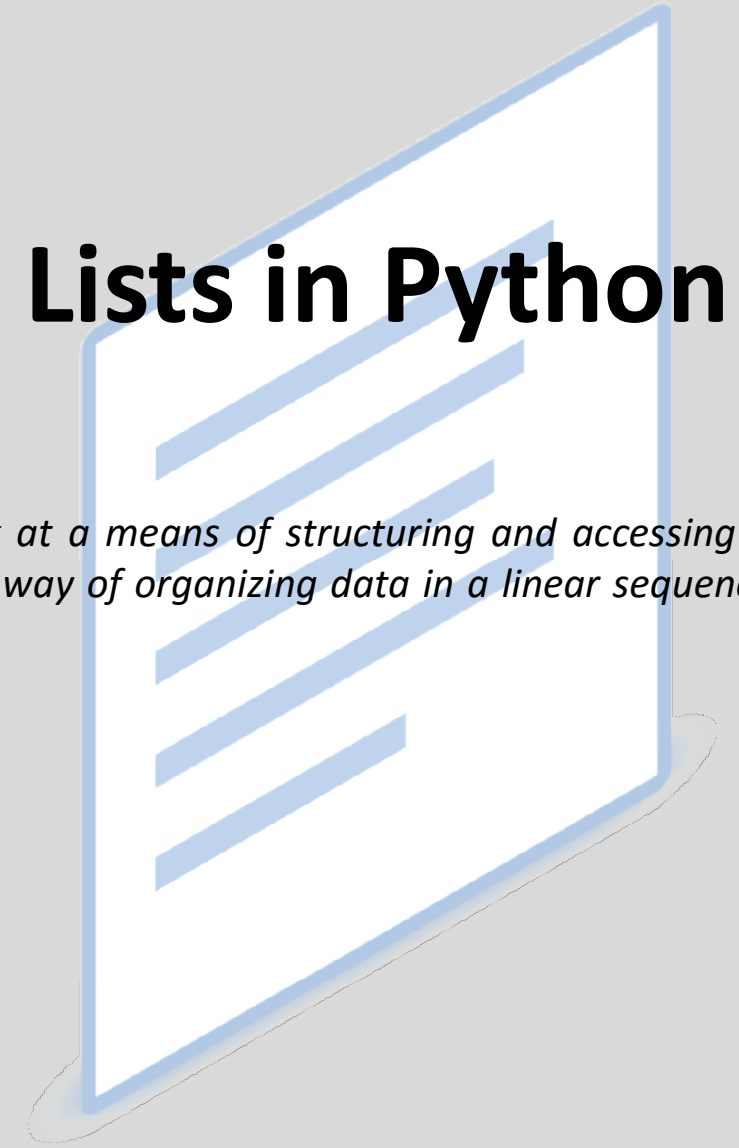
Python Collections (containers or Arrays)

There are four collection data types in the Python programming language, They are

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

Lists in Python

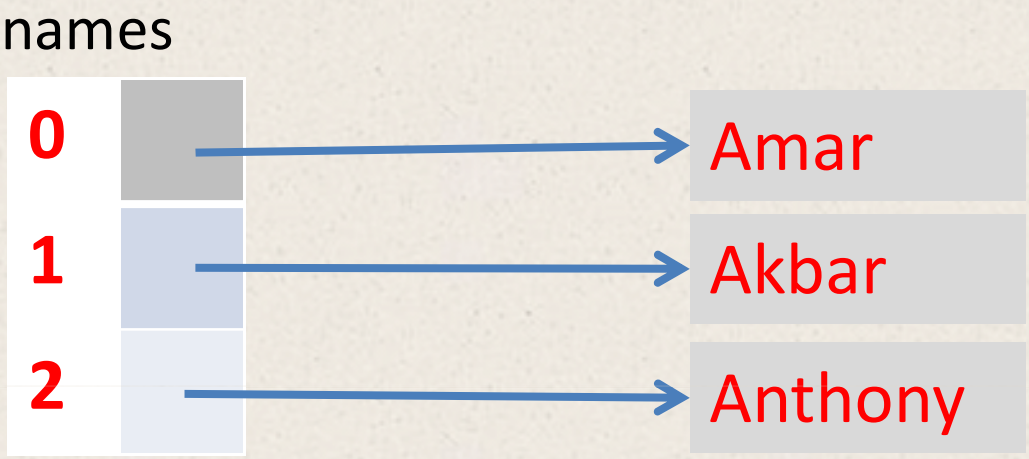


In this chapter, we look at a means of structuring and accessing a collection of data. In particular, we look at a way of organizing data in a linear sequence, generally referred to as a list.

list: collection of items

- It is a data structure, can be homogeneous or heterogeneous
- It Allows duplicate items
- It has 0 or more elements
- There is no name for each element
- The elements are accessed by using index or subscript
- The index starts from 0
- The size of the list is not fixed. The list can grow or shrink.
- We can find the number of items in a list at any point in time.
- It is mutable

names=[“Amar”, “Akbar”; “Anthony”]



names[0] gives Amar
names[1] gives Akbar

print(names)

All items of the list

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

Common List Operations

Operations commonly performed on lists include:

- access (retrieve)
- update
- append
- insert
- delete (remove)

List Traversal

A **list traversal** is a means of accessing, one-by-one, each element of a list.

List traversal may be used, for example, to:

- search for a particular item in a list
- add up all the elements of a list

Lists (Sequences) in Python

A **list** in Python is a mutable, linear data structure of variable length, allowing mixed-type elements.

By *mutable* it is meant that the contents of the list may be altered. Lists in Python use zero-based indexing. Thus, all lists have index values 0..n-1, where n is the number of elements in the list.

Lists are denoted by a comma-separated list of elements within square brackets,

[1, 2, 3] ['one', 'two', 'three'] ['apples', 50, True]

An **empty list** is denoted by an empty pair of square brackets, []. Elements of a list are accessed by use of an index value within square brackets,

Example:

lst = [1, 2, 3],

lst[0] → 1 access of first element

lst[1] → 2 access of second element

lst[2] → 3 access of third element

For example, the following prints the first element of list lst,

```
print (lst[0])
```

The elements of lst can be summed as follows,

```
sum = lst[0] + lst[2] + lst[2]
```

To update,	lst[2] = 4	*replacement of 3 with 4 at index 2*
To delete,	del lst[2]	*removal of 4 at index 2*
To insert,	lst.insert(1,3)	*insertion of 3 at index 1*
To append,	lst.append(4)	appends 4 to end of list

Common List Operations

Operation	fruit = ['banana', 'apple, 'cherry']	
Replace	`fruit[2] = 'coconut'`	`['banana', 'apple', 'coconut']`
Delete	`del fruit[0]`	`['apple', 'cherry']`
Insert	`fruit.insert(2, 'pear')`	`['banana', 'apple', 'pear', 'cherry']`
Append	`fruit.append('peach')`	`['banana', 'apple', 'cherry', 'peach']`
Sort	`fruit.sort()`	`['apple', 'banana', 'cherry']`
Reverse	`fruit.reverse()`	`['cherry', 'apple', 'banana']`

List.shuffle()