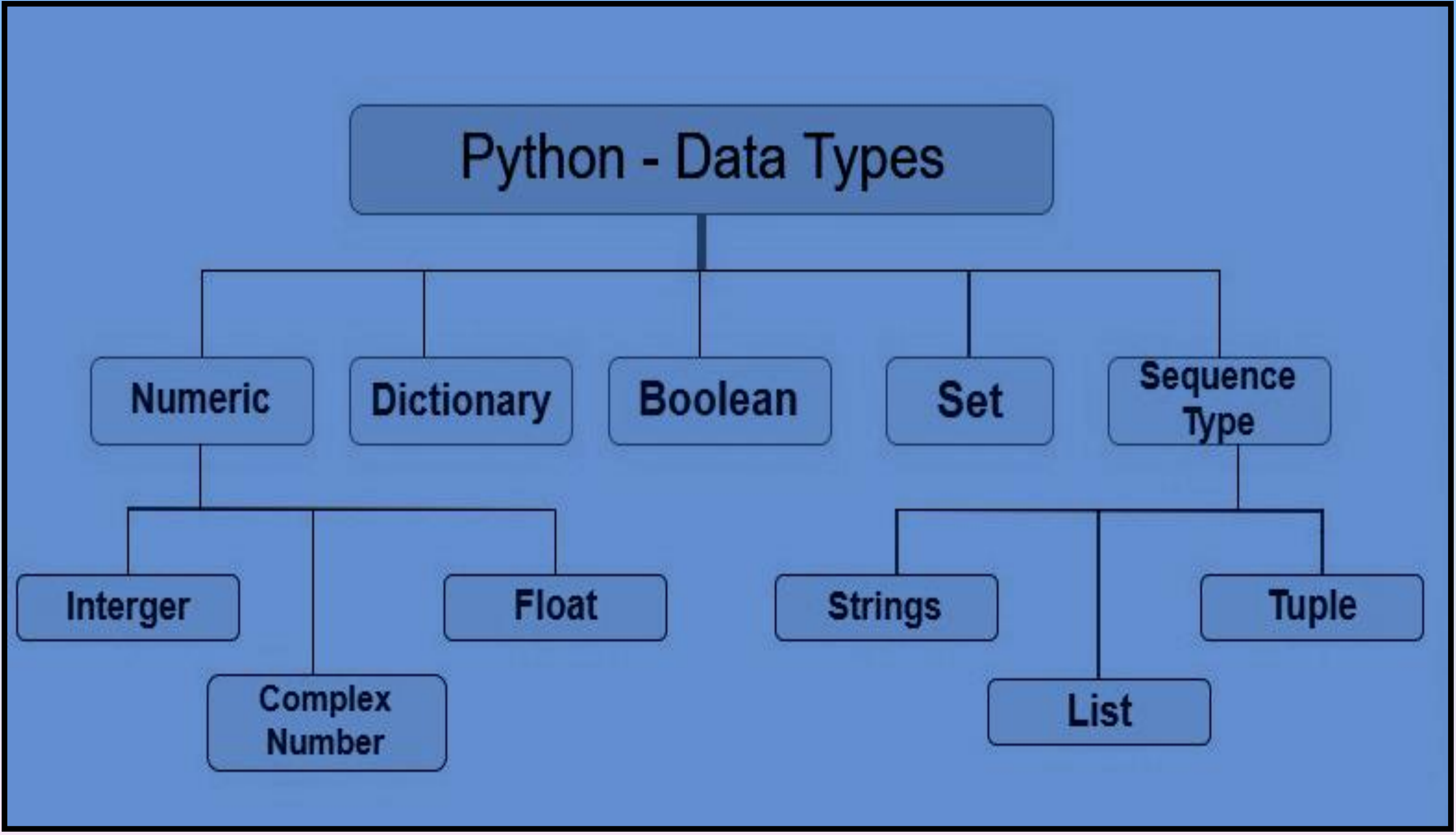
PYTHON LIST
CLASS XI
(MODULE-1)

BY

Mrs. SUJATA PRADHAN,
PGT(SS), Computer Science,
AECS,ANUPURAM

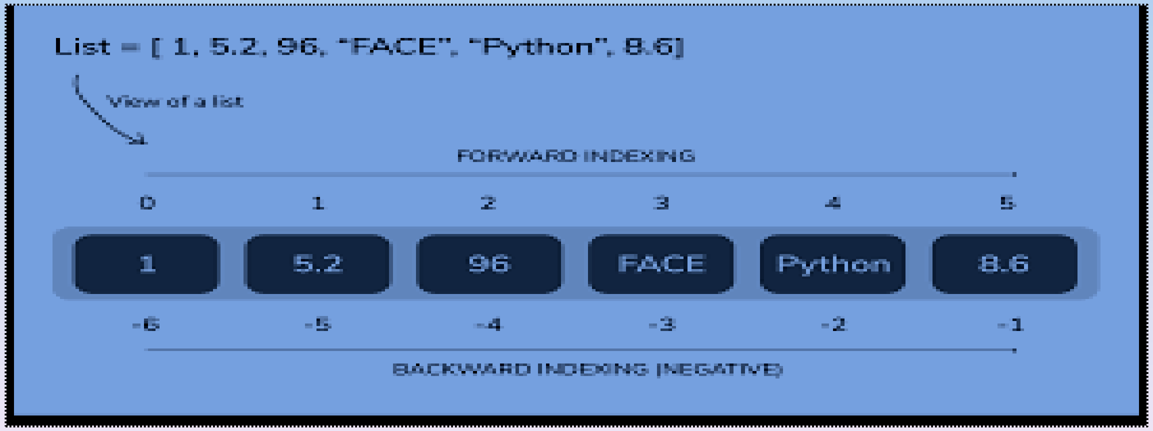
Python Datatypes

Data types are the classification or categorization of data items. Each programming language has its own classification. Data types represent a kind of value which determines what operations can be performed on that data. Python has an in-built function **type()** to ascertain the data type of a certain value.



Python List

A **sequence** is an ordered collection of similar or different data types. Python consists of several data types which are capable of storing sequences, but the most common and reliable type is the **list**. A list in Python is used to store the sequence of various types of data. It also allows duplicate members. A list can be defined as a collection of values or items of same or different types. The items of a list are separated with comma (,) and enclosed with the square brackets [].


List = [1, 5.2, 96, "FACE", "Python", 8.6]
View of a list
FORWARD INDEXING
0
1
2
3
4
5
1
5.2
96
FACE
Python
8.6
-6
-5
-4
-3
-2
-1
BACKWARD INDEXING (NEGATIVE)

Characteristics of Lists

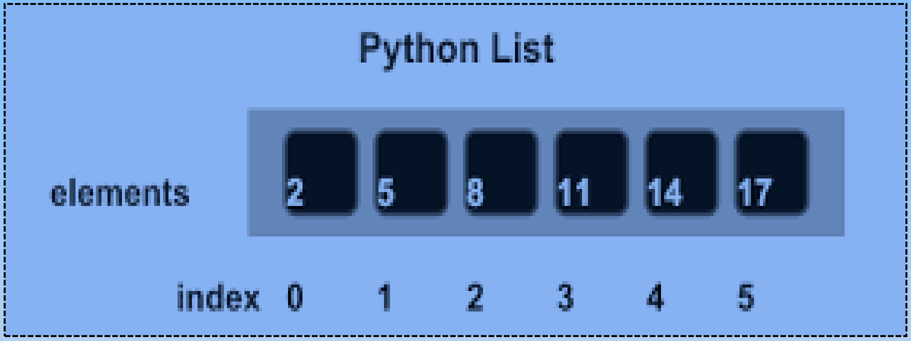
The list has following **characteristics**:

•The lists are ordered or sequenced.

•Each element of the list is accessed by index.

•The lists are mutable types.

•A list can store different types of elements.



Note: Each element of a sequence is assigned a number i.e. its position or index. The first index is zero, the second index is one, and so on. We can access an item from a list using index. In Python, indices start from 0. So, a list having 5 elements will have an index from 0 to 4.Trying to access indices other than mentioned range will raise an IndexError. The index must be an integer. We can't use float or other types, otherwise this will result in TypeError.Nested lists are accessed using nested indexing.

How to create a list?

In Python programming, a list is created by placing all the items (elements) inside square brackets [], separated by commas. It can have any number of items and they may be of different types (integer, float, string, tuples etc.).

```
# empty list
list1 = []
 # list of integers
list2 = [1, 2, 3]
# list of floats
 list3 = [11.22, 9.9, 78.34, 12.0]
 # list with mixed data types
list4 = [1, "Hello", 3.4]
```

If we try to print the type of L1 and L2 using **type()** function then it will come out to be lists.

```
L1 = [“Sam", 102, "USA"]
L2 = [11, 20, 3, 47, 55, 96]
print(type(L1))
print(type(L2))
```

Output:

```
<class 'list'>
<class 'list'>
```

List Operations

The most basic **data structure** in Python is the **sequence**. We can do certain operations like indexing, slicing, adding, multiplying, and checking for membership with sequences. In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements.

# List indexing

```
 mylist = ['p', 'r', 'o', 'b', 'e']
 print(mylist[0])
```

Output: p

```
print(mylist[2])
```

Output: o

```
print(mylist[4])
```

Output: e

# Nested List

```
L = ["Happy", [2, 0, 1, 5]]
```

# Nested indexing

```
 print(L[0][1])
print(L[1][3])
```

Output

```
'a'
 5
```

# Error! Only integer can be used for indexing

```
 print(mylist[4.0])
```

List Indexing & List Slices:

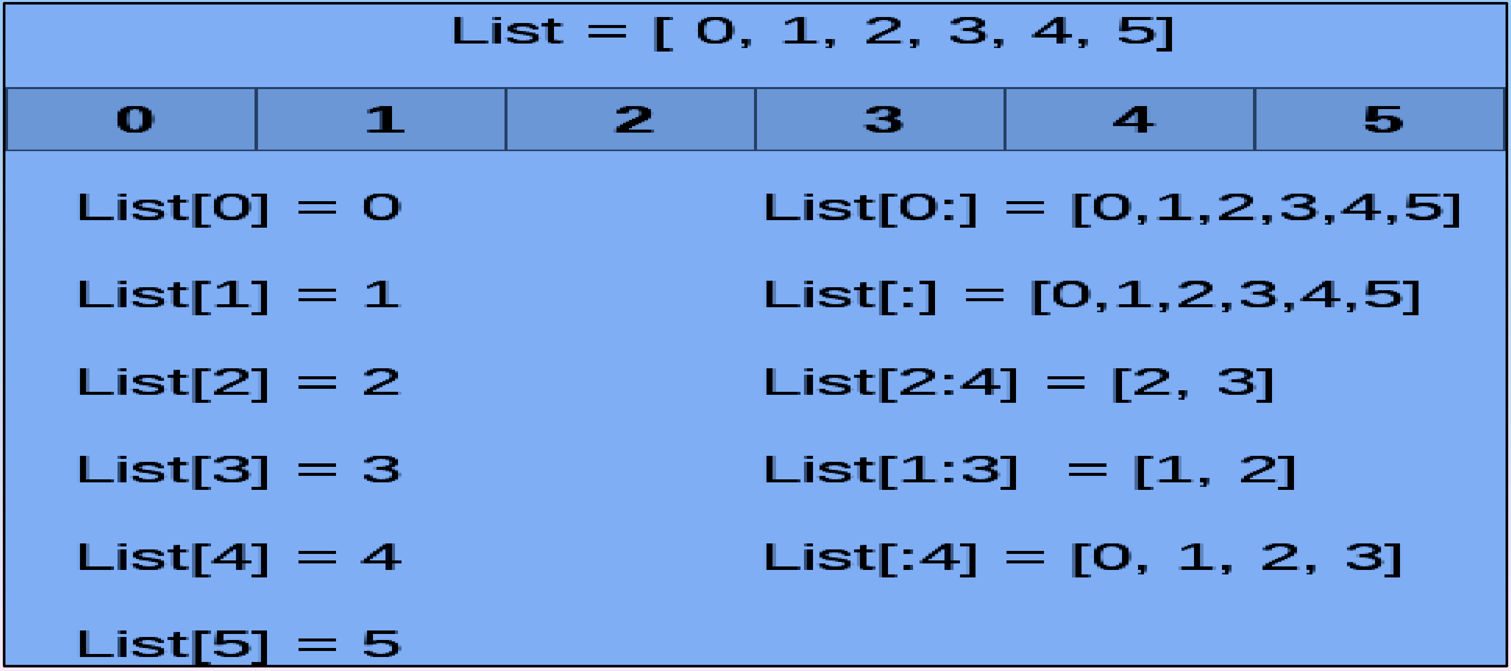
List slicing is an operation that extracts a subset of elements from a list and packages them as another list. We can get a sublist from a list in Python using slicing operation.

Syntax:

Listname[start:stop]

Listname[start:stop:step]

- default start value is 0
- default stop value is n-1
- [:] will print the entire list
- [n:n] will create a empty slice



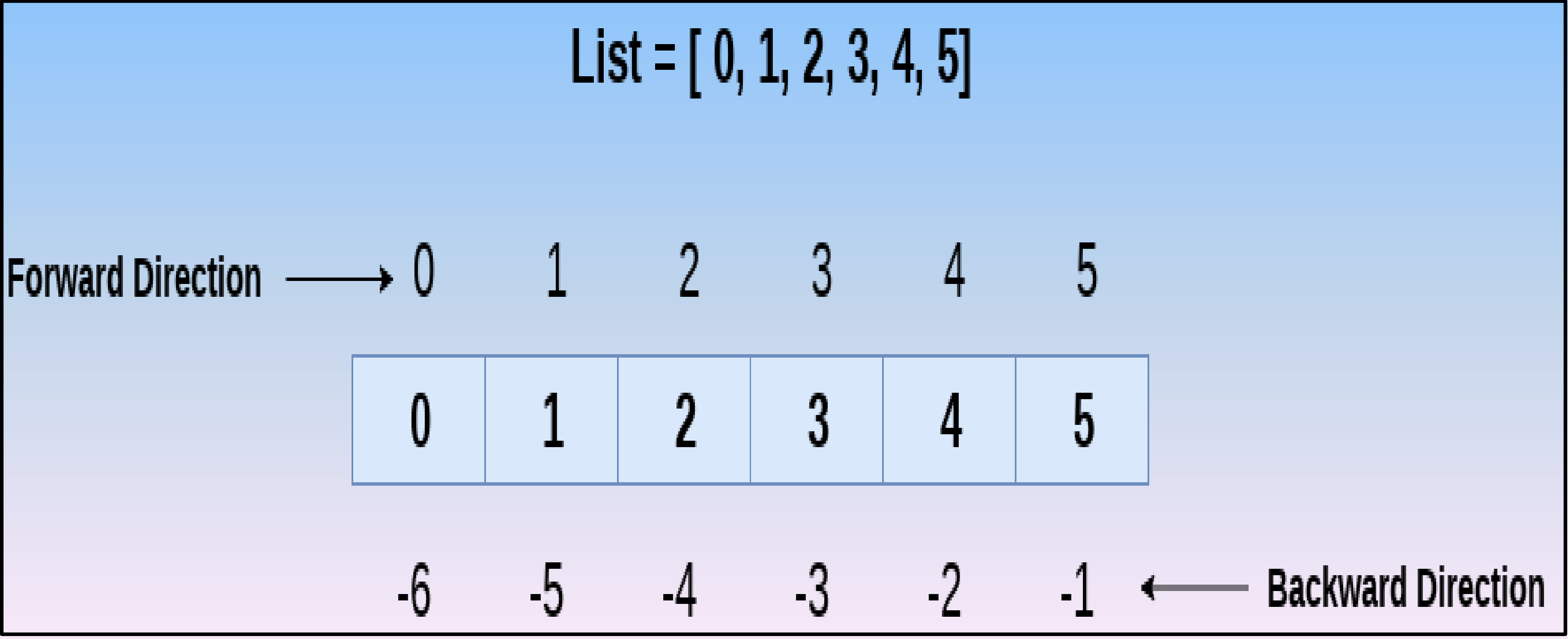
List = [0, 1, 2, 3, 4, 5]

0	1	2	3	4	5

```
List[0] = 0          List[0:] = [0,1,2,3,4,5]
List[1] = 1          List[:] = [0,1,2,3,4,5]
List[2] = 2          List[2:4] = [2, 3]
List[3] = 3          List[1:3]  = [1, 2]
List[4] = 4          List[:4] = [0, 1, 2, 3]
List[5] = 5
```

Positive & Negative Index

Python programming language supports **positive as well as negative indexing** of list which is not available in most other programming languages. We can access the elements of an array by going through their indices. This means that the index value of -1 gives the last element, and -2 gives the second last element of an array. The negative indexing starts from where the list ends. This means that the last element of the array is the first element in the negative indexing which is -1.



## Negative indexing

Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

# Negative indexing in lists

```
mylist = ['p',‘x','o',‘m',‘s']
print(mylist[-1])
print(mylist[-5])
```

output:

```
s p
```

# a list of strings

```
 my_list = ["hello", "world", "hi", "bye"]
print(mylist[-1])
print(mylist[-3])
print(mylist[-4])
```

Output:

```
bye
World
 hello
```

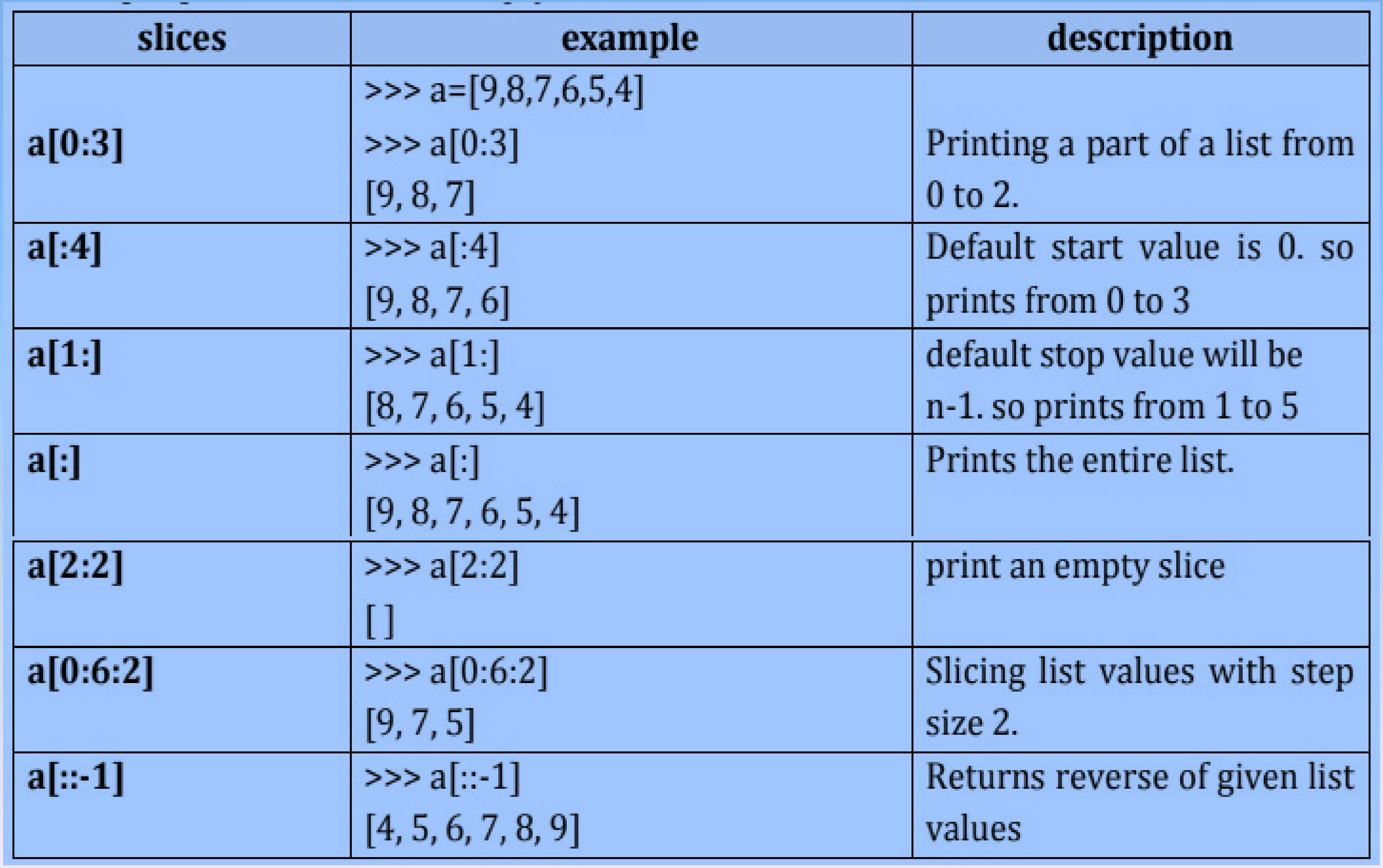
# a list of numbers

```
arr = [10, 20, 30, 40, 50]
print(arr[1])
print(arr[−1])
print(arr[−2])
```

Output:

```
20
50
40
```

Slices….



slices	example	description

a[0:3]	>>> a=[9,8,7,6,5,4]
	>>> a[0:3]
	[9, 8, 7]
		Printing a part of a list from 0 to 2.

a[:4]	>>> a[:4] [9, 8, 7, 6]	Default start value is 0. so prints from 0 to 3
a[1:]	>>> a[1:] [8, 7, 6, 5, 4]	default stop value will be n-1. so prints from 1 to 5
a[:]	>>> a[:] [9, 8, 7, 6, 5, 4]	Prints the entire list.
a[2:2]	>>> a[2:2] [ ]	print an empty slice
a[0:6:2]	>>> a[0:6:2] [9, 7, 5]	Slicing list values with step size 2.
a[::-1]	>>> a[::-1] [4, 5, 6, 7, 8, 9]	Returns reverse of given list values

Mutability:

Python defines variety of data types for **objects** and they are stored in memory. **Mutability** is the ability for certain types of data to be changed without entirely recreating it and object mutability also depends upon the type, like Lists and Dictionaries. They are mutable. it means that we can change their content without changing their identity. Python lists are mutable as we can modify its element. Other objects like Integers, Floats, Strings and Tuples are immutable. An immutable object is an object whose state can never be modified after it is created.

```
# Python code to test that  lists
      are mutable
color = ["red", "blue", "green"]
print(color)

color[0] = "pink"
color[-1] = "orange"
print(color)
```

OUTPUT

```
['red', 'blue', 'green']
['pink', 'blue', 'orange']
```

```
my_list = [10, 20, 30]
my_list[0] = 40
print(my_list)
```

OUTPUT

```
[40, 20, 30]
```

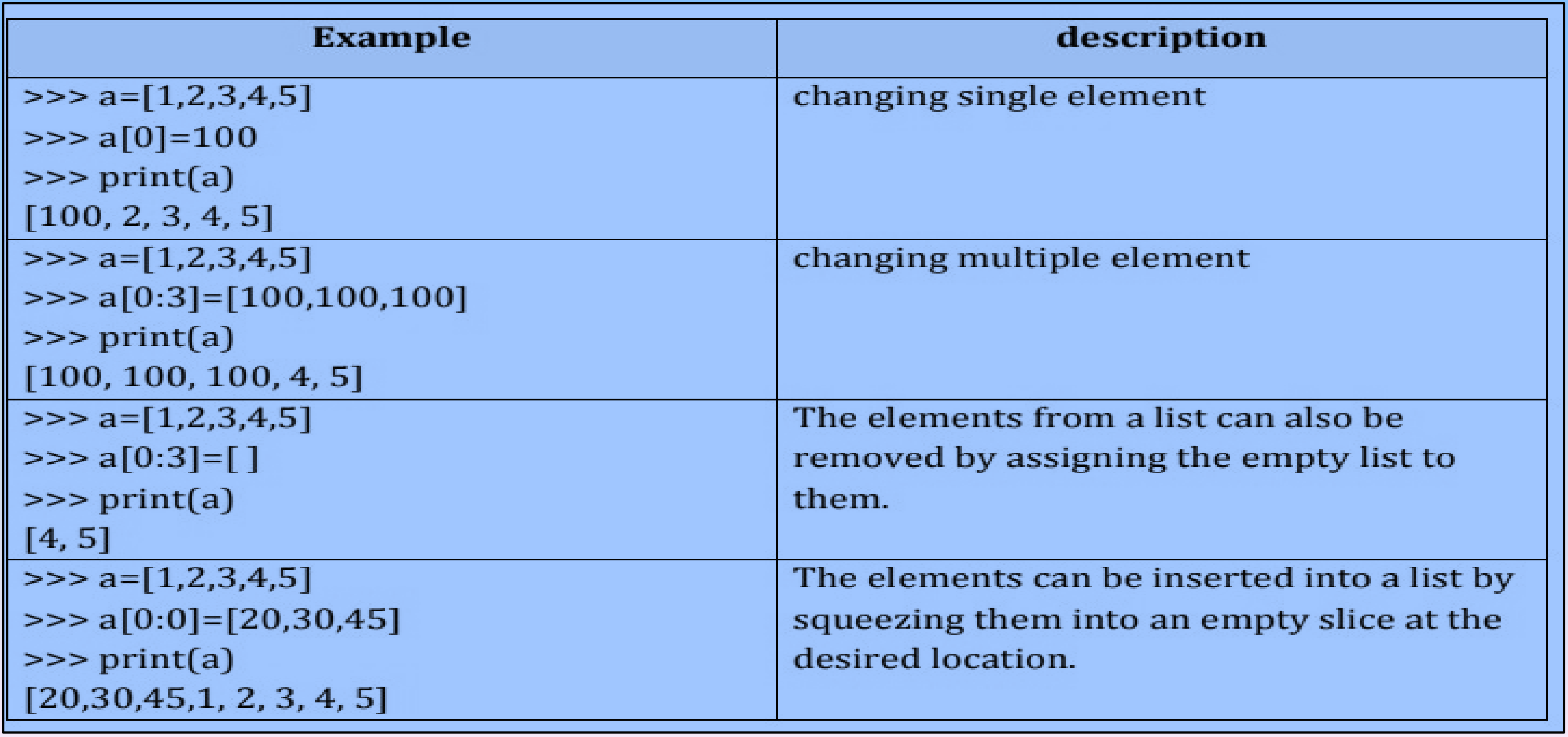
```
my_tuple = (10, 20, 30)
my_tuple[0] = 40
print(my_tuple)
```

OUTPUT

```
Traceback (most recent call last):
  File "test.py", line 3, in < module >
    my_tuple[0] = 40
TypeError: 'tuple' object does not support item assignment
```

Mutability:

Lists are mutable. An item can be changed in a list by accessing it directly as part of the assignment statement. It is possible to add, delete, insert, and rearrange items in a list or dictionary. Hence, they are mutable objects.



Example	description

>>> a=[1,2,3,4,5]
>>> a[0]=100
>>> print(a)
[100, 2, 3, 4, 5]
	changing single element

>>> a=[1,2,3,4,5]
>>> a[0:3]=[100,100,100]
>>> print(a)
[100, 100, 100, 4, 5]
	changing multiple element

>>> a=[1,2,3,4,5]
>>> a[0:3]=[ ]
>>> print(a)
[4, 5]
	The elements from a list can also be removed by assigning the empty list to them.

>>> a=[1,2,3,4,5]
>>> a[0:0]=[20,30,45]
>>> print(a)
[20,30,45,1, 2, 3, 4, 5]
	The elements can be inserted into a list by squeezing them into an empty slice at the desired location.

SUMMARY

1. List structure and characteristics
2. Basic List operations
3. Positive & Negative Indexing
4. List slicing
5. Mutability

Thank You