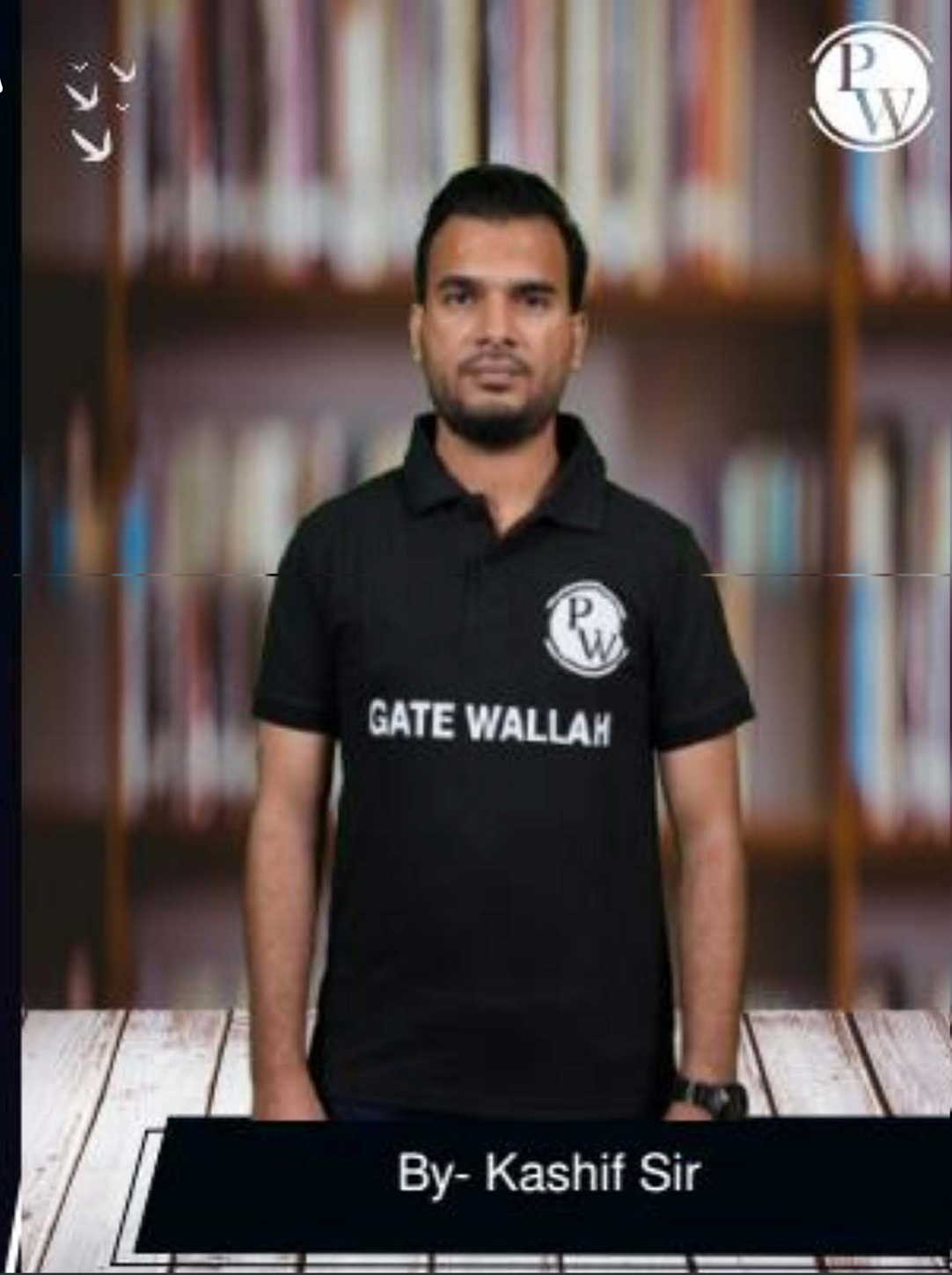
Data Science & Artificial Intelligence

Python for Data Science

Python Collections and String Handling

Lecture No. 03



By- Kashif Sir

RECAP

String methods

- strip
- rstrip
- lstrip
- join
- split
- replace
- islower
- isupper
- ...

List Initialization
Accessing of list
Adding the elements
(append, insert, extend)

Agenda -> List, Tuple

Topics *to be covered*

- Tuples, Sets, and Frozen Sets

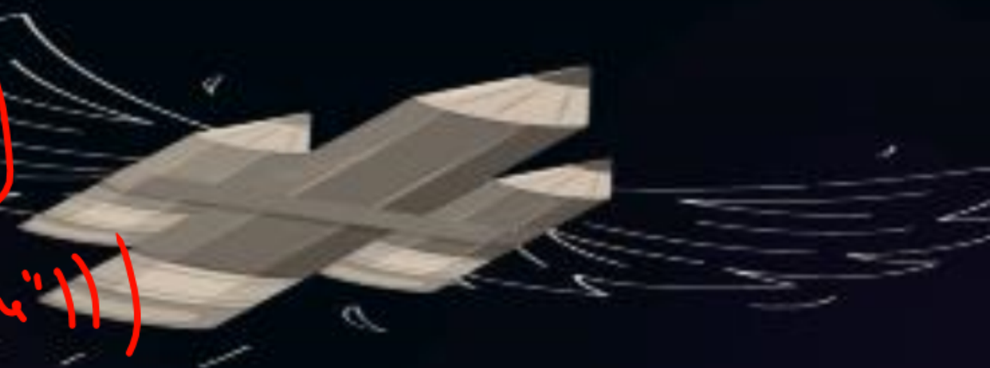
LIST

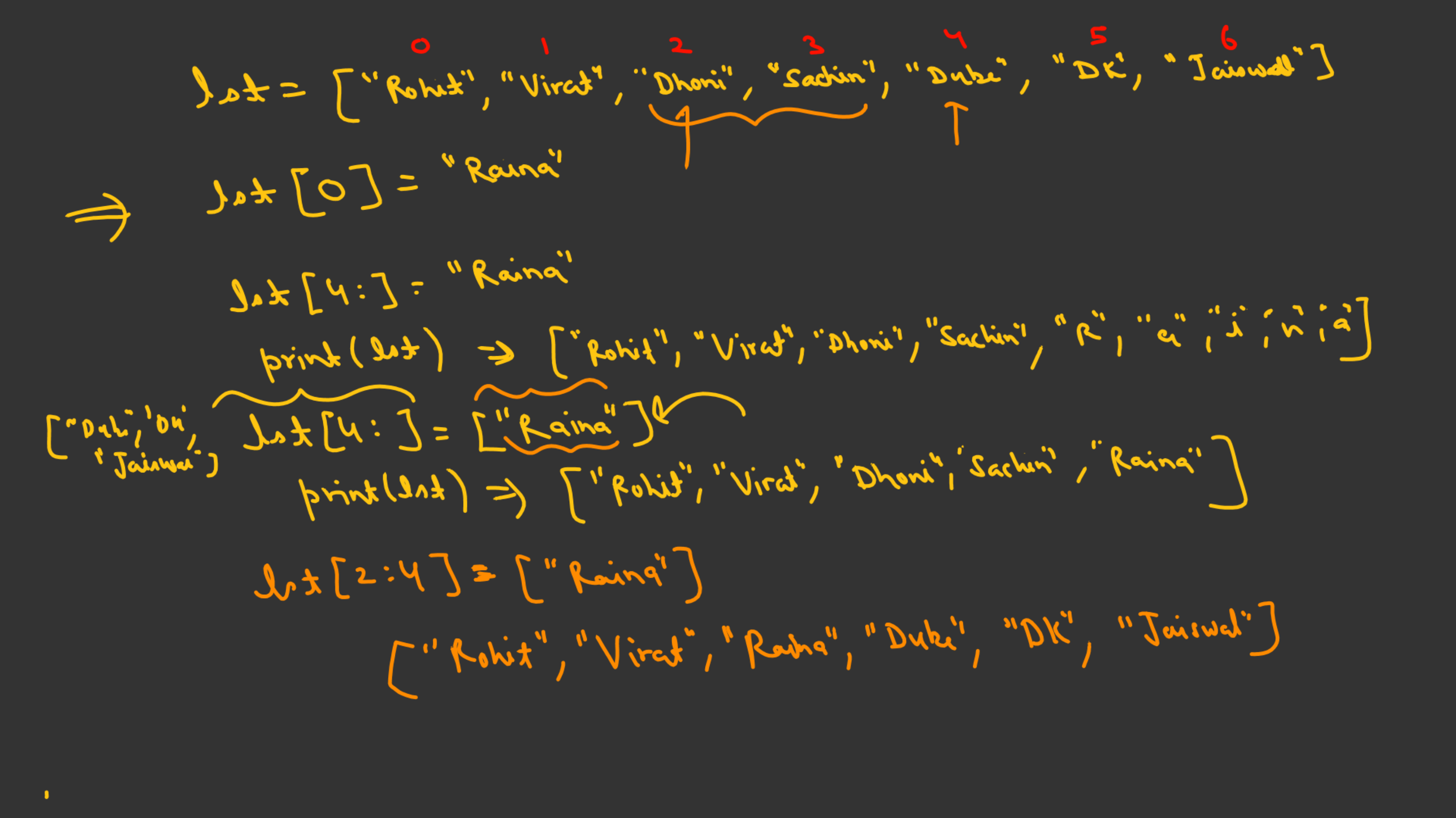
User input for List

① `a = input("Enter values").split()`

`1 2 3 4 5` = `"1 2 3 4 5"`

②
```
lst = []  a = ["1", "2", "3", "4", "5"]
n = int(input("Enter no. of elements"))
for i in range(n)
    lst.append(int(input("Enter value")))
print(lst)
```




0 1 2 3 4 5 6
lst = ["Rohit", "Virat", "Dhoni", "Sachin", "Dube", "DK", "Jaiswal"]
⇒ lst[0] = "Raina"
lst[4:] = "Raina"
print(lst) ⇒ ["Rohit", "Virat", "Dhoni", "Sachin", "R", "a", "i", "n", "a"]
["Dube", "DK", "Jaiswal"]
lst[4:] = ["Raina"]
print(lst) ⇒ ["Rohit", "Virat", "Dhoni", "Sachin", "Raina"]
lst[2:4] = ["Raina"]
["Rohit", "Virat", "Raina", "Dube", "DK", "Jaiswal"]

remove, del, clear, pop

remove ⇒ lst = [10, 20, 30, 40, 50]
indexes: 0, 1, 2, 3, 4 / -5, -4, -3, -2, -1

lst[1:-1]

lst.remove(30)
- 30 → value
- remove → first occurrence

lst.remove(60) → error

del ⇒ del lst[1] ⇒ [10, 30, 40, 50]
- 1 → index

del lst[1:3] ⇒ [10, 40, 50]

del lst[:] ⇒ print(lst) ⇒ []

del lst ⇒ print(lst) ⇒ error

clear ⇒

lst.clear() ⇒ print(lst) ⇒ []

pop ⇒ list = [10, 20, 30, 40, 50]

list.pop()

print (list.pop(3))

index

⇒ 40

List Comprehension (concise way to create list)

```
n = []
for i in range(1,101)
    n.append(i)
```

[1,2 ..... ,100] 1-100

```
n = [i for i in range(1,101)]
```

```
l = [i*2 for i in range(1,6)]
```

1,2,3,4,5

[2, 4, 6, 8, 10]

[expression for item in Iterable]

```
l = [i+5 for i in range(1,5)]
```

[6, 7, 8, 9]

```
lst = [ i for i in range(11) if i%2==0 ]
```

```
lst = [0, 2, 4, 6, 8, 10]
```

```
⇒ lst = [ i+5 if i%2==0 else i+10 for i in range(1,6) ]
```

```
lst = [11, 7, 13, 9, 15]
```

```
lst = [(i,j) for i in range(1,3) for j in rang(1,3) ]
```

(1,4) (1,2)

```
lst = [(1,1), (1,2), (2,1), (2,2)]
```

sort() sorted() reverse() reversed()

lst = [1, 8, 2, 7, 6, 5, 9, 12]

lst.sort() ⇒ Inplace sorting

print(lst) [1, 2, 5, 6, 7, 8, 9, 12]

lst.sort(reverse = True)

lst = ["a", "z", "c", "b"]

lst.sort() ⇒ ["a", "b", "c", "z"]

⇒ ["abcdef", "b"]

sorted(lst)

sorted(lst, reverse=True)

lst = ["1", "2", "abc", "c"]

lst = [1, 7, 9, 11]

lst.reverse() ⇒ In place

[11, 9, 7, 1]

lst[::-1]

reversed(lst)

TUPLE

Initialization

```
t = tuple()
t = ()
```

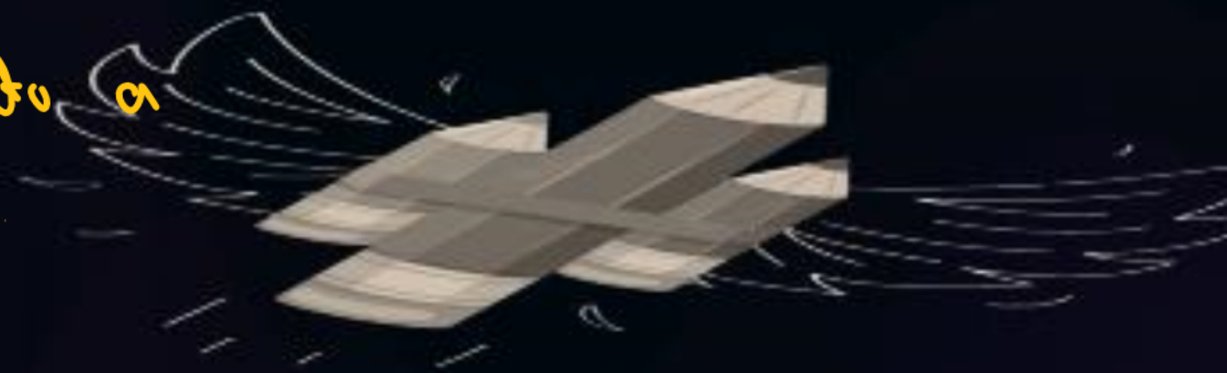
```
t1 = (1)
t2 = 1
t3 = 1,
t4 = (1,)
```

tuple

```
t5 = "a", "b", "c"
t6 = ("a", "b", "c")
```

Tuple Packing

The process of combining values into a tuple is called Tuple Packing



Tuple Unpacking

The process of extracting individual values of tuple into multiple variables.

```
a, b, c, d = (1, 2, 3, 4)
```

```
a, *b, c, d = (1, 2, 3, 4, 5, 6, 7)
```

a = 1
b = [2, 3, 4, 5]
c = 3
d = 4

```
a, b = (1, 2, 3, 4, 5, 6, 7)
```

```
a, *b = (1, 2, 3, 4, 5, 6, 7)
```

a = 1
b = [2, 3, 4, 5, 6, 7]

Accessing

```
t = tuple("python")
```

-6	-5	-4	-3	-2	-1
"p"	"y"	"t"	"h"	"o"	"n"
0	1	2	3	4	5

t=(1,4,9)

t[0] ⇒ 1

t[2] ⇒ 9

t[0] ⇒ "p"

t[-2] ⇒ "o"

```
l = list("python")
l[0] = "m"
print(l) => mython
t = tuple("python")
t[0] = "m"   # error
```

Slicing

t[1:4] ⇒ ("y", "t", "h")

t[:3] ⇒ ("p", "y", "t")

l = [1, 2, 3, 4, 5]

t = (1, 2, 3, l)

print(t) ⇒ (1, 2, 3, [1, 2, 3, 4, 5])

l[0] = 20

print(l) ⇒ [20, 2, 3, 4, 5]

print(t) ⇒ (1, 2, 3, [20, 2, 3, 4, 5])

t[3] = [4, 5, 6] ✗

t[3][1] = 100 ✓

a = [1, 2, 3]

{ b = [1, 2, 3]

c = b

b, c → [1, 2, 3]

a → [1, 2, 3]

t1 = (1, 2, 3)

{ t2 = (1, 2, 3)

t3 = t1

Update Tuple

```
lang = ("Java", "C", "Python")
lst = list(lang)
lst[1] = "C++"
```

Tuple → list

Update

tuple

sort() ⇒ inplace sorting

sorted()

```
lst = ["Java", "C++", "Python"]
lang = tuple(lst)
```

reversed()

THANK - YOU