LECTURE

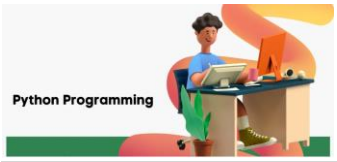
SEVEN

Tuples

- Tuples are similar to lists, except tuples are immutable (unchangeable). i.e., we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.
- Once a tuple is created you cannot delete or change the values of the items stored in it. You cannot add new values either.
- Tuples are also faster than lists.
- Python lists versus tuples:

Feature	List	Tuple
Mutability	mutable	immutable
Creation	`lst = [i, j]`	`tpl = (i, j)`
Element access	`a = lst[i]`	`a = tpl[i]`
Element modification	`lst[i] = a`	*Not possible*
Element addition	`lst += [a]`	*Not possible*
Element removal	`del lst[i]`	*Not possible*
Slicing	`lst[i:j:k]`	`tpl[i:j:k]`
Slice assignment	`lst[i:j] = []`	*Not possible*
Iteration	`for elem in lst:...`	`for elem in tpl:...`

- A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. The parentheses are optional, however, it is a good practice to use them.
- A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).
- We can access item in tuple using **indexing** and a **range** of characters using **slicing** with the **bracket** operator. **Index** starts from **0**. Trying to access a character out of index range will raise an **IndexError**. The index must be an **integer**. Python allows **negative** indexing for its sequences.



Example: Create Tuple, Tuple Indexing and Tuple Slicing

```
tp = (1, 2, 3.9, 4, "hi", 6, 7,8)      # Create a tuple
print(tp)                              # Print the tuple
print(tp[0])                           # Access an element (Positive Index)
print(tp[-2])                          # Access an element (Negative Index)

for i in tp:                 # Iterate over the elements of a tuple
        print(i, end=' ')
print(tp [2:5])      # Slice a tuple
```

Tuple Index

index	0	1	2	3	4	5	6	7
item	1	2	3.9	4	hi	6	7	8
index	-8	-7	-6	-5	-4	-3	-2	-1

Operations and Functions on Tuple

Tuple have their own set of permissible operations. In general, lists can be:

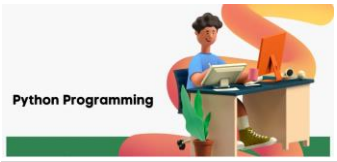
- Concatenated (joined): The **+** operator does this in Python. Simply to combine two tuple together.
- Replicated: The ***** operator can be used to repeats a tuple for the given number of times.
- python len function: it returns the number of items (length) in an object. use the len() to get the length of the given tuple.

Example: Concatenated Tuple and Replicated and len

```
f = (1, 3, 5)
k=f + (9, 7, 5)
r= f * 3
print(k)
print(r)
print(len(r))
```

Output

```
(1, 3, 5, 9, 7, 5)
(1, 3, 5, 1, 3, 5, 1, 3, 5)
9
```



Common Python Tuple Methods

There are numerous methods available with the tuple object, some of the commonly used methods are:

Method	Description
index()	Returns the index of the first matched item
count()	Returns the count of the number of items passed as an argument

Example: Common Python Tuple Methods

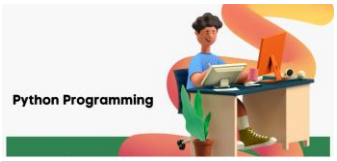
```
my_tuple = (3, 8, 1, 6, 0, 8, 4)
print(my_tuple.index(8))
print(my_tuple.count(8))
```

Output

1
2

Example (1): Write Python Program to read Tuple of numbers and print item of tuple

```
z=[]
n = int(input("Enter length of list:"))
for i in range(n):
    it = int(input("enter item"))
    z. append(it)
tp=tuple(z)
print(tp)
```



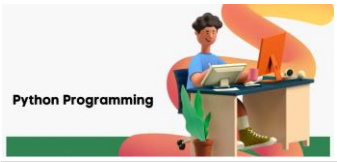
Example (2): Write Python Program to read tuple and find the count of each item in tuple

```
n=int(input("enter length= "))
ls=[]
for i in range(n):
        x=input("enter item =")
        ls.append(x)

tp=tuple(z)
for i in tp:
        print(i,tp. count(i))
```

Example (3): Write Python Program to read tuple and find last item in tuple

```
n=int(input("enter length= "))
ls=[]
for i in range(n):
        x=input("enter item =")
        ls.append(x)
tp=tuple(ls)
print("last item ",tp[-1])
```



Key Points to Remember:

Since tuples are quite similar to lists, both of them are used in similar situations. However, there are certain advantages of implementing a tuple over a list. Below listed are some of the main advantages:

- We generally use tuples for heterogeneous (different) data types and lists for homogeneous (similar) data types.
- Since tuples are immutable, iterating through a tuple is faster than with list. So there is a slight performance boost.
- Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.
- If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

Tuple	Lists
Tuple is immutable i.e. cannot be changed after assignment.	Lists are mutable i.e. can be changed.
Tuple uses Parenthesis for comma-separated values. (Optional Parenthesis)	Lists uses square brackets for comma-separated values. (Mandatory square brackets)
We cannot modify Tuples.	We can modify Lists.
Faster	Slower, compared to Tuple
Create a Tuple: mytuple = (1,25,100);	Create a List: list1 = [10, 25, 100]