PYTHON-TUPLE DATA STRUCTURE

1.Tuple data structure

a) We can create tuple data structure by using,

```
- Parenthesis () symbol.
- Predefined tuple(p) function.
```

b) A tuple can store group of objects or elements.

- A tuple can store same (Homogeneous) type of elements.
- A tuple can store different (Heterogeneous) type of elements.

c) In tuple insertion order is preserved or fixed.

- If we insert elements into 10, 20, 30 then output also will display as 10, 20, 30 then this is called as insertion order is preserved or fixed.

In [12]:
```
# example:
tuple=('10,20,30')
print('10,20,30')
```

```
10,20,30
```

1. Duplicate elements are allowed.
2. Tuple having immutable nature.
   - Immutable means once we create a tuple object then we cannot change or modifty the content of tuple object.

1.Store elements by using index.

- A tuple data structure supports both positive and negative indexes.
- Positive index means from left to right
- Negative index means right to left

Note:

```
- tuple is a predefined class in python
- once if we create tuple object means internally object is
creating for tuple class.
- inside tuple ever object can be separted by comma separator.
```

2. When should we go for tuple data structure?

- If we are going to define a data which never change over all the period, then we should go for tuple data structure.

example:

1. week days names
2. month names
3. year names

In [14]:
```
# tuple having same type of objects

employee_ids=(10,20,30,40,50)
print(employee_ids)
print(type(employee_ids))
```

```
(10, 20, 30, 40, 50)
<class 'tuple'>
```

3. syntax surprise 1:single value tuple

- If tuple having only one object, then that object should end with comma separator otherwise python internally not considered as it is tuple.

In [15]:
```
# a single value with tuple syntax,but it's not tuple

number=(9)

print(number)
print(type(number))
```

```
9
<class 'int'>
```

In [16]:
```
name=("saikiran")

print(name)
print(type(name))
```

```
saikiran
<class 'str'>
```

In [17]:
```
name=("saikiran",)

print(name)
print(type(name))
```

```
('saikiran',)
<class 'tuple'>
```

```
4.syntax suprise 2. parenthesis is optional for tuple
```

```
  - while creating a tuple paraenthesis is optional
```

In [18]:
```
# example:

emp_ids=10,20,30,40
print(emp_ids)
```

```
(10, 20, 30, 40)
```

5.Different ways to create a tuple

1.empty tuple

-we can create an empty tuple by using empty parenthesis.

In [19]:
```
# example:

emp_id=()
print(emp_id)
print(type(emp_id))
```

```
()
<class 'tuple'>
```

2. Tuple with group of values

- Tuple can contain group of objects; those objects can be same type or different type.

In [2]:
```
# example:

emp_id=(11,12,13)
std_id=120,130,140
print(emp_id)
print(std_id)
```

```
(11, 12, 13)
(120, 130, 140)
```

In [3]:
```
t=(11,12,13,"sai")
print(t)
```

```
(11, 12, 13, 'sai')
```

3.By using tuple(p) function

- we can create tuple by tuple(p)function.

In [4]:
```
a=[11,22,33]
t=tuple(a)
print(t)
```

```
(11, 22, 33)
```

6. Accessing elements of tuple:

- we can access tuple elements by using,
  - index
  - slice operter

6.1 Index

- index means position where element stores

In [5]:
```
t=(10,20,30,40,50,60)

print(t[0])       #   10
print(t[-1])      #   60
```

```
10
60
```

6.2.slice operator:

```
- A group of objects from starting point to ending point
```

In [6]:
```
t=(10,20,30,40,50,60)

print(t[2:5])
print(t[2:100])
print(t[::2])
```

```
(30, 40, 50)
(30, 40, 50, 60)
(10, 30, 50)
```

7.tuple vs immutability:

```
- tuple having immutable nature.
- if we create a tuple then we cannot modify the elements of
existing tuple.
```

In [7]:
```
t=(10,20,30,40)
print(t[1])
t[1]=70
```

```
20

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[7], line 3
      1 t=(10,20,30,40)
      2 print(t[1])
----> 3 t[1]=70

TypeError: 'tuple' object does not support item assignment
```

```
8. Mathematical operators on tuple:
```

```
    - we can apply plus (+) and multiplication (*) opertors on tuple.
    - + opertor works as concatenation.
    - * opertor works as multiplication.
```

In [ ]:
```
8.1.Concatenation operator (+):

    - + opertor concatenates two tuples and returns single tuple
```

In [8]:
```
t1=(10,20,30)
t2=(40,50,60)
t3=t1+t2

print(t3)
```

```
(10, 20, 30, 40, 50, 60)
```

Multiplication operator(*)

- multipilcation opertor works as repetition opertor

In [9]:
```
t1=(10,20,30)
t2=t1*3
print(t2)
```

```
(10, 20, 30, 10, 20, 30, 10, 20, 30)
```

len(p)function

- To return number of elements present in the tuple

In [10]:
```
t=(10,20,30,40)
print(len(t))
```

```
4
```

Method in tuple data structure

- as discussed,tuple is a predefined class.
- So,tuple class can contain methods because methods can be created inside of class only.
- we can check these methods by using dir(p) predefined function.
- so,internally tuple class contains two types of methods,
  - with underscore symbol methods.
    - we no need to focus
  - without underscore symbol methods.
    - we need to focus much on these

In [11]:
```
print(dir(tuple))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr_
_', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribut
e__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash_
_', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__l
t__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__re
pr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook
__', 'count', 'index']
```

Important point

- As per object-oriented principle,

```
- if we want to access instance method then we should
access by using object name.
- so,all tuple methods we can access by using tuple
object.
```

Methods in tuple

1. count(parameter1)method
2. index(parameter1)method

count(p) method

```
- count(p) is a method ,we should access this method by using
 tuple object.
- this method returns the number of occurrences of specified
 item in the tuple.
```

In [13]:
```
t=(10,20,10,10,20)
print(t.count(10))
```

```
3
```

index(P)method

- returns index of first occurrence of the given element.
- if the specified element is not available ,then we will get ValueError.

In [15]:
```
t=(10,20,30)
print(t.index(30))
```

```
2
```

In [16]:
```
t=(10,20,30)
print(t.index(88))
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[16], line 2
      1 t=(10,20,30)
----> 2 print(t.index(88))

ValueError: tuple.index(x): x not in tuple
```

In [ ]:
```
# can i add elements to this tuple t=(11,22,[33,44],55,66)
    - yes we can add elements to list in tuple.
    - in second index position list is available,to that we can add
```

In [17]:
```
t=(11,22,[33,44],55,66)

t[2].append(77)
print(t)
```

```
(11, 22, [33, 44, 77], 55, 66)
```

In [ ]:

In [ ]: