Tuple

1. Tuple is a Sequence of values, this values can be any type
2. Tuple are immutable once we create tuple modifications are not allowed.
3. Duplicates are allowed, Insertion order is preserved, occupies less space.
4. In Tuple index call and slicing is allowed.

```
#A tuple can be comma separated list of values/elements
tpl = 10, 20, 30, 40, 50, "One"
print(tpl) # (10, 20, 30, 40, 50, 'One')
```

```
#If we are using tuple not necessary to enclose tuples in parameters
tpl = (10, 20, 30, 40, 50, "One", "Two")
print(tpl) # (10, 20, 30, 40, 50, 'One', 'Two')
```

```
#Duplicates are allowed and Insertion Order is Preserved
tpl = (10, 20, 30, 40, 50, 10, 20, "One", "Two")
print(tpl) # (10, 20, 30, 40, 50, 10, 20 'One', 'Two')
```

```
#Use Comma if you are using single element/value then it treat as int
tpl = (20)
print(type(tpl)) #<class 'int'>

tpl = (20,)
print(type(tpl)) #<class 'tuple'>

#If there is no comma it treats as string
tpl = "One"
print(type(tpl)) # <class 'str'>
```

```
#Repeating the tuple
tpl = (10, 20, 30, 40)
print(tpl*2) # (10, 20, 30, 40, 10, 20, 30, 40)

#Repeating the list
lst = [10,20,30,40]
print(lst*2) # [10, 20, 30, 40, 10, 20, 30, 40]
```

```
#Count the repeated numbers
tpl = (20, 30, 20, 40, 20)
print(tpl.count(20)) # 3

# Find the index of tuple
tpl = (10, 20, 30, 40, 50)
#       0   1   2   3   4
print(tpl.index(20))  # 1 # index(value)
```

```
#Tuple is immutable

tpl = (10, 20, 30, 40)
tpl[2] = 50  # We get error
print(tpl)
```

Output

```
Traceback (most recent call last):
  File "F:\Data 03 Python\Data 01\Eclipse Workspace\Day03_Tuple\Ex3.py", line 4, in <module>
    tpl[2] = 50  # We get error
TypeError: 'tuple' object does not support item assignment
```

```
# tuple Slicing
tpl = 10, 20, 30, 40, 50, 60, 70
print(tpl[3:6]) # (40, 50, 60)
print(tpl[6]) # 70
print(tpl[3:]) # (40, 50, 60, 70)
```

```
#Break string into characters using list and tuple

x = list("Hello")
print(x) # ['H', 'e', 'l', 'l', 'o']
print(type(x)) # <class 'list'>

y = tuple("World")
print(y) # ('W', 'o', 'r', 'l', 'd')
print(type(y)) # <class 'tuple'>

z = []
z.extend("Hello")
print(z) # ['H', 'e', 'l', 'l', 'o']
```

```
# Creating copy of tuple in multiple ways
x = ("userOne", "userTwo", "userThree")
print(x) # ('userOne', 'userTwo', 'userThree')

# slicing tuple
print(x[:]) # ('userOne', 'userTwo', 'userThree')

# multiply tuple
print(x*1) # ('userOne', 'userTwo', 'userThree')

# concatenate tuple
print(x + ()) # ('userOne', 'userTwo', 'userThree')
```

```
# tuple Concatenation
t1 = (1,2,3,4)
print(t1 + (5,6)) # (1, 2, 3, 4, 5, 6)

t2 = (1,2,3,4)
print(t2 + (5,6) + (7,8)) # (1, 2, 3, 4, 5, 6, 7, 8)
```

```
# Tuple Repetition and Multiplication
t1 = (1,2,3,4,5)
t2 = (1,2,3,4,5)
print(t1*2) # (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
print(t1*3) # (1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
print(t1*t2) # TypeError: can't multiply sequence by non-int of type 'tuple'
```

UnPacking: we can use star before a variable to have a tuple of unpacking values

```
# unpacking tuple to different variables
t1, *t2 = (1,2,3,4,5)
print(t1) # 1
print(t2) # [2, 3, 4, 5]

t1, *t2, t3 = (1,2,3,4,5)
print(t1) # 1
print(t2) # [2, 3, 4]
print(t3) # 5

# packing tuple into a single variable
t = (1,2,3,4,5)
print(t) # (1, 2, 3, 4, 5)
```

```
t = tuple()
print(dir(t))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```