Tuples are ordered collections of heterogeneous data that are unchangeable.

Tuple has the following characteristics:

1. Ordered
2. Immutable
3. Heterogenous
4. Duplicates

Creating a tuple

1. Using Paranthesis()
2. Using tuple() constructor

In [ ]:

```
number_tuple = (10, 20, 25.75)
print(number_tuple)
```

```
(10, 20, 25.75)
```

In [ ]:

```
sample_tuple = ('Sanvee', 30, 45.75)
print(sample_tuple)
```

```
('Jessa', 30, 45.75)
```

In [ ]:

```
tuple3 = (True, False, False)
print(tuple3)
```

```
(True, False, False)
```

In [ ]:

```
sample_tuple2 = tuple(('sanvee', 30, 45.75))
type(sample_tuple2)
```

Out[7]:

```
tuple
```

In [ ]:

```
print(sample_tuple2)
```

```
('sanvee', 30, 45.75)
```

In [ ]:

```
#without comma
single_tuple = ('Hello')
print(type(single_tuple))
# Output class 'str'
print(single_tuple)
```

```
<class 'str'>
Hello
```

In [ ]:

```
# with comma
single_tuple1 = ('Hello',)
# output class 'tuple'
print(type(single_tuple1))
# Output ('Hello',)
print(single_tuple1)
```

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

In [ ]:

```
#duplicates
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry', 'apple', 'cherry')
```

In [ ]:

```
tuple1 = ('P', 'Y', 'T', 'H', 'O', 'N')
# length of a tuple
print(len(tuple1))
```

```
6
```

Access Tuple Items

In [ ]:

```
#positive indexing
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

```
banana
```

In [ ]:

```
#negative indexing
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

```
cherry
```

In [ ]:

```
#Slicicng
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
```

```
('cherry', 'orange', 'kiwi')
```

In [21]:

```
tuple1 = (10, 20, 30, 40, 50)

# get index of item 30
position = tuple1.index(30)
print(position)
```

```
2
```

In [20]:

```
#Immutable
thistuple[2] = "Muskmelon"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-b207f7da3410> in <module>()
      1 #Immutable
----> 2 thistuple[2] = "Muskmelon"

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

In [23]:

```
tuple1 = (10, 20, 30, 40, 50, 60, 70, 80)
# checking whether item 50 exists in tuple
print(50 in tuple1)
print(500 in tuple1)
```

```
True
False
```

Tuple methods and Functions used with Tuple

we can convert the tuple to a list, add items, and then convert it back to a tuple.

In [24]:

```
tuple1 = (0, 1, 2, 3, 4, 5)

# converting tuple into a list
sample_list = list(tuple1)
# add item to list
sample_list.append(6)
```

In [25]:

```
# converting list back into a tuple
tuple1 = tuple(sample_list)
print(tuple1)
```

```
(0, 1, 2, 3, 4, 5, 6)
```

In [30]:

```
#list in tuple can modify
tuple1 = (10, 20, [25, 75, 85])
# before update
print(tuple1)

# modify last item's first value
tuple1[2][0] = 250
# after update
print(tuple1)
```

```
(10, 20, [25, 75, 85])
(10, 20, [250, 75, 85])
```

Removing items from a tuple

In [31]:

```
#The del keyword will delete the entire tuple.

sampletup1 =(0,1,2,3,4,5,6,7,8,9,10)
del sampletup1

print(sampletup1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-31-2b4e408d48d3> in <module>()
      4 del sampletup1
      5
----> 6 print(sampletup1)

NameError: name 'sampletup1' is not defined
```

In [32]:

```
tuple1 = (0, 1, 2, 3, 4, 5)

# converting tuple into a list
sample_list = list(tuple1)
# reomve 2nd item
sample_list.remove(2)

# converting list back into a tuple
tuple1 = tuple(sample_list)
print(tuple1)
```

```
(0, 1, 3, 4, 5)
```

In [33]:

```
tuple1 = (10, 20, 60, 30, 60, 40, 60)
# Count all occurrences of item 60
tuple1.count(60)
```

Out[33]:

```
3
```

In [35]:

```
tuple1 = (1, 2, 3, 4, 5)
tuple2 = (3, 4, 5, 6, 7)

# concatenate tuples using + operator
tuple3 = tuple1 + tuple2
print(tuple3)
```

```
(1, 2, 3, 4, 5, 3, 4, 5, 6, 7)
```

In [34]:

```
tuple1 = (1, 2, 3, 4, 5)
tuple2 = (3, 4, 5, 6, 7)

# using sum function
tuple3 = sum((tuple1, tuple2), ())
print(tuple3)
```

```
(1, 2, 3, 4, 5, 3, 4, 5, 6, 7)
```

In [36]:

```
tuple1 = (11, 22, 10, 4)
# The Maximum value in a integer tuple
print(max(tuple2))
```

```
22
```

In [38]:

```
tuple2 = ('xyz', 'zara', 'abc')
# The Maximum value in a string tuple
print(max(tuple1))
```

```
zara
```

In [39]:

```
print(min(tuple1))
```

```
abc
```

In [40]:

```
print(min(tuple2))
```

```
4
```

In [42]:

```
#can't use on heterogenous tuple
tuple3 = ('a', 'e', 11, 22, 15)
# max item
print(max(tuple3))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-42-5f761f8cc7d6> in <module>()
      2 tuple3 = ('a', 'e', 11, 22, 15)
      3 # max item
----> 4 print(max(tuple3))

TypeError: '>' not supported between instances of 'int' and 'str'
```

1. There are no append() or extend() to add items and similarly no remove() or pop() methods to remove items. This ensures that the data is write-protected. As the tuples are Unchangeable, they can be used to represent read-only or fixed data that does not change.
2. As they are immutable, they can be used as a key for the dictionaries, while lists cannot be used for this purpose.
3. As they are immutable, the search operation is much faster than the lists. This is because the id of the items remains constant.
4. Tuples contain heterogeneous (all types) data that offers huge flexibility in data that contains combinations of data types like alphanumeric characters.

In [ ]:

```
```