Tuple items are ordered, unchangeable, and allow duplicate values.

In [83]:
```
a=("Digicomm","Semiconductor")
print(type(a))
```

<class 'tuple'>

In [46]:
```
a=("digicomm")
print(type(a))
```

<class 'str'>

In [48]:
```
a=("digicomm",)
print(type(a))
```

<class 'tuple'>

In [63]:
```
dept = ("dv", "pd", "psv")
mytuple = dept * 2
print(mytuple)
```

('dv', 'pd', 'psv', 'dv', 'pd', 'psv')

In [56]:
```
a=tuple(1,2,3)) # we have to use two brackets while declaration
print(type(a))
```

<class 'tuple'>

In [7]:
```
# note the double round-brackets
#otherwise it shows error
a=tuple(1,2,3)
print(type(a))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 3
      1 # note the double round-brackets
      2 #otherwise it shows error
----> 3 a=tuple(1,2,3)
      4 print(type(a))

TypeError: tuple expected at most 1 argument, got 3
```

In [85]:
```
a=tuple((1,))
print(type(a))
```

<class 'tuple'>

In-built Functions Meaning

```
    len() Returns the number of elements in the tuple
    max() Returns the element with the greatest value
    min() Returns the element with the minimum value
    sum() Returns the sum of all the elements of tuple
    index(x) Returns the index of element x
    count(x) Returns the number of occurrence of element x
```

In [50]:
```
thistuple = ("dv", "pd", "psv")
print(len(thistuple))
```

3

In [11]:
```
l=(1,3,6,3,8)
print("min value: ",max(l))
print("min value: ",min(l))
```

max value: 8
min value: 1

In [51]:
```
# indexing for accessig the tuple
thistuple = ("dv", "pd", "psv")
print(thistuple[1])
print(thistuple[-1])

pd
psv
```

In [65]:
```
#The index() method finds the first occurrence of the specified value.
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)

x = thistuple.index(7)

print(x)
```

2

In [66]:
```
# ranging
thistuple = ("dv", "pd", "psv", "analog", "dft", "mod", "pv")
print(thistuple[2:5])
print(thistuple[:4])
print(thistuple[2:])
print(thistuple[-4:-1])
```

('psv', 'analog', 'dft')
('dv', 'pd', 'psv', 'analog')
('psv', 'analog', 'dft', 'mod', 'pv')
('analog', 'dft', 'mod')

In [70]:
```
thistuple = ("dv", "pd", "psv", "analog", "dft", "mod", "pv")
thistuple[1]="design"
print(thistuple)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[70], line 2
      1 thistuple = ("dv", "pd", "psv", "analog", "dft", "mod", "pv")
----> 2 thistuple[1]="design"
      3 print(thistuple)

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

In [87]:
```
# for adding the elements, we have to convert the tuple into list
thistuple = ("dv", "pd", "psv")
y = list(thistuple)
y.append("dft")
thistuple = tuple(y)
print(thistuple)
```

('dv', 'pd', 'psv', 'dft')

In [88]:
```
used can delete
thistuple = ("dv", "pd", "psv")
y = ("dft",)
thistuple += y
print(thistuple)
```

('dv', 'pd', 'psv', 'dft')

In [57]:
```
# remove
thistuple = ("dv", "pd", "psv")
y = list(thistuple)
y.remove("psv")
thistuple = tuple(y)
print(thistuple)
```

('pd', 'psv')

In [59]:
```
thistuple = ("dv", "pd", "psv")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[59], line 3
      1 thistuple = ("dv", "pd", "psv")
      2 del thistuple
----> 3 print(thistuple)

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

When we create a tuple, we normally assign values to it. This is called "packing" a tuple.But, in Python, we are also allowed to extract the values back into variables. This is called "unpacking"

In [86]:
```
#packed
dept = ("dv", "pd", "psv")
print(dept)
```

('dv', 'pd', 'psv')

In [87]:
```
# un-packed
dept = ("dv", "pd", "psv")

(verfication, pysical_design, validation) = dept
#(verfication, pysical_design, validation) = ("dv", "pd", "psv")
print(verfication)
print(pysical_design)
print(validation)

dv
pd
psv
```

In [60]:
```
# un-packed
dept = ("dv", "pd", "psv")

verfication, pysical_design, validation = dept

print(verfication)
print(pysical_design)
print(validation)

dv
pd
psv
```

In [90]:
```
#Using Asterisk*
'''If the number of variables is less than the number of values, you can add an * to the variable name
and the values will be assigned to the variable as a list!'''

fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")

(green, yellow, *red) = fruits
#(green, yellow, red) = fruits #too many values to unpack (expected 3)
```

In [26]:
```
fruits = ("apple", "mango", "papaya", "pineapple", "cherry")

(green, *tropic, red) = fruits

print(green)
print(tropic)
print(red)
```

apple
['mango', 'papaya', 'pineapple']
cherry

In [27]:
```
#join to tuples
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

('a', 'b', 'c', 1, 2, 3)

In [93]:
```
t=[(1, "Amit"),(2, "Divya"),(3, "Sameer")]
for i in t:
    print(i)
    print(type(i))
```

(1, 'Amit')
<class 'tuple'>
(2, 'Divya')
<class 'tuple'>
(3, 'Sameer')
<class 'tuple'>

In [96]:
```
# ZIP function:
''' It takes items in sequence from a number of collections
to make a list of tuples, where each tuple contains one item from each of the collections. The
function is often used to group items from a list which has the same index.'''
A1=[1,2,3]
A2="XYZ"
A3="Hoo!"
A3=[3,4,5]
A4=[6,7,8]
A5=(4,6,8)
AB=list(zip(A0,A7))
AB=list(zip(A2,A2))
print("string",AB)
print("tuples",AB)
AA=list(zip(A1,A2))
print(AA)
AB=list(zip(A1,A3))
print(AB)

string [('X', 'X'), ('Y', 'Y'), ('Z', 'Z')]
tuples [(1, 4), (3, 6), (5, 8)]
[(1, 'X'), (2, 'Y'), (3, 'Z')]
[(1, 3), (2, 4), (3, 5)]
```

In [137]:
```
LS=['Laptop', 'Desktop', 'Mobile'] #Create List1
L2=(40000, 30000, 15000) #Create List2
LS=tuple(list(zip(LS,L2)))) #Group item from Lists 1 and 2
print(LS)
```

(('Laptop', 40000), ('Desktop', 30000), ('Mobile', 15000))

In [103]:
```
LS=['Black','White','Grey'] #Create List L1
L2=[255,5,100] #Create List L2
for Colour, Code in zip(LS,L2): #Use of zip in for loop
    print(Colour,Code)
```

('Black', 255)
('White', 5)
('Grey', 100)

In [97]:
```
M=[('APPLE',50000),('DELL',30000)] #List of tuples
Laptop,Price=zip(*M) #Unpacking Values
#print(*M)
print(Laptop)
print(Price)
#print(type(*M))
```

('APPLE', 'DELL')
(50000, 30000)

The function zip*() also performs the same operation, i.e. unpacks a sequence into positional arguments.

In [94]:
```
#Transpose of a matrix
Matrix=[(1,2),(3,4),(5,6),(7,8)]
x=zip(*Matrix)
print(tuple(x))
```

((1, 3, 5, 7), (2, 4, 6, 8))

In [62]:
```
# Write a program to demonstrate the return multiple values to the caller.
def Return_Multiple_Values(a,b):
    sum1 = a + b
    diff = a - b
    mul = a * b
    div = a / b
    return sum1,diff,mul,div

sum, r Return_Multiple_Values(20,10)
print(sum)
print(type(sum))
```

(30, 10, 200, 2.0)
<class 'tuple'>

In [64]:
```
a = ((1,2),)*2
print(a)
```

((1, 2), (1, 2), (1, 2), (1, 2), (1, 2), (1, 2), (1, 2), (1, 2))

In [69]:
```
# Tuple Comparison

tuple1 = (1,2,3,4,5)
tuple2 = (1,2,3,4,5)
tuple3 = ("A", "AB", "a", "ab")
tuple4 = ("a", "AAB")


# Integer Equality Operator(== & !=)
if tuple1 == tuple2:
    print("Tuple1 and Tuple2 are same")
elif tuple1 != tuple2:
    print("Tuples and Tuple2 are different")
else:
    print("ERROR")
```

Tuple1 and Tuple2 are same

```
if tuple1 > tuple2:
    print("Tuple1 is greater than Tuple2")
elif tuple1 < tuple2:
    print("Tuple2 is greater than Tuple1")
elif tuple1 >= tuple2:
    print("Tuple1 is greater than equal to Tuple2")
elif tuple1 <= tuple2:
    print("Tuple1 is less than equal to Tuple2")
else:
    print("ERROR")
```

Tuple1 is greater than equal to Tuple2

```
if tuple3 == tuple4:
    print("Tuple3 and Tuple4 are same")
elif tuple3 != tuple4:
    print("Tuple3 and Tuple4 are different")
else:
    print("ERROR")
```

Tuple3 and Tuple4 are different

```
# String Greater than and Less than
if tuple3 > tuple4:
    print("Tuple3 is greater than Tuple4")
elif tuple3 < tuple4:
    print("Tuple3 is greater than Tuple4")
elif tuple3 >= tuple4:
    print("Tuple3 is greater than equal to Tuple4")
elif tuple3 <= tuple4:
    print("Tuple3 is less than equal to Tuple4")
else:
    print("ERROR")
```

Tuple3 is greater than Tuple4

```
# Tuple Functions


inp_tuple = (12,4,2,5,13,2,7,1)

# index(Number whose index we want)
print("Index: ", inp_tuple.index(5))

# count(The number we want to count)
print("Count: ", inp_tuple.count(2))

# len(tuple)
print("Length: ", len(inp_tuple))

# max(tuple)
print("Max Value: ", max(inp_tuple))

# min(tuple)
print("Min Value: ", min(inp_tuple))

# sum(tuple)
print("Sum of element: ", sum(inp_tuple))
```

Index: 3
Count: 2
Length: 8
Max Value: 13
Min Value: 1
Sum of element: 46

In [98]:
```
# Generating a tuple using for-loop

tuple_size = int(input("Enter the size of the tuple: "))

inp_tuple = ()
for i in range(tuple_size):
    inp_tuple = inp_tuple + (i,)

print(inp_tuple)# Generating a tuple using for-loop
```

(0, 1, 2, 3)

In [81]:
```
import random
tuple_size = int(input("Enter the size of the tuple: "))

inp_tuple = ()
for i in range(tuple_size):
    inp_tuple = inp_tuple + (random.randrange(1,100),)
print(inp_tuple)
```

(2, 45, 82, 44)

In [82]:
```
tuple_size = int(input("Enter the size of the tuple: "))

inp_tuple = ()
for i in range(tuple_size):
    element=(input("enter the tuple element : ")
    inp_tuple =inp_tuple + (element,)
print(inp_tuple)
```

```
{'1', '4', '5', '8', '9'}
```

In [ ]: