Tuple

In [ ]:
```
1. Tuple is a built-in collection data type.
2. Tuples are used to store multiple items in a single variable.
3. Tuples are written with round brackets.
                eg : t = (1,2,3,4)
```

In [2]:
```
#Create a Tuple

t =(1,2,3,4,5)
print("Example Tuple:",t)
```

```
Example Tuple: (1, 2, 3, 4, 5)
```

Characteristics of a Tuple

In [ ]:
```
Tuple items are :
1. Ordered                  : Items in a tuple have a defined order, and that order will not change.
2. Unchangeable (Immutable) : we cannot change, add or remove items after the tuple has been created.
3. Allow duplicate values   : Allow same values more than one time since tuple is indexed.

Tuple items are indexed, the first item has index [0], the second item has index [1] etc.
```

In [4]:
```
#Tuple are ordered.
t =(10,20,30,40)
print(t) #Print the items in the same order.
```

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

In [5]:
```
#allow duplicates
t=(10,20,30,40,10,20,30)
print(t) #allow duplicates
```

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

Tuple Length

In [ ]:
```
len() :Method to find the length of a tuple.Length means the no. of items in a tuple.
```

In [6]:
```
#Length of a tuple
t = (1,2,3,4,5)
print("Length of tuple is: ",len(t))
```

```
Length of tuple is:  5
```

Create Tuple With One Item

In [ ]:
```
To create a tuple with only one item, you have to add a comma after the item,
otherwise Python will not recognize it as a tuple.
```

In [11]:
```
#Tuple with one item
t =("Welcome",)
t1=("welcome")
print(type(t1)) # consider as string
print(type(t)) # consider as tuple
print(t)
```

```
<class 'str'>
<class 'tuple'>
('Welcome',)
```

Tuple Items - Data Types

In [ ]:
```
Tuple items can be of any data type such as integer, float, string etc.
Tuple can have different datatypes as their elements.

type() : Method to find the type of a variale
```

In [12]:
```
#tuple with different data items.
t = (1,2,"Welcome",5,"hello",67.90)
print(t)
```

```
(1, 2, 'Welcome', 5, 'hello', 67.9)
```

In [13]:
```
#type() method to determine datatype
t = ("welcome","to","tuple")
print(type(t)) # t is a tuple
```

```
<class 'tuple'>
```

The tuple() Constructor

In [ ]:
```
tuple() : Constructor of class Tuple, used to make a tuple.
```

In [14]:
```
#tuple constructor
l = [1,2,3,4,5]
t =tuple(l)
print(t) # list turned to a tuple
print(type(t))
```

```
(1, 2, 3, 4, 5)
<class 'tuple'>
```

Tuple Methods

In [ ]:
```
Python has two built-in methods that you can use on tuples:

Method                          Description

count()     Returns the number of times a specified value occurs in a tuple
index()     Searches the tuple for a specified value and returns the position of where it was found.
```

Access Tuple Items

We can access tuple items by referring to the index number, inside square brackets.

In [1]:
```
#Access tuple items.
t = (1,2,3,4,5)
print(t[0]) #access first item
print(t[1]) #access second item
print(t[4]) #access fifth item
```

```
1
2
5
```

Negative Indexing

Negative indexing means start from the end. -1 refers to the last item, -2 refers to the second last item etc.

In [2]:
```
#Negative indexing
t = (1,2,3,4,5)
print(t[-1]) #access last item
print(t[-2]) #access second last item
print(t[-5]) #access first item
```

```
5
4
1
```

Range of Indexes

Specify a range of indexes by specifying where to start and where to end the range. When specifying a range, the return value will be a new tuple with the specified items.

In [6]:
```
# To return a specified range of items from list
t = (2,4,6,8,10,12)
print(t[2:5]) # range start at index 2 (included) end at index 5(not included)

print(t[:4]) #Start from the 0th index end at index 4(not included)
             # By leaving out the start value, the range will start at the first item

print(t[3:])  #Start from the 3rd index to the end of tuple.
              # By leaving out the end value, the range will go on to the end of the tuple
```

```
(6, 8, 10)
(2, 4, 6, 8)
(8, 10, 12)
```

Range of Negative Indexes

Specify negative indexes if you want to display from the end of the tuple.

In [1]:
```
## To return a specified range of items from the end of the list
t = (2,4,6,8,10,12)
print(t[-5:-1]) #start from index -5 and print up to index -2
```

```
(4, 6, 8, 10)
```

Check if Item Exists

In [ ]:
```
in : keyword to determine a specified item is present in a tuple.
```

In [5]:
```
#check if an item present or not in a given list.

t = (12,14,16,18,20)
if 14 in t:
    print("yes,item present in the list")
else:
    print("no,item not present in the list")
```

```
yes,item present in the list
```

Update Tuples

Tuples are unchangeable, meaning that you cannot change, add, or remove items once the tuple is created. But there is a solition if you want to update a tuple , that is You can convert the tuple into a list, change the list, and convert the list back into a tuple.

In [7]:
```
#converting a tuple to list to change an item in the tuple.
t = (2,4,6,8)
l =list(t)
l[1] =40
t = tuple(l)
print(t)
```

```
(2, 40, 6, 8)
```

Add Items

We are allowed to add tuples to tuples,if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple.

In [8]:
```
#adding one tuple to another

t1 = (1,2,3,4)
t2 = (5,6,7,8)
t1+=t2 # t1 = t1+t2
print(t1)
```

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

Delete a Tuple

In [ ]:
```
del - Keyword used to delete a tuple.
```

In [11]:
```
#Delete a tuple
t = (1,2,3,4)
print(t) # print tuple t
del t
print(t) # this shows error since tuple t was deleted
```

```
(1, 2, 3, 4)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 5
      3 print(t)
      4 del t
----> 5 print(t)

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

Unpack Tuples

In [ ]:
```
Packing a tuple   : Assigning values to tuple when we create it.
Unpacking a tuple : Unpacking allowes to extract the values back into variables.
                    The number of variables must match the number of values in the tuple,
                    if not, you must use an asterisk to collect the remaining values as a list

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
```

In [12]:
```
#Unpacking a tuple
fruits = ("Mango","Orange","Apple")
(yellow,orange,red) = fruits
print(yellow)
print(orange)
print(red)
```

```
Mango
Orange
Apple
```

In [26]:
```
#Unpacking a tuple using *

fruits = ("Mango","Orange","Apple","tomato","Cherry","rasberry")
#(yellow,orange,red) = fruits #Here no.of varialbe less than no.of items in the tuple

(yellow,orange,*red) = fruits # So use astric with variale red so remaining items in the tuple stored in variable 'red'

#(*yellow,orange,red) = fruits #if you want variable 'yellow' as a list

#(yellow,*orange,red) = fruits #if you want variable 'orange' as a list

print("yellow = ",yellow)
print("orange = ",orange)
print("red = ",red)
```

```
yellow =  Mango
orange =  ['Orange', 'Apple', 'tomato', 'Cherry']
red =  rasberry
```

Loop Tuples

We can use for loop to iterate through tuple items.

In [27]:
```
#loop through tuples

t = (11,22,33,44,55)
for i in t:
    print(i)
```

```
11
22
33
44
55
```

Loop through index numbers

We can loop through index numbers using range() and len() method.

In [29]:
```
#Loop through index numbers.

t = (11,22,33,44,55)
```

```
for i in range(len(t)):
    print(i)
```

```
0
1
2
3
4
```

Using a While Loop

In [31]:
```
#iterate through tuple items using while loop

t = (11,22,33,44,55)
i=0
while(i<len(t)):
    print(t[i])
    i+=1
```

```
11
22
33
44
55
```

In [32]:
```
#iterate through tuple items' indexes using while loop

t = (11,22,33,44,55)
i=0
while(i<len(t)):
    print(i)
    i+=1
```

```
0
1
2
3
4
```

Join Tuples

In [ ]:
```
1. '+' operator  : Join two tuples
2. '*' operator  :Multiply the content of a tuple a given number of times.
```

In [34]:
```
#Join two tuples using '+'

t1 = (1,2,3,4)
t2 = (5,6,7,8)
t3 = t1 + t2
print(t3)
```

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

In [38]:
```
#Multiply the content of a tuple a given number of times using '*'

t1 = (1,2,3,4)
t2 =("Apple","Orange","Grapes")
t3 = t1*3
t4 = t2*2
print(t3)
print(t4)
```

```
(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)
('Apple', 'Orange', 'Grapes', 'Apple', 'Orange', 'Grapes')
```

End