Python - Tuples

Tuple is data structure that holds collection of different objects in it. For example, a tuple can hold numerical as well as string values in it along with the other collections like list, tuples etc.

Creation

Empty tuple can be created with angular brackets.

In [36]:

```
tuple1 = ()
print("tuple1----> ", tuple1)
```

```
tuple1---->  ()
```

Tuple elements are enclosed within angular brackets.

In [4]:

```
my_tuple = ('a', 'b', 'c', 'd', 'e')
my_tuple
```

Out[4]:

```
('a', 'b', 'c', 'd', 'e')
```

Tuple can hold elements belonging to the different data types.

In [5]:

```
my_mixed_tuple = ('a', 1, 1.1, True)
my_mixed_tuple
```

Out[5]:

```
('a', 1, 1.1, True)
```

Type of tuple variable is tuple.

In [6]:

```
print(type(my_tuple))
print(type(my_mixed_tuple))
```

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

Type of individual elements can be same or different which can be checked as follows.

In [8]:

```
type(my_tuple[0])
```

Out[8]:

```
str
```

In [10]:

```
type(my_mixed_tuple[1])
```

Out[10]:

```
int
```

Size of the tuple can be determined using the len() funtion

In [12]:

```
len(my_tuple)
```

Out[12]:

```
5
```

In [13]:

```
len(my_mixed_tuple)
```

Out[13]:

```
4
```

Indexing

Tuple elements can be accessed using the bracket operator. The index value starts from 0 and goes upto len() -1 .

In [15]:

```
print("First element ---> " , my_mixed_tuple[0], "      ", "type of element ----> ", type(m
print("Second element ---> " , my_mixed_tuple[1], "      ", "type of element ----> ", type(
print("Third element ---> " , my_mixed_tuple[2], "      ", "type of element ----> ", type(m
print("Fourth element ---> " , my_mixed_tuple[3], "      ", "type of element ----> ", type(
```

```
First element --->  a        type of element ---->  <class 'str'>
Second element --->  1        type of element ---->  <class 'int'>
Third element --->  1.1        type of element ---->  <class 'float'>
Fourth element --->  True        type of element ---->  <class 'bool'>
```

-ve indexing also possible on the tuples elements.

In [17]:

```
print("Last element ---> " , my_mixed_tuple[-1])
print("Second Last element ---> " , my_mixed_tuple[-2])
print("Third element from Last---> " , my_mixed_tuple[-3])
print("Fourth element from Last---> " , my_mixed_tuple[-4])
```

```
Last element --->  True
Second Last element --->  1.1
Third element from Last--->  1
Fourth element from Last--->  a
```

Slicing

The part of tuples can be accessed using the slicing.

In [19]:

```
my_mixed_tuple[ 0 : 2]   # extracts first two elements of tuple
```

Out[19]:

```
('a', 1)
```

In [21]:

```
my_mixed_tuple[ 1 : ]   # extracts all elements of tuple starting from second element
```

Out[21]:

```
(1, 1.1, True)
```

In [22]:

```
my_mixed_tuple[ : 3]   # extracts first three elements of tuple
```

Out[22]:

```
('a', 1, 1.1)
```

In [24]:

```
my_mixed_tuple[ 0 : : 2] # extract every alternate element of tuple
```

Out[24]:

```
('a', 1.1)
```

In [26]:

```
my_mixed_tuple[ : : -1 ]  #access the elements of tuple in reverse manner
```

Out[26]:

```
(True, 1.1, 1, 'a')
```

Concatenation

Two tuples can be joined using the '+' operator.

In [29]:

```
tuple1 = ('1', '2', '3', '4')
tuple2 = ('5', '6')
new_tuple = tuple1 + tuple2  # concat two tuples using +
new_tuple
```

Out[29]:

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

In [30]:

```
tuple1 = ('1', '2', '3', '4')
new_tuple = tuple1 + ('5', '6')  # concat two tuples using +
new_tuple
```

Out[30]:

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

Repetition

Tuple elements can be repeated using "*" operator.

In [99]:

```
(1, 2) * 3  # create a tuple with elements '1' and '2' repeated three times
```

Out[99]:

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

In [103]:

```
my_tuple = ( 'a', 'b')
new_tuple = my_tuple * 2   #create a tuple with elements 'a' and 'b' repeated two times
new_tuple
```

Out[103]:

```
('a', 'b', 'a', 'b')
```

Immutability

Tuple elements can not be altered.

In [32]:

```
my_tuple = ('a', 'b', 'c', 'd', 'e')
```

In [34]:

```
my_tuple[0] = 'p'    #throws error as individual tuple elements can not be altered
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-34-74504a297aef> in <module>
----> 1 my_tuple[0] = 'p'    #throws error as individual tuple elements can
 not be altered

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

New elements can not be added to the tuple using append().

In [43]:

```
my_tuple.append('f')   #throws error as tuple can not be altered
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-43-dbe5b72caaed> in <module>
----> 1 my_tuple.append('f')   #throws error as tuple can not be altered

AttributeError: 'tuple' object has no attribute 'append'
```

In [45]:

```
my_tuple.remove(0)   # elements can not be removed from the tuple
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-45-113a1b74b409> in <module>
----> 1 my_tuple.remove(0)   # elements can not be removed from the tuple

AttributeError: 'tuple' object has no attribute 'remove'
```

Tuple can be assigned to another tuple variable.

In [46]:

```
my_tuple = ('a', 'b', 'c', 'd', 'e')
new_tuple = my_tuple

print("my_tuple---->", my_tuple)
print("new_tuple---->", new_tuple)
```

```
my_tuple----> ('a', 'b', 'c', 'd', 'e')
new_tuple----> ('a', 'b', 'c', 'd', 'e')
```

Operations

In [80]:

```
my_tuple = ('a', 'b', 'c', 'c', 'e')
```

index() can be used to get index of first instance of element value.

In [58]:

```
my_tuple.index('c')  #get first index of element 'c' in tuple
```

Out[58]:

```
2
```

In [62]:

```
my_tuple.index('b')  #get first index of element 'b' in tuple
```

Out[62]:

```
1
```

In [60]:

```
my_tuple.index('v')  #get first index of element 'v' in tuple, as value is not present thro
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-60-6cd5cd5489f8> in <module>
----> 1 my_tuple.index('v')  #get first index of element 'v' in tuple, as va
lue is not present throws error

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

count() - counts the number of instances of element value in tuple.

In [63]:

```
my_tuple.count('a') # count how many times 'a' appeared in tuple
```

Out[63]:

```
1
```

In [64]:

```
my_tuple.count('c') # count how many times 'c' appeared in tuple
```

Out[64]:

```
2
```

In [105]:

```
my_tuple.count('m') # count how many times 'm' appeared in tuple (which is not present as e
```

Out[105]:

```
0
```

Iterable

Tuple is iterable. One can traverse through the individual tuple elements using loops.

In [68]:

```
for i in range(len(my_tuple)):   #access tuple elements with for loop
    print("(", i, ") ", my_tuple[i])
```

```
( 0 )  a
( 1 )  b
( 2 )  c
( 3 )  c
( 4 )  e
```

Sorting

Tuple elements can be sorted as follows :

In [69]:

```
Scores = (10, 8 , 3, 2 , 4, 0 , 0, 3)
Scores
```

Out[69]:

```
(10, 8, 3, 2, 4, 0, 0, 3)
```

In [71]:

```
sorted(Scores)  # sort the tuple values using the sorted
```

Out[71]:

```
[0, 0, 2, 3, 3, 4, 8, 10]
```

In [73]:

```
grades = ('a', 'c', 'd', 'e', 'b')
grades
```

Out[73]:

```
('a', 'c', 'd', 'e', 'b')
```

In [74]:

```
sorted(grades)
```

Out[74]:

```
['a', 'b', 'c', 'd', 'e']
```

Deletion

Tuple can be removed from the memory by using del().

In [82]:

```
my_tuple     # tuple elements are printed
```

Out[82]:

```
('a', 'b', 'c', 'c', 'e')
```

In [83]:

```
del(my_tuple)  # remove the tuple
```

In [84]:

```
my_tuple  # throws error as tuple no more available
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-84-92689c492a76> in <module>
----> 1 my_tuple  # throws error as tuple no more available

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

Exercise

Q1. Accept 5 string values from user and store them as part of tuple. Print the elements of tuple with their indices.

In [85]:

```
#Try it here
```

In [92]:

```
#Try it here
```

Q2. Accept 5 string values from user and store them as part of tuple. Print the elements of tuple with their indices in reverse order.

In [ ]:

```
```