Tuples

Part-03

Rohini Mittal

- Tuple is an ordered sequence of element of different data types such as integer, float, string, list or even a tuple.
- Tuples are enclosed with paranthesis (round brackets) & separated by comma.
- Tuples are immutable.

```
# tuple with 0 element
    t1 = ()
# tuple with 1 element
    t2 = (74,)
```

Accessing Elements in Tuples

elements of a tuple can be accessed in the same way as a list or string using indexing and slicing.

Example

```
t1 = (10, 20, 30, 40)          O/P
print(t1[0])                   10
print(t1[-1])                  10
```

Tuple is Immutable

Tuples are immutable that means elements of a tuple cannot be changed after it has been created.

Example-1

```
t1 = (10, 20, 30)
t1[1] = 40
```

O/P

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

however an element of a tuple may be a mutable type eg. list.

Example 2

```
t2 = (10, 20, 30, [60, 70])
t2[3][1] = 10
print(t2)
```

o/p

```
(10, 20, 30, [60, 10])
```

Tuple operation

SNO	operation	operator	Example

1.	concatenation	+	t1 = (1, 3, 5, 7)
			t2 = (2, 4, 6, 8)
			print (t1 + t2)
			o/p (1, 3, 5, 7, 2, 4, 6, 8)

2.	Repetition	*	t1 = (1, 2, 3)
			print(t1 * 2)
			o/p (1, 2, 3, 1, 2, 3)

3.	Membership	in	t1 = (10, 20, 70, 80)
			print(20 in t1)
			print(90 in t1)
			o/p True
			False

		not in	t2 = (34, 67, 98) print(94 not in t2) print (34 not in t2) o/p True False
4.	slicing	:	Example t1 = (10, 20, 30, 40, 50, 60) print (t1 [2:]) o/p (30, 40, 50, 60)

Nested Tuples

A tuple inside another tuple is called nested tuple.

Ex t1 = (10, 20, (30, 40), (50, 60))

TUPLES METHODS AND BUILT-IN FUNCTIONS

Method	Description	Example

len()	**Returns the length or the number of elements of the tuple passed as the argument**	`>>> tuple1 = (10,20,30,40,50)`
		`>>> len(tuple1)`
		`5`

tuple()	**Creates an empty tuple if no argument is passed**
	
	Creates a tuple if a sequence is passed as argument
		`>>> tuple1 = tuple()`
		`>>> tuple1`
		`( )`
		`>>> tuple1 = tuple('aeiou')`
		`>>> tuple1`
		`('a', 'e', 'i', 'o', 'u')`
		`>>> tuple2=tuple([1,2,3])`
		`>>> tuple2`
		`(1, 2, 3)`
		`>>> tuple3 = tuple(range(5))`
		`>>> tuple3`
		`(0, 1, 2, 3, 4)`

count()	**Returns the number of times the given element appears in the tuple**	`>>> tuple1(10,20,30,10,40,10,50)`
		`>>> tuple1.count(10)`
		`3`
		`>>> tuple1.count(90)`
		`0`

min()
 	
max()
 	
sum()
	Returns minimum or smallest element of the tuple
	Returns maximum or largest element of the tuple.
	Returns sum of the elements of the tuple
		`>>> tuple1=(19,12,56,18,9,87,34)`
		`>>> min(tuple1)`
		`9`
		`>>> max(tuple1)`
		`87`
		`>>> sum(tuple1)`
		`235`

index()	**Returns the index of the first occurrence of the element in the given tuple**	`>>> tuple1 = (10,20,30,40,50)`
		`>>> tuple1.index(30)`
		`2`
		`>>> tuple1.index(90)`
		`ValueError: tuple.index(x): x not in tuple`

sorted()	**Takes elements in the tuple and returns a new sorted list. It should be noted that, sorted() does not make any change to the original tuple**	`>>> tuple1 = ("Rama","Heena","Raj","Mohsin","Aditya")`
		`>>> sorted(tuple1)`
		`('Aditya', 'Heena', 'Mohsin', 'Raj', 'Rama')`

Dictionaries

Dictionaries data type falls under mapping.
It is a mapping between set of keys & set of values.

syntax dict1 = {<key>: <value>}

Items in dictionaries are unordered, so we may not get back the data in the same order.

①Creating a dictionary

- To create a dictionary, the items entered are separated by comma and enclosed in curly braces.
- Each item is a key-value pair separated through colon (:)
- Keys of dictionary must be unique and should be of any immutable data type i.e. number, string or tuple.
- Values of dictionary can be repeated & can be of any data type.

```
# empty dictionary
   dict1 = { }
   print(dict1)
O/P  {}
```

```
# empty dictionary using built-in
  function
   dict2 = dict()
   print(dict2)
O/P  {}
```

```
# dictionary with keys & values
   dict3 = {'Mohan': 95 , 'Ram' : 90 , 'Shyam' : 92 }
   print(dict3)
O/P {'Mohan': 95 , 'Ram' : 90 , 'Shyam': 92}
```

② Accessing Items in a dictionary

Sequence (list, string, tuple) items are accessed by indexing but the items of dictionary are accessed via the keys. Each key serves as the index and maps to a value.

Example

```
dict1 = {'Mohan':95 , 'Ram':90 , 'Shyam':92}
print(dict1['Ram'])
print(dict1['Sangeeta']
```

O/P

```
90
KeyError: 'Sangeeta'
```

●

③ Adding a new item

We can add new item to the dictionary

Example -

```
dict1 ={'Mohan':95 , 'Ram':90}
dict1['Shyam'] = 92
print(dict1)
```

O/P

```
{'Mohan':95 , 'Ram':90 , 'Shyam':92}
```

●

④ Modifying an existing item

The existing dictionary can be modified by just overwriting the key-value pair.

Example -

```
dict1 = {'Mohan':95, 'Ram':90, 'Shyam':92}
dict1['Shyam'] = 94
print(dict1)
```

O/P

```
{'Mohan':95 , 'Ram':90 , 'Shyam':94}
```

Dictionary Operations

① Membership

in operator checks if the key is present in the dictionary and returns True else returns False.

Example

```
dict1 = {'Mohan':95 , 'Ram' :90 , 'Shyam':94}
print('Sangeeta' in dict1)
print('Mohan' in dict1)
```

O/P
```
False
True
```

not in operator checks if the key is not present in the dictionary then returns True, else return False

Example

```
dict1 = {'Mohan':95 , 'Ram':90 , 'Shyam':94}
print('Sangeeta' not in dict1)
print(' Mohan not in dict1)
```

O/P
```
True
False
```

Traversing a Dictionary

We can traverse each item of the dictionary by using for loop. dict1 = {'Mohan' : 95, 'Ram':90, 'Shyam': 94}

Method 1

```
for key in dict1:
        print( key, ":" , dict1[key])
```

O/P
```
Mohan : 95
Ram : 90
Shyam : 94
```

Method 2 .items() method

```
for item in dict1.items():
        print(item)
```

o/p

```
('Mohan' , 95)
('Ram', 90)
('Shyam', 94)
```

returns tuple of $(\overset{0}{\text{key}}, \overset{1}{\text{value}})$

```
for key, value in dict1.items():
        print( key , ":" , value)
```

o/p

```
Mohan : 95
Ram : 90
Shyam : 94
```

Method 3 .keys() method

```
for key in dict1.keys():
        print( key, ":" dict1[key])
```

o/p

```
Mohan : 95
Ram : 90
Shyam : 94
```

Method 4 .values() method

```
for value in dict1.values():
        print (value)
```

o/p

```
95
90
94
```

Dictionary methods and Built-in functions

Method	Description	Example

len()	**Returns the length or number of key: value pairs of the dictionary passed as the argument**	`>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92, 'Sangeeta':85}`
		`>>> len(dict1)`
		`4`

dict()	**Creates a dictionary from a sequence of key-value pairs**	`pair1 = [('Mohan',95),('Ram',89), ('Suhel',92),('Sangeeta',85)]`
		`>>> pair1`
		`[('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)]`
		`>>> dict1 = dict(pair1)`
		`>>> dict1`
		`{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}`

keys()	**Returns a list of keys in the dictionary**	`>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> dict1.keys()`
		`dict_keys(['Mohan', 'Ram', 'Suhel', 'Sangeeta'])`

values()	**Returns a list of values in the dictionary**	`>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> dict1.values()`
		`dict_values([95, 89, 92, 85])`

update()	**appends the key-value pair of the dictionary passed as the argument to the key-value pair of the given dictionary**	`>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> dict2 = {'Sohan':79,'Geeta':89}`
		`>>> dict1.update(dict2)`
		`>>> dict1`
		`{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Sohan': 79, 'Geeta': 89}`
		`>>> dict2`
		`{'Sohan': 79, 'Geeta': 89}`

del()	**Deletes the item with the given key**
	To delete the dictionary from the memory we write:
	`del Dict_name`
		`>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> del dict1['Ram']`
		`>>> dict1`
		`{'Mohan':95,'Suhel':92, 'Sangeeta': 85}`
		
		`>>> del dict1['Mohan']`
		`>>> dict1`
		`{'Suhel': 92, 'Sangeeta': 85}`
		`>>> del dict1`
		`>>> dict1`
		`NameError: name 'dict1' is not defined`

clear()	**Deletes or clear all the items of the dictionary**	`>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> dict1.clear()`
		`>>> dict1`
		`{ }`