TUPLE

Module 4

Module Outline:

- Tuple Overview
- Tuple Representation in Python
- Tuple Basic Operations
  - Accessing
  - Update
  - Deletion
- Special Tuple Operations
- Other Basic Tuple Operations

What is a Tuple?

Overview

- A tuple is a sequence of ***immutable*** Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the ***tuples cannot be changed*** unlike lists and tuples use parentheses, whereas lists use square brackets.

Tuple in Python

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tup1 = ("computer", "engineering", 2019, 2020)
tup2 = (1, 2, 3, 4, 5)
tup3 = "a", "b", "c", "d"
tup4 = ()

print (tup1)
print (tup2)
print (tup3)
print (tup4)
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Tuple Operations

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

```
tup1 = ("computer", "engineering", 2019, 2020)
tup2 = (1, 2, 3, 4, 5, 6, 7 )

print ("tup1[0]: ", tup1[0] )
print ("tup2[1:5]: ", tup2[1:5] )
```

What is the output?

Updating Tuples

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates –

Updating Tuples

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')
# Following action is not valid for tuples
# tup1[0] = 100


# So let's create a new tuple as follows

tup3 = tup1 + tup2
print (tup3)
```

What is the output?

Deleting List Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

Deleting List Elements

To explicitly remove an entire tuple, just use the **del** statement. For example -

```
tup = (computer', engineering', 2019, 2020)
print (tup)
del (tup)

print "After deleting tup : "
print (tup)
```

What is the output?

How tuples can be modified?

Append an Element in Tuple

Suppose we have ***tupleObj = (12, 34, 45, 22, 33)***

```
# Append 19 at the end of tuple
tupleObj = tupleObj + (19,)
print("Modified Tuple : ", tupleObj)
print("******* Insert an element at specific index in tuple *******")
print("Original Tuple : ", tupleObj)
```

What is the output?

Inserting an Element in Tuple

Suppose we have ***tupleObj = (12, 34, 45, 22, 33)***

```
# Insert 19 in tuple at index 2
tupleObj = tupleObj[:2] + (19,) + tupleObj[2:]
print("Modified Tuple : ", tupleObj)
print("******* Modify / Replace the element at specific index in tuple *******")
print("Original Tuple : ", tupleObj)
```

What is the output?

Replace / Modify an Element in Tuple

Suppose we have ***tupleObj = (12, 34, 45, 22, 33)***

```
# Replace the element at index 2 to 'Test'
tupleObj = tupleObj[:2] + ('test',) + tupleObj[3:]
print("Modified Tuple : ", tupleObj)
print("******* Delete the element at specific
index in tuple *******")
print("Original Tuple : ", tupleObj)
```

What is the output?

Replacing / Modifying an Element in Tuple

Suppose we have ***tupleObj = (12, 34, 45, 22, 33)***

```
tupleObj = tupleObj[:2] + ('test',) + tupleObj[3:]
print("Modified Tuple : ", tupleObj)
print("******* Delete the element at specific
index in tuple *******")
print("Original Tuple : ", tupleObj)
```

What is the output?

Deleting an Element in Tuple

Suppose we have ***tupleObj = (12, 34, 45, 22, 33)***

```
# Delete the element at index 2
tupleObj = tupleObj[:2] + tupleObj[3:]
print("Modified Tuple : ", tupleObj)
```

What is the output?

Other Basic Tuple Operations

- Tuples respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string.

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	[1, 2, 3, 4, 5, 6]	Concatenation
('Hi!')* 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Reference:

The instructor does not take the credits on the contents of this presentation.

https://www.tutorialspoint.com/python_data_structure/