Python Tuples

Tuples are ordered collection of data items. They store multiple items in a single variable. Tuple items are separated by commas and enclosed within round brackets (). Tuples are unchangeable meaning we can not alter them after creation.

Example 1:

```
tuple1 = (1,2,2,3,5,4,6)
tuple2 = ("Red", "Green", "Blue")
print(tuple1)
print(tuple2)
```

Output:

```
(1, 2, 2, 3, 5, 4, 6)
('Red', 'Green', 'Blue')
```

Example 2:

```
details = ("Abhijeet", 18, "FYBScIT", 9.8)
print(details)
```

Output:

```
('Abhijeet', 18, 'FYBScIT', 9.8)
```

Tuple Indexes

Each item/element in a tuple has its own unique index. This index can be used to access any particular item from the tuple. The first item has index [0], second item has index [1], third item has index [2] and so on.

Example:

```
country = ("Spain", "Italy", "India",)
#           [0]      [1]      [2]
```

Accessing tuple items:

I. Positive Indexing:

As we have seen that tuple items have index, as such we can access items using these indexes.

Example:

```
country = ("Spain", "Italy", "India",)
#          [0]      [1]      [2]
print(country[0])
print(country[1])
print(country[2])
```

Output:

```
Spain
Italy
India
```

II. Negative Indexing:

Similar to positive indexing, negative indexing is also used to access items, but from the end of the tuple. The last item has index [-1], second last item has index [-2], third last item has index [-3] and so on.

Example:

```
country = ("Spain", "Italy", "India", "England", "Germany")
#          [0]      [1]      [2]      [3]        [4]
print(country[-1]) # Similar to print(country[len(country) - 1])
print(country[-3])
print(country[-4])
```

Output:

```
Germany
India
Italy
```

III. Check for item:

We can check if a given item is present in the tuple. This is done using the `in` keyword.

Example 1:

```
country = ("Spain", "Italy", "India", "England", "Germany")
if "Germany" in country:
    print("Germany is present.")
else:
    print("Germany is absent.")
```

Output:

```
Germany is present.
```

Example 2:

```
country = ("Spain", "Italy", "India", "England", "Germany")
if "Russia" in country:
   print("Russia is present.")
else:
   print("Russia is absent.")
```

Output:

```
Russia is absent.
```

IV. Range of Index:

You can print a range of tuple items by specifying where do you want to start, where do you want to end and if you want to skip elements in between the range.

Syntax:

```
Tuple[start : end : jumpIndex]
```
Note: jump Index is optional. We will see this in given examples.

Example: Printing elements within a particular range:

```
animals = ("cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow")
print(animals[3:7])     #using positive indexes
print(animals[-7:-2])   #using negative indexes
```

Output:

```
('mouse', 'pig', 'horse', 'donkey')
('bat', 'mouse', 'pig', 'horse', 'donkey')
```
Here, we provide index of the element from where we want to start and the index of the element till which we want to print the values. Note: The element of the end index provided will not be included.

Example: Printing all element from a given index till the end

```
animals = ("cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow")
print(animals[4:])      #using positive indexes
print(animals[-4:])     #using negative indexes
```

Output:

```
('pig', 'horse', 'donkey', 'goat', 'cow')
('horse', 'donkey', 'goat', 'cow')
```
When no end index is provided, the interpreter prints all the values till the end.

Example: printing all elements from start to a given index

```
animals = ("cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow")
```

```
print(animals[:6])      #using positive indexes
print(animals[:-3])     #using negative indexes
```

Output:

```
('cat', 'dog', 'bat', 'mouse', 'pig', 'horse')
('cat', 'dog', 'bat', 'mouse', 'pig', 'horse')
```

When no start index is provided, the interpreter prints all the values from start up to the end index provided.

Example: Print alternate values

```
animals = ("cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow")
print(animals[::2])     #using positive indexes
print(animals[-8:-1:2]) #using negative indexes
```

Output:

```
('cat', 'bat', 'pig', 'donkey', 'cow')
('dog', 'mouse', 'horse', 'goat')
```

Here, we have not provided start and end index, which means all the values will be considered. But as we have provided a jump index of 2 only alternate values will be printed.

Example: printing every 3rd consecutive withing given range

```
animals = ("cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow")
print(animals[1:8:3])
```

Output:

```
('dog', 'pig', 'goat')
```

Here, jump index is 3. Hence it prints every 3rd element within given index.

Manipulating Tuples

Tuples are immutable, hence if you want to add, remove or change tuple items, then first you must convert the tuple to a list. Then perform operation on that list and convert it back to tuple.

Example:

```
countries = ("Spain", "Italy", "India", "England", "Germany")
temp = list(countries)
temp.append("Russia")      #add item
temp.pop(3)                #remove item
temp[2] = "Finland"        #change item
countries = tuple(temp)
print(countries)
```

Output:

```
('Spain', 'Italy', 'Finland', 'Germany', 'Russia')
```

Thus, we convert the tuple to a list, manipulate items of the list using list methods, then convert list back to a tuple.

However, we can directly concatenate two tuples without converting them to list.

Example:

```
countries = ("Pakistan", "Afghanistan", "Bangladesh", "ShriLanka")
countries2 = ("Vietnam", "India", "China")
southEastAsia = countries + countries2
print(southEastAsia)
```

Output:

```
('Pakistan', 'Afghanistan', 'Bangladesh', 'ShriLanka', 'Vietnam', 'India', 'China')
```

Tuple methods

As tuple is immutable type of collection of elements it have limited built in methods.They are explained below

count() Method

The count() method of Tuple returns the number of times the given element appears in the tuple.

Syntax:

```
tuple.count(element)
```

Example

```
Tuple1 = (0, 1, 2, 3, 2, 3, 1, 3, 2)
res = Tuple1.count(3)
print('Count of 3 in Tuple1 is:', res)
```

Output

```
3
```

index() method

The Index() method returns the first occurrence of the given element from the tuple.

Syntax:

```
tuple.index(element, start, end)
```

Note: This method raises a ValueError if the element is not found in the tuple.

Example

```
Tuple = (0, 1, 2, 3, 2, 3, 1, 3, 2)
res = Tuple.index(3)
print('First occurrence of 3 is', res)
```

Output

```
3
```