Tuples

Python Lab
12-02-2014

By Soofi Shafiya

Python Tuples

```
mytuple = ("apple", "banana", "cherry")
```

- Tuples are used to store multiple items in a single variable.
- Tuple items are **ordered, unchangeable, and allow duplicate values.**
- When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
- Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.
- Since tuples are indexed, they can have items with the same value.

```
Tuple length:
print(len(thistuple))
```

- 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.

```
thistuple = ("apple",)
```

- Tuple items can be of any data type.
- A tuple can contain different data types

```
print(type(mytuple))
```

The tuple() Constructor

```
thistuple = tuple(("apple", "banana", "cherry")) #
note the double round-brackets
print(thistuple)
```

Access Tuple items

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

```
print(thistuple[1])
```

Negative indexing

```
print(thistuple[-1])
```

Range of Indexes

You can 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.

- ```
  thistuple =
  ("apple", "banana", "cherry", "orange", "kiwi", "me
  lon", "mango")
  print(thistuple[2:5])
  ```
  ```

```
print(thistuple[:4])
O/P ????
print(thistuple[2:])
O/P ????
print(thistuple[-4:-1])
O/P ????
```

Check if Item Exists

- To determine if a specified item is present in a tuple use the ?? keyword.

WAP to check if "apple" is present in the tuple.

```
thistuple = ("apple", "banana", "cherry")

if ‘apple’ ?? Thistuple:
     print(“yes”)
```

Change Tuple Values

- Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.
- But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

Convert the tuple into a list to be able to change it:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

Add Items

Since tuples are immutable, they do not have an inbuilt append() method, but there are other ways to add items to a tuple.

1. **Convert into a list**: Just like the workaround for *changing* a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

Example:

Convert the tuple into a list, add "orange", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

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

Example

Create a new tuple with the value "orange", and add that tuple:

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y

print(thistuple)
```

Remove Items

Note: You cannot remove items from a tuple.

Tuples are **unchangeable**, so you cannot remove items from it, but you can use the same workaround as we used for changing and adding tuple items.

Example

Convert the tuple into a list, remove "apple", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```

Try it Yourself

- Alternatively, we can delete the tuple completely using del keyword.

Try it yourself

Unpack Tuples

Extracting the values back into variables.

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)
print(yellow)
print(red)
```

Note: 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 *

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")

(green, yellow, *red) = fruits
```

- `(green, *tropic, red) = fruits`

Output ???

Loop Tuples

For loop:

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

```
for i in range(len(thistuple)):
  print(thistuple[i]) # loop through index number
```

Exercise

Use WHILE loop!!

Join Tuples

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

Multiply Tuples

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2
```

EXERCISE

Tuples

Fruits = (“cherry”, “banana”, “apple”, ”orange”)

1. Print the first item of tuple
2. Use the correct syntax to print the number of items in the tuple
3. Use negative indexing to print the last item in the tuple.
4. Use a range of indexes to print the third, fourth, and fifth item in the tuple.