Python Tuples

Tuple

- A tuple is a collection which is ordered and immutable(**unchangeable)**. In Python tuples are written with round brackets.
- **Example**

Create a Tuple:

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

Access Tuple Items

- You can access tuple items by referring to the index number, inside square brackets:
- **Example**
- Print the second item in the tuple:
- thistuple = ("apple", "banana", "cherry")
  print(thistuple[1])

Negative Indexing

- Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second last item etc.
- **Example**
- Print the last item of the tuple:
- thistuple = ("apple", "banana", "cherry")
  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.
- **Example**
- Return the third, fourth, and fifth item:

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

- **Note:** The search will start at index 2 (included) and end at index 5 (not included).
- Remember that the first item has index 0.

Range of Negative Indexes

- Specify negative indexes if you want to start the search from the end of the tuple:
- **Example**
- This example returns the items from index -4 (included) to index -1 (excluded)

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])
```

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.
- **Example**
- 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)
```

Loop Through a Tuple

- You can loop through the tuple items by using a for loop.
- **Example**
- Iterate through the items and print the values:

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

Check if Item Exists

- To determine if a specified item is present in a tuple use the in keyword:
- **Example**
- Check if "apple" is present in the tuple:

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
    print("Yes, 'apple' is in the fruits tuple")
```

Tuple Length

- To determine how many items a tuple has, use the len() method:
- **Example**
- Print the number of items in the tuple:

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

Add Items

- Once a tuple is created, you cannot add items to it. Tuples are **unchangeable**.
- **Example**
- You cannot add items to a tuple:

```
thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange" # This will raise an error
print(thistuple)
```

Create Tuple With One Item

To create a tuple with only one item, you have add a comma after the item, unless Python will not recognize the variable as a tuple.

Example

One item tuple, remember the commma:

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

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Remove Items

- **Note:** You cannot remove items in a tuple.
- Tuples are **unchangeable**, so you cannot remove items from it, but you can delete the tuple completely:
- **Example**
- The del keyword can delete the tuple completely:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

Join Two Tuples

- To join two or more tuples you can use the + operator:
- **Example**
- Join two tuples:

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

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

The tuple() Constructor

- To create a tuple with only one item, you have add a comma after the item, unless Python will not recognize the variable as a tuple.
- **Example**
- One item tuple, remember the commma:

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

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Tuple Methods

- Python has two built-in methods that you can use on tuples.

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found