Tuple

- Tuples are used to store multiple items in a single variable.
- A tuple is a collection which is ordered and unchangeable.
- Tuples are written with round brackets.
- A tuple can contain different data types
- Tuple items are ordered, unchangeable, and allow duplicate values.
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Creating a Tuple

Create a Tuple

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

```
('apple', 'banana', 'cherry')
```

Using the tuple() method to make a tuple

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

```
('apple', 'banana', 'cherry')
```

Tuple Length

To determine how many items a tuple has, use the len() function.

Print the number of items in the tuple

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

```
3
```

Create Tuple With One Item

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.

One item tuple, remember the comma

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

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

```
<class 'tuple'>
<class 'str'>
```

Access Tuple Items

You can access tuple items by referring to the index number, inside square brackets.

Print the second item in the tuple

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

```
banana
```

Note: The first item has index 0.

Negative Indexing

Negative indexing means start from the end.

-1 refers to the last item, -2 refers to the second last item etc.

Print the last item of the tuple

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

```
cherry
```

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.

Return the third, fourth, and fifth item

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

```
('cherry', 'orange', 'kiwi')
```

Note: The search will start at index 2 (included) and end at index 5 (not included).

By leaving out the start value, the range will start at the first item

```
# This example returns the items from the beginning to, but NOT included, "kiwi":

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

```
('apple', 'banana', 'cherry', 'orange')
```

By leaving out the end value, the range will go on to the end of the list

```
# This example returns the items from "cherry" and to the end:

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

```
('cherry', 'orange', 'kiwi', 'melon', 'mango')
```

Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the tuple.

```
# 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])
```

```
('orange', 'kiwi', 'melon')
```

Check if Item Exists

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

```
# Check if "apple" is present in the tuple:

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

```
Yes, 'apple' is in the fruits tuple
```

Update Tuples

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)
```

```
('apple', 'kiwi', 'cherry')
```

Add Items

Since tuples are immutable, they do not have a build-in 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.

```
  # 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)
  print(thistuple)
```

```
('apple', 'banana', 'cherry', 'orange')
```

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.

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

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

print(thistuple)
```

```
('apple', 'banana', 'cherry', 'orange')
```

Remove Items

Note: You cannot remove items in 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.

```
# Convert the tuple into a list, remove "apple", and convert it back into a tup

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

```
('banana', 'cherry')
```

Or you can delete the tuple completely

```
# 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
```

Unpacking a Tuple

When we create a tuple, we normally assign values to it. This is called "packing" a tuple.

```
# Packing a tuple:

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

In Python, we are also allowed to extract the values back into variables. This is called "unpacking".

```
# Unpacking a tuple:

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

(green, yellow, red) = fruits

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

```
apple
banana
cherry
```

Using Asterisk

If the number of variables is less than the number of values, you can add an * to the variable name and the values will be assigned to the variable as a list.

```
# Assign the rest of the values as a list called "red":

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

(green, yellow, *red) = fruits

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

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

If the asterisk is added to another variable name than the last, Python will assign values to the variable until the number of values left matches the number of variables left.

```
# Add a list of values the "tropic" variable:

fruits = ("apple", "mango", "papaya", "pineapple", "cherry")

(green, *tropic, red) = fruits

print(green)
print(tropic)
print(red)
```

```
apple
['mango', 'papaya', 'pineapple']
cherry
```

Loop Through a Tuple

You can loop through the tuple items by using a for loop.

```
# Iterate through the items and print the values:

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

```
apple
banana
cherry
```

Loop Through the Index Numbers

You can also loop through the tuple items by referring to their index number.

Use the range() and len() functions to create a suitable iterable.

```
  # Print all items by referring to their index number:

  thistuple = ("apple", "banana", "cherry")
  for i in range(len(thistuple)):
    print(thistuple[i])
```

```
apple
banana
cherry
```

Using a While Loop

You can loop through the list items by using a while loop.

Use the len() function to determine the length of the tuple, then start at 0 and loop your way through the tuple items by refering to their indexes.

Remember to increase the index by 1 after each iteration.

```
# Print all items, using a while loop to go through all the index numbers:

thistuple = ("apple", "banana", "cherry")
i = 0
while i < len(thistuple):
  print(thistuple[i])
  i = i + 1
```

```
apple
banana
cherry
```

Join Tuples

Join Two Tuples

To join two or more tuples you can use the + operator.

```
# Join two tuples:

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

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

```
('a', 'b', 'c', 1, 2, 3)
```

Multiply Tuples

If you want to multiply the content of a tuple a given number of times, you can use the * operator.

```
# Multiply the fruits tuple by 2:

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

print(mytuple)
```

```
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
```

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