Tuple

A tuple is a collection which is ordered and **unchangeable**. In Python tuples are written with round brackets.

### **Example**

Create a Tuple:

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

A tuple can be written as the collection of comma-separated values enclosed with the small brackets. A tuple can be defined as follows.

T1 = (101, "Ayush", 22)

T2 = ("Apple", "Banana", "Orange")

## 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])

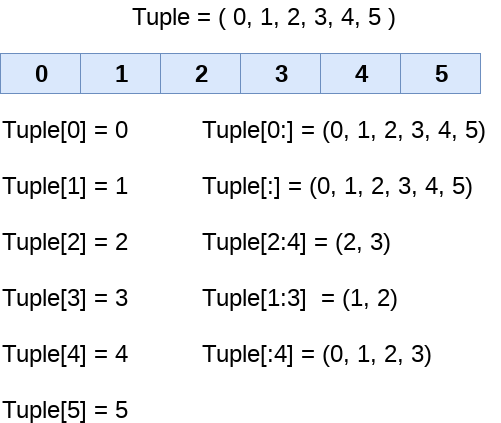
### **Range of Negative Indexes**

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)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_tuple_concat)

Tuple Methods

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| [count()](https://www.w3schools.com/python/ref_tuple_count.asp) | Returns the number of times a specified value occurs in a tuple |
| [index()](https://www.w3schools.com/python/ref_tuple_index.asp) | Searches the tuple for a specified value and returns the position of where it was found |

Definition and Usage

The count() method returns the number of times a specified value appears in the tuple.

Syntax

*tuple*.count(*value*)

### **Example**

Return the number of times the value 5 appears in the tuple:

thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)  
  
x = thistuple.count(5)  
  
print(x)

Definition and Usage

The index() method finds the first occurrence of the specified value.

The index() method raises an exception if the value is not found.

Syntax

*tuple*.index(*value*)

Parameter Values

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *value* | Required. The item to search for |

tuple1 = (10, 20, 30, 40, 50, 60)

**print**(tuple1)

count = 0

**for** i **in** tuple1:

**print**("tuple1[%d] = %d"%(count, i));

[**next →**](https://www.javatpoint.com/python-set)[**← prev**](https://www.javatpoint.com/python-lists)

# **Python Tuple**

Python Tuple is used to store the sequence of immutable python objects. Tuple is similar to lists since the value of the items stored in the list can be changed whereas the tuple is immutable and the value of the items stored in the tuple can not be changed.

A tuple can be written as the collection of comma-separated values enclosed with the small brackets. A tuple can be defined as follows.

1. T1 = (101, "Ayush", 22)
2. T2 = ("Apple", "Banana", "Orange")

## Example

tuple1 = (10, 20, 30, 40, 50, 60)

**print**(tuple1)

count = 0

**for** i **in** tuple1:

**print**("tuple1[%d] = %d"%(count, i));

**Output:**

(10, 20, 30, 40, 50, 60)

tuple1[0] = 10

tuple1[0] = 20

tuple1[0] = 30

tuple1[0] = 40

tuple1[0] = 50

tuple1[0] = 60

Example 2

tuple1 = tuple(input("Enter the tuple elements ..."))

**print**(tuple1)

count = 0

**for** i **in** tuple1:

**print**("tuple1[%d] = %s"%(count, i));

**Output:**

Enter the tuple elements ...12345

('1', '2', '3', '4', '5')

tuple1[0] = 1

tuple1[0] = 2

tuple1[0] = 3

tuple1[0] = 4

tuple1[0] = 5

## Basic Tuple operations

The operators like concatenation (+), repetition (\*), Membership (in) works in the same way as they work with the list. Consider the following table for more detail.

Let's say Tuple t = (1, 2, 3, 4, 5) and Tuple t1 = (6, 7, 8, 9) are declared.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| Repetition | The repetition operator enables the tuple elements to be repeated multiple times. | T1\*2 = (1, 2, 3, 4, 5, 1, 2, 3, 4, 5) |
| Concatenation | It concatenates the tuple mentioned on either side of the operator. | T1+T2 = (1, 2, 3, 4, 5, 6, 7, 8, 9) |
| Membership | It returns true if a particular item exists in the tuple otherwise false. | print (2 in T1) prints True. |
| Iteration | The for loop is used to iterate over the tuple elements. | for i in T1:  print(i)  **Output**  1  2  3  4  5 |
| Length | It is used to get the length of the tuple. | len(T1) = 5 |

1. Which of the following is a Python tuple?  
   a) [1, 2, 3]   
   b) (1, 2, 3)  
   c) {1, 2, 3}  
   d) {}
2. Suppose t = (1, 2, 4, 3), which of the following is incorrect?  
   a) print(t[3])  
   b) t[3] = 45  
   c) print(max(t))  
   d) print(len(t))
3. What will be the output of the following Python code?

t=(1,2,4,3)

t[1:3]

1. What will be the output of the following Python code?

t=(1,2,4,3)

t[1:-1]

1. What is the data type of (1)?  
   a) Tuple  
   b) Integer  
   c) List  
   d) Both tuple and integer

## Set

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

### **Example**

Create a Set:

thisset = {"apple", "banana", "cherry"}  
print(thisset)

**Note:** Sets are unordered, so you cannot be sure in which order the items will appear.

## Access Items

You cannot access items in a set by referring to an index, since sets are unordered the items has no index.

But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

### **Example**

Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}  
  
for x in thisset:  
  print(x)

### **Example**

Check if "banana" is present in the set:

thisset = {"apple", "banana", "cherry"}  
  
print("banana" in thisset)

## Change Items

Once a set is created, you cannot change its items, but you can add new items.

## Add Items

To add one item to a set use the add() method.

To add more than one item to a set use the update() method.

### **Example**

Add an item to a set, using the add() method:

thisset = {"apple", "banana", "cherry"}  
  
thisset.add("orange")  
  
print(thisset)

### **Example**

Add multiple items to a set, using the update() method:

thisset = {"apple", "banana", "cherry"}  
  
thisset.update(["orange", "mango", "grapes"])  
  
print(thisset)

## Get the Length of a Set

To determine how many items a set has, use the len() method.

### **Example**

Get the number of items in a set:

thisset = {"apple", "banana", "cherry"}  
  
print(len(thisset))

## Remove Item

To remove an item in a set, use the remove(), or the discard() method.

### **Example**

Remove "banana" by using the remove() method:

thisset = {"apple", "banana", "cherry"}  
  
thisset.remove("banana")  
  
print(thisset)

**Note:** If the item to remove does not exist, remove() will raise an error.

### **Example**

Remove "banana" by using the discard() method:

thisset = {"apple", "banana", "cherry"}  
  
thisset.discard("banana")  
  
print(thisset)

**Note:** If the item to remove does not exist, discard() will **NOT** raise an error.

You can also use the pop(), method to remove an item, but this method will remove the last item. Remember that sets are unordered, so you will not know what item that gets removed.

The return value of the pop() method is the removed item.

### **Example**

Remove the last item by using the pop() method:

thisset = {"apple", "banana", "cherry"}  
  
x = thisset.pop()  
  
print(x)  
  
print(thisset)

**Note:** Sets are unordered, so when using the pop() method, you will not know which item that gets removed.

### **Example**

The clear() method empties the set:

thisset = {"apple", "banana", "cherry"}  
  
thisset.clear()  
  
print(thisset)

### **Example**

The del keyword will delete the set completely:

thisset = {"apple", "banana", "cherry"}  
  
del thisset  
  
print(thisset)

## Join Two Sets

There are several ways to join two or more sets in Python.

You can use the union() method that returns a new set containing all items from both sets, or the update()method that inserts all the items from one set into another:

### **Example**

The union() method returns a new set with all items from both sets:

set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
  
set3 = set1.union(set2)  
print(set3)

### **Example**

The update() method inserts the items in set2 into set1:

set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
  
set1.update(set2)  
print(set1)

**Note:** Both union() and update() will exclude any duplicate items.

Set Methods

Python has a set of built-in methods that you can use on sets.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | Adds an element to the set |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) | Remove the specified item |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two other sets |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Removes an element from the set |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specified element |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | inserts the symmetric differences from this set and another |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with the union of this set and others |

https://www.w3schools.com/python/python\_sets.asp