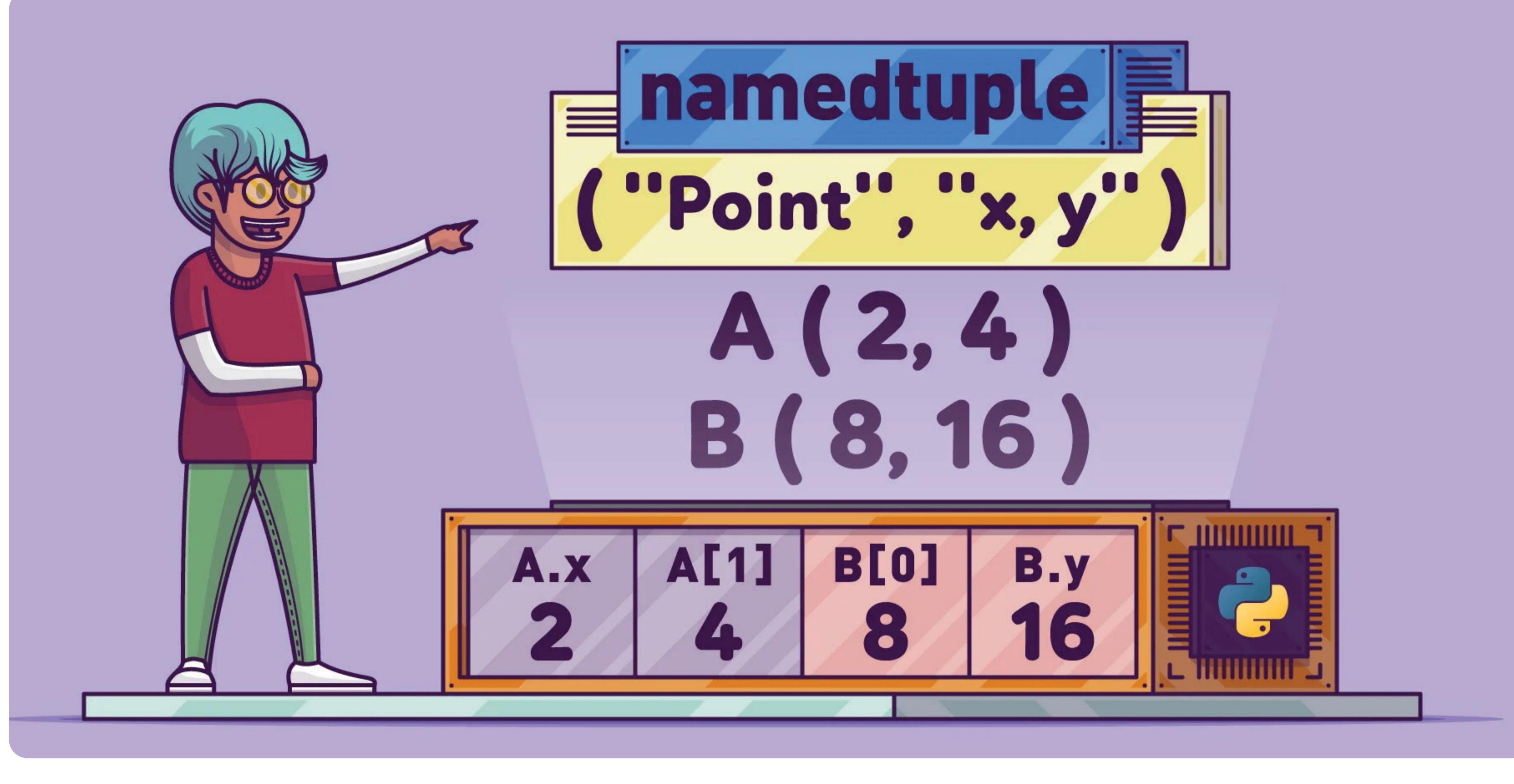


Tuples



• • •

```
t1 = (22,45,67,89)
t2 = (1,3)
t3 = t1+t2
t3*2
```

(22, 45, 67, 89, 1, 3, 22, 45, 67, 89, 1, 3)

Tuples are immutable lists and cannot be changed in any way once it is created.

- Tuples are defined in the same way as lists.
- They are enclosed within parenthesis and not within square braces.
- Tuples are ordered, indexed collections of data.
- Similar to string indices, the first value in the tuple will have the index [0], the second value [1]
- Negative indices are counted from the end of the tuple, just like lists.
- Tuple also has the same structure where commas separate the values.
- Tuples can store duplicate values.
- Tuples allow you to store several data items including string, integer, float in one variable.

• • •

```
# Take a tuple
tuple_1 = ('Hello', 'Python', 3.14, 1.618, True, False, 32,
[1,2,3], {1,2,3}, {'A': 3, 'B': 8}, (0, 1))
tuple_1
```

('Hello',
'Python',
3.14,
1.618,
True,
False,
32,
[1, 2, 3],
{1, 2, 3},
{'A': 3, 'B': 8},
(0, 1))

• • •

```
print(type(tuple_1))
print(len(tuple_1))
```

<class 'tuple'>

Indexing

● ● ●

```
# Printing the each value in a tuple using both positive and negative indexing
tuple_1 = ('Hello', 'Python', 3.14, 1.618, True, False, 32,
[1,2,3], {1,2,3}, {'A': 3, 'B': 8}, (0, 1))
print(tuple_1[0])
print(tuple_1[1])
print(tuple_1[2])
print(tuple_1[-1])
print(tuple_1[-2])
print(tuple_1[-3])
```

Hello

Python

3.14

(0, 1)

{'A': 3, 'B': 8}

{1, 2, 3}

● ● ●

```
# Printing the type of each value in the tuple
tuple_1 = ('Hello', 'Python', 3.14, 1.618, True, False, 32,
[1,2,3], {1,2,3}, {'A': 3, 'B': 8}, (0, 1))
print(type(tuple_1[0]))
print(type(tuple_1[2]))
print(type(tuple_1[4]))
print(type(tuple_1[6]))
print(type(tuple_1[7]))
print(type(tuple_1[8]))
print(type(tuple_1[9]))
print(type(tuple_1[10]))
```

```
<class 'str'>
<class 'float'>
<class 'bool'>
<class 'int'>
<class 'list'>
<class 'set'>
<class 'dict'>
<class 'tuple'>
```

Concatenation of tuples

To concatenate tuples, + sign is used

● ● ●

```
tuple_2 = tuple_1 + ('Hello World!', 2022)
tuple_2
```

```
('Hello',
'Python',
3.14,
1.618,
True,
False,
32,
[1, 2, 3],
{1, 2, 3},
{'A': 3, 'B': 8},
(0, 1),
'Hello World!',
2022)
```

Repetition of a tuple

● ● ●

```
rep_tup = (1,2,3,4)
rep_tup*2
```

(1, 2, 3, 4, 1, 2, 3, 4)

Membership

```
● ● ●  
rep_tup = (1,2,3,4)  
print(2 in rep_tup)  
print(2 not in rep_tup)  
print(5 in rep_tup)  
print(5 not in rep_tup)
```

True
False
False
True

Iteration

```
● ● ●  
rep_tup = (1,2,3,4)  
for i in rep_tup:  
    print(i)
```

1
2
3
4

Slicing

To obtain a new tuple from the current tuple, the slicing method is used.

```
● ● ●  
# Obtaining a new tuple from the index 2 to index 6  
tuple_1 = ('Hello', 'Python', 3.14, 1.618, True, False, 32,  
[1,2,3], {1,2,3}, {'A': 3, 'B': 8}, (0, 1))  
tuple_1[2:7]
```

(3.14, 1.618, True, False, 32)

```
● ● ●  
# Obtaining tuple using negative indexing  
tuple_1 = ('Hello', 'Python', 3.14, 1.618, True, False, 32,  
[1,2,3], {1,2,3}, {'A': 3, 'B': 8}, (0, 1))  
tuple_1[-4:-1]
```

([1, 2, 3], {1, 2, 3}, {'A': 3, 'B': 8})

Different Tuple operation

min() function

```
● ● ●  
rep_tup = (1,2,3,4)  
min(rep_tup)
```

1

min() function

```
● ● ●  
rep_tup = (1,2,3,4)  
max(rep_tup)
```

4

tup(seq) function

It converts a specific sequence to a tuple

```
● ● ●  
seq = 'ATGCGTATTGCCAT'  
tuple(seq)
```

('A', 'T', 'G', 'C', 'G', 'T', 'A', 'T', 'T', 'G', 'C', 'C', 'A', 'T')

len() function

To obtain how many elements there are in the tuple, use len() function.

```
● ● ●  
tuple_1 = ('Hello', 'Python', 3.14, 1.618, True, False, 32,  
[1,2,3], {1,2,3}, {'A': 3, 'B': 8}, (0, 1))  
len(tuple_1)
```

11

Sorting tuple

```
● ● ●  
# Tuples can be sorted and save as a new tuple.  
tuple_3 = (0,9,7,4,6,2,9,8,3,1)  
sorted_tuple_3 = sorted(tuple_3)  
sorted_tuple_3
```

[0, 1, 2, 3, 4, 6, 7, 8, 9, 9]

Nested tuple

In Python, a tuple written inside another tuple is known as a nested tuple.

```
● ● ●  
# Take a nested tuple  
nested_tuple =('biotechnology', (0, 5), ('fermentation', 'ethanol'), (3.14, 'pi', (1.618, 'golden ratio'))) )  
nested_tuple  
  
('biotechnology',  
(0, 5),  
(('fermentation', 'ethanol'),  
(3.14, 'pi', (1.618, 'golden ratio'))))
```

```
● ● ●  
# Now printing the each element of the nested tuple  
print('Item 0 of nested tuple is', nested_tuple[0])  
print('Item 1 of nested tuple is', nested_tuple[1])  
print('Item 2 of nested tuple is', nested_tuple[2])  
print('Item 3 of nested tuple is', nested_tuple[3])
```

Item 0 of nested tuple is biotechnology
Item 1 of nested tuple is (0, 5)
Item 2 of nested tuple is ('fermentation', 'ethanol')
Item 3 of nested tuple is (3.14, 'pi', (1.618, 'golden ratio'))

```
● ● ●  
# Using second index to access other tuples in the nested tuple  
print('Item 1, 0 of the nested tuple is', nested_tuple[1][0])  
print('Item 1, 1 of the nested tuple is', nested_tuple[1][1])  
print('Item 2, 0 of the nested tuple is', nested_tuple[2][0])  
print('Item 2, 1 of the nested tuple is', nested_tuple[2][1])  
print('Item 3, 0 of the nested tuple is', nested_tuple[3][0])  
print('Item 3, 1 of the nested tuple is', nested_tuple[3][1])  
print('Item 3, 2 of the nested tuple is', nested_tuple[3][2])  
# Accessing to the items in the second nested tuples using a third index  
print('Item 3, 2, 0 of the nested tuple is', nested_tuple[3][2][0])  
print('Item 3, 2, 1 of the nested tuple is', nested_tuple[3][2][1])
```

Item 1, 0 of the nested tuple is 0
Item 1, 1 of the nested tuple is 5
Item 2, 0 of the nested tuple is fermentation
Item 2, 1 of the nested tuple is ethanol
Item 3, 0 of the nested tuple is 3.14
Item 3, 1 of the nested tuple is pi
Item 3, 2 of the nested tuple is (1.618, 'golden ratio')
Item 3, 2, 0 of the nested tuple is 1.618
Item 3, 2, 1 of the nested tuple is golden ratio

Tuples are immutable

```
● ● ●  
# Take a tuple  
tuple_4 = (1,3,5,7,8)  
tuple_4[0] = 9  
print(tuple_4)  
# The output shows the tuple is immutable
```

11

TypeError Input In [21], in <cell line: 3>()

```
1 # Take a tuple  
2 tuple_4 = (1,3,5,7,8)  
3 tuple_4[0] = 9  
4 print(tuple_4)
```

Traceback (most recent call last)

---->

```
5 tuple_4[0] = 9  
6 print(tuple_4)
```

```
7 print(tuple_4)
```

```
8 print(tuple_4)
```

```
9 print(tuple_4)
```

```
10 print(tuple_4)
```

```
11 print(tuple_4)
```

```
12 print(tuple_4)
```

```
13 print(tuple_4)
```

```
14 print(tuple_4)
```

```
15 print(tuple_4)
```

```
16 print(tuple_4)
```

```
17 print(tuple_4)
```

```
18 print(tuple_4)
```

```
19 print(tuple_4)
```

```
20 print(tuple_4)
```

```
21 print(tuple_4)
```

```
22 print(tuple_4)
```

```
23 print(tuple_4)
```

```
24 print(tuple_4)
```

```
25 print(tuple_4)
```

```
26 print(tuple_4)
```

```
27 print(tuple_4)
```

```
28 print(tuple_4)
```

```
29 print(tuple_4)
```

```
30 print(tuple_4)
```

```
31 print(tuple_4)
```

```
32 print(tuple_4)
```

```
33 print(tuple_4)
```

```
34 print(tuple_4)
```

```
35 print(tuple_4)
```

```
36 print(tuple_4)
```

```
37 print(tuple_4)
```

```
38 print(tuple_4)
```

```
39 print(tuple_4)
```

```
40 print(tuple_4)
```

```
41 print(tuple_4)
```

```
42 print(tuple_4)
```

```
43 print(tuple_4)
```

```
44 print(tuple_4)
```

```
45 print(tuple_4)
```

```
46 print(tuple_4)
```

```
47 print(tuple_4)
```

```
48 print(tuple_4)
```

```
49 print(tuple_4)
```

```
50 print(tuple_4)
```

```
51 print(tuple_4)
```

```
52 print(tuple_4)
```

```
53 print(tuple_4)
```

```
54 print(tuple_4)
```

```
55 print(tuple_4)
```

```
56 print(tuple_4)
```

```
57 print(tuple_4)
```

```
58 print(tuple_4)
```

```
59 print(tuple_4)
```

```
60 print(tuple_4)
```

```
61 print(tuple_4)
```

```
62 print(tuple_4)
```

```
63 print(tuple_4)
```

```
64 print(tuple_4)
```

```
65 print(tuple_4)
```

```
66 print(tuple_4)
```

```
67 print(tuple_4)
```

```
68 print(tuple_4)
```

```
69 print(tuple_4)
```

```
70 print(tuple_4)
```

```
71 print(tuple_4)
```

```
72 print(tuple_4)
```

```
73 print(tuple_4)
```

```
74 print(tuple_4)
```

```
75 print(tuple_4)
```

```
76 print(tuple_4)
```

```
77 print(tuple_4)
```

```
78 print(tuple_4)
```

```
79 print(tuple_4)
```

```
80 print(tuple_4)
```

```
81 print(tuple_4)
```

```
82 print(tuple_4)
```

```
83 print(tuple_4)
```

```
84 print(tuple_4)
```

```
85 print(tuple_4)
```

```
86 print(tuple_4)
```

```
87 print(tuple_4)
```

```
88 print(tuple_4)
```

```
89 print(tuple_4)
```

```
90 print(tuple_4)
```

```
91 print(tuple_4)
```

```
92 print(tuple_4)
```

```
93 print(tuple_4)
```

```
94 print(tuple_4)
```

```
95 print(tuple_4)
```

```
96 print(tuple_4)
```

```
97 print(tuple_4)
```

```
98 print(tuple_4)
```

```
99 print(tuple_4)
```

```
100 print(tuple_4)
```

```
101 print(tuple_4)
```

```
102 print(tuple_4)
```

```
103 print(tuple_4)
```

```
104 print(tuple_4)
```

```
105 print(tuple_4)
```

```
106 print(tuple_4)
```

```
107 print(tuple_4)
```

```
108 print(tuple_4)
```

```
109 print(tuple_4)
```

```
110 print(tuple_4)
```

```
111 print(tuple_4)
```

```
112 print(tuple_4)
```

```
113 print(tuple_4)
```

```
114 print(tuple_4)
```

```
115 print(tuple_4)
```

```
116 print(tuple_4)
```

```
117 print(tuple_4)
```

```
118 print(tuple_4)
```

```
119 print(tuple_4)
```

```
120 print(tuple_4)
```

```
121 print(tuple_4)
```

```
122 print(tuple_4)
```

```
123 print(tuple_4)
```

```
124 print(tuple_4)
```

```
125 print(tuple_4)
```

```
126 print(tuple_4)
```

```
127 print(tuple_4)
```

```
128 print(tuple_4)
```

```
129 print(tuple_4)
```

```
130 print(tuple_4)
```

```
131 print(tuple_4)
```

```
132 print(tuple_4)
```

```
133 print(tuple_4)
```

```
134 print(tuple_4)
```

```
135 print(tuple_4)
```

```
136 print(tuple_4)
```

```
137 print(tuple_4)
```

```
138 print(tuple_4)
```

```
139 print(tuple_4)
```

```
140 print(tuple_4)
```

```
141 print(tuple_4)
```

```
142 print(tuple_4)
```

```
143 print(tuple_4)
```

```
144 print(tuple_4)
```

```
145 print(tuple_4)
```

```
146 print(tuple_4)
```

```
147 print(tuple_4)
```

```
148 print(tuple_4)
```

```
149 print(tuple_4)
```

```
150 print(tuple_4)
```

```
151 print(tuple_4)
```

```
152 print(tuple_4)
```

```
153 print(tuple_4)
```

```
154 print(tuple_4)
```

Delete a tuple

An element in a tuple can not be deleted since it is immutable.

But a whole tuple can be deleted

```
● ● ●  
tuple_4 = (1,3,5,7,8)  
print('Before deleting:', tuple_4)  
del tuple_4  
print('After deleting:', tuple_4)
```

Before deleting: (1, 3, 5, 7, 8)

NameError

Input In [22], in <cell line: 4>()
2 print('Before deleting:', tuple_4)
3 del tuple_4
----> 4 print('After deleting:', tuple_4)

Traceback (most recent call last)

NameError: name 'tuple_4' is not defined

count() method

This method returns the number of time an item occurs in a tuple.

```
● ● ●  
tuple_5 = (1,1,3,3,5,5,5,5,6,6,7,8,9)  
tuple_5.count(5)
```

4

index() method

It returns the index of the first occurrence of the specified value in a tuple

```
● ● ●  
tuple_5 = (1,1,3,3,5,5,5,5,6,6,7,8,9)  
print(tuple_5.index(5))  
print(tuple_5.index(1))  
print(tuple_5.index(9))
```

4
0
12

One element tuple

If a tuple includes only one element, you should put a comma after the element.
Otherwise, it is not considered as a tuple.

```
● ● ●  
tuple_6 = (0)  
print(tuple_6)  
print(type(tuple_6))  
# Here, you see that the output is an integer
```

0
<class 'int'>

```
● ● ●  
tuple_7 = (0,)  
print(tuple_7)  
print(type(tuple_7))  
# You see that the output is a tuple
```

(0,)
<class 'tuple'>