Кортежи
(класс tuple)

Кортеж — это неизменяемый список.

```
foo = 5, 10, 12, 0
bar = (5, 10, 12, 0)

print("type(foo):", type(foo))
print("type(bar):", type(bar))
print("foo:      ", foo)
print("bar:      ", bar)
```

```
type(foo): <class 'tuple'>
type(bar): <class 'tuple'>
foo:       (5, 10, 12, 0)
bar:       (5, 10, 12, 0)
```

Попытка изменить содержимое кортежа

```
foo = (5, 10, 12, 0)
foo[0] = 100
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

```
foo = (5, [], 12, 0)
print("foo:", foo)

foo[1].append(10)
print("foo:", foo)
```

```
foo: (5, [], 12, 0)
foo: (5, [10], 12, 0)
```

```
foo = (5, [], 12, 0)
print("foo:", foo)

x = foo[1]
x.append(10)
print("foo:", foo)
```

```
foo: (5, [], 12, 0)
foo: (5, [10], 12, 0)
```

Содержимое класса tuple

```
print(dir(tuple))
```

['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']

Распаковка (unpacking)

```
foo = (10, 20, 30)
a, b, c = foo

print("a:", a)
print("b:", b)
print("c:", c)
```

```
a: 10
b: 20
c: 30
```

Распаковка (unpacking)

```
foo = [10, 20, 30]
a, b, c = foo

print("a:", a)
print("b:", b)
print("c:", c)
```

```
a: 10
b: 20
c: 30
```

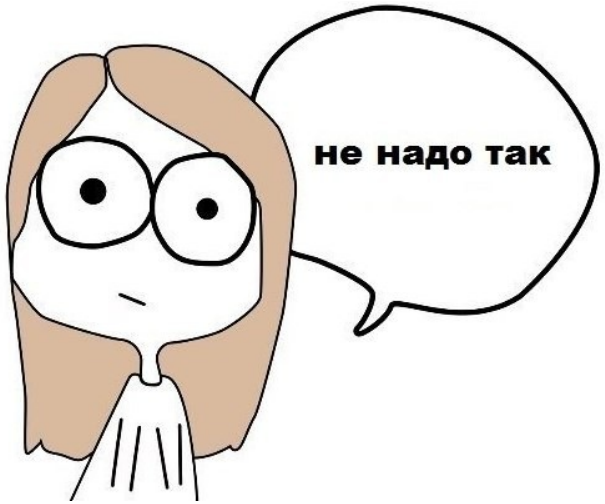
```python
# Обмен значениями
x = 10
y = 20

x, y = y, x

print("x:", x, "y:", y)
```

```
x: 20 y: 10
```

Перебор элементов с использованием индексов (не оптимальный способ)


```
foo = [5, 10, 12, 0, 17, 22]
for n in range(len(foo)):
    print(n, "->", foo[n])
```

```
0 -> 5
1 -> 10
2 -> 12
3 -> 0
4 -> 17
5 -> 22
```



Использование класса enumerate

```
foo = [5, 10, 12, 0, 17, 22]
for n, item in enumerate(foo):
    print(n, "->", item)
```

```
0 -> 5
1 -> 10
2 -> 12
3 -> 0
4 -> 17
5 -> 22
```

```
foo = [5, 10, 12, 0, 17, 22]
bar = list(enumerate(foo))

print("bar:", bar)
```

```
bar: [(0, 5), (1, 10), (2, 12), (3, 0), (4, 17), (5, 22)]
```

```
foo = [5, 10, 12, 0, 17, 22]
for x in enumerate(foo):
    print(x[0], "->", x[1])
```

```
0 -> 5
1 -> 10
2 -> 12
3 -> 0
4 -> 17
5 -> 22
```

Класс zip

zip(**iterables, strict=False*)

Iterate over several iterables in parallel, producing tuples with an item from each one.

Класс zip

zip(**iterables, strict=False*)

Iterate over several iterables in parallel, producing tuples with an item from each one.

```
foo = [10, 20, 30, 40]
bar = ["hello", "world", "from", "Python"]
spam = zip(foo, bar)

print("type(spam):", type(spam))
print("list(spam):", list(spam))
```

```
type(spam): <class 'zip'>
list(spam): [(10, 'hello'), (20, 'world'), (30, 'from'), (40, 'Python')]
```

```
foo = [10, 20, 30, 40]
bar = ["hello", "world", "from", "Python"]

for f, b in zip(foo, bar):
    print(f, "->", b)
```

```
10 -> hello
20 -> world
30 -> from
40 -> Python
```

```
foo = [10, 20, 30, 40]
bar = ["hello", "world", "from", "Python"]
spam = [5.5, 4.2, 3.3, 0.5]

for f, b, s in zip(foo, bar, spam):
    print(f, "->", b, "->", s)
```

```
10 -> hello -> 5.5
20 -> world -> 4.2
30 -> from -> 3.3
40 -> Python -> 0.5
```

```
foo = [10, 20, 30, 40]
print("list(zip(foo)):", list(zip(foo)))
```

```
list(zip(foo)): [(10,), (20,), (30,), (40,)]
```