

Python. Кортежи

Кортежи

```
user = ("Tom", 23)
```

```
print(user)
```

```
user = "Tom", 23
```

```
print(user)
```

#создание кортежа из одного элемента

```
user = ("Tom",)
```

Кортежи. Кортеж из списка

```
users_list = ["Tom", "Bob", "Kate"]  
users_tuple = tuple(users_list)  
print(users_tuple)  
# ("Tom", "Bob", "Kate")
```

Кортежи. Элементы кортежей

```
users = ("Tom", "Bob", "Sam", "Kate")  
print(users[0])          # Tom  
print(users[2])          # Sam  
print(users[-1])         # Kate  
  
# получим часть кортежа со 2 элемента по 4  
print(users[1:4])  
# ("Bob", "Sam", "Kate")
```

Кортежи. Неизменяемость кортежей

```
users = ("Tom", "Bob", "Sam", "Kate")
```

```
#следующая строка работать не будет
```

```
users[1] = "Tim"
```

Кортежи. Разложение кортежа

```
user = ("Tom", 22, False)
name, age, isMarried = user
print(name)           # Tom
print(age)            # 22
print(isMarried)      # False
```

Кортежи. Использование кортежа в функции

```
def get_user():  
    name = "Tom"  
    age = 22  
    is_married = False  
    return name, age, is_married
```

Кортежи. Использование кортежа в функции (продолжение)

```
user = get_user()
print(user[0])      # Tom
print(user[1])      # 22
print(user[2])      # False
```


Кортежи. Использование кортежа в функции (продолжение)

```
name, age, is_married = get_user()
print(name)           # Tom
print(age)            # 22
print(is_married)     # False
```

Кортежи. Функция len()

```
user = ("Tom", 22, False)
print(len(user))           # 3
```

Кортежи. Проверка наличия элемента

```
user = ("Tom", 22, False)
name = "Tom"

if name in user:
    print("Пользователя зовут Tom")
else:
    print("Пользователь имеет другое имя")
```

Кортежи. Перебор кортежей

```
user = ("Tom", 22, False)
for item in user:
    print(item)
```

Кортежи. Перебор кортежей

```
user = ("Tom", 22, False)
```

```
i = 0
```

```
while i < len(user):
```

```
    print(user[i])
```

```
    i += 1
```

Кортежи. Перебор кортежей

```
user = ("Tom", 22, False)
```

```
print(*user)
```

```
print(*user, sep='\n')
```

Кортежи. Операция конкатенации + и умножения на число *

```
print((1, 2, 3, 4) + (5, 6, 7, 8))
```

```
print((7, 8) * 3)
```

```
print((0,) * 10)
```

```
#(1, 2, 3, 4, 5, 6, 7, 8)
```

```
#(7, 8, 7, 8, 7, 8)
```

```
#(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
```

Кортежи. Операция конкатенации + и умножения на число *

```
a = (1, 2, 3, 4)
```

```
b = (7, 8)
```

```
a += b      # добавляем к кортежу a кортеж b
```

```
b *= 5      # повторяем кортеж b 5 раз
```

```
print(a)
```

```
print(b)
```


Кортежи. Встроенные функции `sum()`, `min()`, `max()`

```
numbers = (3, 4, 10, 3333, 12, -7, -5, 4)
```

```
print('Сумма всех элементов кортежа =', sum(numbers))
```

```
print('Минимальный элемент кортежа =', min(numbers))
```

```
print('Максимальный элемент кортежа =', max(numbers))
```

Кортежи. Метод index()

```
names = ('Gvido', 'Roman', 'Timur')
```

```
position = names.index('Timur')
```

```
print(position) #2
```

```
position = names.index('Anders')
```

```
print(position)
```

```
#ValueError: tuple.index(x): x not in tuple
```

Кортежи. Метод index()

```
names = ('Gvido', 'Roman', 'Timur')
if 'Anders' in names:
    position = names.index('Anders')
    print(position)
else:
    print('Такого значения нет в кортеже')
```

Кортежи. Метод count()

```
names = ('Timur', 'Gvido', 'Roman', 'Timur',  
         'Anders', 'Timur')
```

```
cnt1 = names.count('Timur') #3
```

```
cnt2 = names.count('Gvido') #1
```

```
cnt3 = names.count('Josef') #0
```

Кортежи. Вложенные кортежи

```
countries = (  
    ("Germany", 80.2, (("Berlin", 3.326),  
    ("Hamburg", 1.718))),  
    ("France", 66, (("Paris", 2.2), ("Marsel",  
1.6)))  
)
```

Кортежи. Вложенные кортежи (продолжение)

```
for country in countries:
    countryName, countryPopulation, cities =
country

    print("\nCountry: {} population:
{}").format(countryName, countryPopulation)

    for city in cities:
        cityName, cityPopulation = city

        print("City: {} population:
{}").format(cityName, cityPopulation)
```

Кортежи. Сравнение кортежей

```
print((1, 8) == (1, 8))  
print((1, 8) != (1, 10))  
print((1, 9) < (1, 2))  
print((2, 5) < (6,))  
print(('a', 'bc') > ('a', 'de'))
```

Кортежи. Сортировка кортежей

```
not_sorted_tuple = (34, 1, 8, 67, 5, 9, 0, 23)
```

```
print(not_sorted_tuple)
```

```
sorted_tuple = tuple(sorted(not_sorted_tuple))
```

```
print(sorted_tuple)
```


Кортежи. Сортировка кортежей

```
not_sorted_tuple = ('cc', 'aa', 'dd', 'bb')  
tmp = list(not_sorted_tuple)  
tmp.sort()  
  
sorted_tuple = tuple(tmp)  
print(sorted_tuple)
```

Кортежи. Преобразование кортежа в строку и наоборот

```
notes = ('Do', 'Re', 'Mi', 'Fa', 'Sol', 'La', 'Si')
```

```
string1 = ''.join(notes)
```

```
#DoReMiFaSolLaSi
```

```
string2 = ' '.join(notes)
```

```
#Do.Re.Mi.Fa.Sol.La.Si
```

Кортежи. Преобразование кортежа в строку и наоборот

```
letters = 'abcdefghijkl'  
tpl = tuple(letters)  
#('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',  
  'k', 'l')
```

Кортежи. Упаковка кортежей

```
tuple1 = 1, 2, 3
```

```
tuple2 = 'b',
```

```
print(type(tuple1)) #<class 'tuple'>
```

```
print(type(tuple2)) #<class 'tuple'>
```

Кортежи. Распаковка кортежей

```
colors = ('red', 'green', 'blue', 'cyan')
```

```
(a, b, c, d) = colors
```

```
print(a)    #red
```

```
print(b)    #green
```

```
print(c)    #blue
```

```
print(d)    #cyan
```

Кортежи. Распаковка кортежей

```
colors = ('red', 'green', 'blue', 'cyan')
```

```
a, b, c, d = colors
```

```
colors = ('red', 'green', 'blue', 'cyan')
```

```
a, b = colors      #ValueError: ...
```

```
colors = ('red', 'green', 'blue')
```

```
a, b, c, d = colors  #ValueError:...
```

Кортежи. Распаковка кортежей

```
colors = ('red', 'green', 'blue')
```

```
a, b, _ = colors
```

```
print(a)
```

```
print(b)
```

Кортежи. Распаковка кортежей

```
a, b, c = 3, 2, 1
```

```
b, a, c = c, a, b
```

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


Кортежи. * при распаковке кортежей

```
a, b, *tail = 1, 2, 3, 4, 5, 6
```

```
# a = 1
```

```
# b = 2
```

```
# tail = [3, 4, 5, 6]
```

Кортежи. * при распаковке кортежей

```
a, b, *tail = 1, 2, 3
```

```
# a = 1
```

```
# b = 2
```

```
# tail = [3]
```

Кортежи. * при распаковке кортежей

```
a, b, *tail = 1, 2
```

```
# a = 1
```

```
# b = 2
```

```
# tail = []
```

Кортежи. * при распаковке кортежей

```
*names, surname = ('Стефани', 'Джоанн',  
'Анджелина', 'Джерманотта')
```

```
print(names)
```

```
# ['Стефани', 'Джоанн', 'Анджелина']
```

```
print(surname)
```

```
# Джерманотта
```

Кортежи. * при распаковке кортежей

```
singer = ('Freddie', 'Bohemian Rhapsody', 'Killer  
Queen', 'Love of my life', 'Mercury')
```

```
name, *songs, surname = singer
```

```
print(name) # Freddie
```

```
print(songs) # ['Bohemian Rhapsody', 'Killer  
Queen', 'Love of my life']
```

```
print(surname) # Mercury
```

Python. Словари

Словари

```
dictionary = { ключ1:значение1, ключ2:  
значение2, .... }
```

```
users = {1: "Tom", 2: "Bob", 3: "Bill"}
```

```
elements = {"Au": "Золото", "Fe": "Железо",  
"H": "Водород", "O": "Кислород"}
```

Словари

```
objects = {1: "Tom", "2": True, 3: 100.6}
```

```
objects = {}
```

```
objects = dict()
```


Словари. Преобразование из списка в словарь

```
users_list = [  
    ["+111123455", "Tom"],  
    ["+384767557", "Bob"],  
    ["+958758767", "Alice"]  
]  
  
users_dict = dict(users_list)  
  
print(users_dict)    # {"+111123455": "Tom",  
    "+384767557": "Bob", "+958758767": "Alice"}
```

Словари. Преобразование из кортежей в словарь

```
users_tuple = (  
    "+111123455", "Tom",  
    "+384767557", "Bob",  
    "+958758767", "Alice"  
)  
users_dict = dict(users_tuple)  
print(users_dict)
```

Словари. Получение и изменение элементов

```
users = {  
    "+11111111": "Tom",  
    "+33333333": "Bob",  
    "+55555555": "Alice"  
}
```

Словари. Получение и изменение элементов (продолжение)

```
# получаем элемент с ключом "+11111111"
```

```
print(users["+11111111"])          # Tom
```

```
# установка значения элемента с ключом  
"+33333333"
```

```
users["+33333333"] = "Bob Smith"
```

```
print(users["+33333333"])          # Bob Smith
```

Словари. Получение и изменение элементов (продолжение)

```
users["+4444444"] = "Sam"
```

```
user = users["+445"]      # KeyError
```

Словари. Получение и изменение элементов (продолжение)

```
key = "+445"  
  
if key in users:  
    user = users[key]  
    print(user)  
  
else:  
    print("Элемент не найден")
```

Словари. Получение и изменение элементов. Метод `get()`

`get(key)` : возвращает из словаря элемент с ключом `key`.
Если элемента с таким ключом нет, то возвращает значение `None`

`get(key, default)` : возвращает из словаря элемент с ключом `key`. Если элемента с таким ключом нет, то возвращает значение по умолчанию `default`

Словари. Получение и изменение элементов. Метод get()

```
key = "+555555555"
```

```
user = users.get(key)
```

```
user = users.get(key, "Unknown user")
```


Словари. Удаление

```
users = {  
    "+11111111": "Tom",  
    "+33333333": "Bob",  
    "+55555555": "Alice"  
}
```

```
del users["+55555555"]  
print(users)
```

Словари. Удаление

```
key = "+55555555"
if key in users:
    user = users[key]
    del users[key]
    print(user, "удален")
else:
    print("Элемент не найден")
```

Словари. Удаление. Метод pop()

`pop(key)` : удаляет элемент по ключу `key` и возвращает удаленный элемент. Если элемент с данным ключом отсутствует, то генерируется исключение `KeyError`

`pop(key, default)` : удаляет элемент по ключу `key` и возвращает удаленный элемент. Если элемент с данным ключом отсутствует, то возвращается значение `default`

Словари. Удаление. Метод pop()

```
users = {  
    "+11111111": "Tom",  
    "+33333333": "Bob",  
    "+55555555": "Alice"  
}
```

Словари. Удаление. Метод pop() (продолжение)

```
key = "+55555555"  
user = users.pop(key)  
print(user)
```

```
user = users.pop("+4444444", "Unknown user")  
print(user)
```

Словари. Удаление. Метод clear()

#удаление всех элементов

```
users.clear()
```

Словари. Копирование и объединение словарей. Метод `copy()`

```
users = {"+1111111": "Tom", "+3333333":  
"Bob", "+5555555": "Alice"}  
users2 = users.copy()
```

Словари. Копирование и объединение словарей. Метод update()

```
users = {"+1111111": "Tom", "+3333333":  
"Bob", "+5555555": "Alice"}  
  
users2 = {"+2222222": "Sam", "+6666666": "Kate"}  
  
users.update(users2)  
  
print(users)      # {"+1111111": "Tom", "+3333333":  
"Bob", "+5555555": "Alice", "+2222222": "Sam",  
"+6666666": "Kate"}  
  
print(users2)     # {"+2222222": "Sam", "+6666666":  
"Kate"}
```


Словари. Копирование и объединение словарей. Метод `update()` (продолжение)

```
users3 = users.copy()  
users3.update(users2)
```

Словари. Перебор словаря

```
users = {  
    "+11111111": "Tom",  
    "+33333333": "Bob",  
    "+55555555": "Alice"  
}  
  
for key in users:  
    print(key, " - ", users[key])
```

Словари. Перебор словаря. Метод items()

```
for key, value in users.items():  
    print(key, " - ", value)
```

Словари. Перебор словаря. Методы keys() и values()

```
for key in users.keys():  
    print(key)
```

```
for value in users.values():  
    print(value)
```

Словари. Комплексные словари

```
users = {  
    "Tom": {  
        "phone": "+971478745",  
        "email": "tom12@gmail.com"  
    },  
}
```

Словари. Комплексные словари (продолжение)

```
"Bob" : {  
    "phone": "+876390444",  
    "email": "bob@gmail.com",  
    "skype": "bob123"  
}
```

```
}
```

Словари. Комплексные словари (продолжение)

```
old_email = users["Tom"]["email"]
```

```
users["Tom"]["email"] = supertom@gmail.com
```

```
tom_skype = users["Tom"]["skype"]      # KeyError
```

Словари. Комплексные словари (продолжение)

```
key = "skype"
if key in users["Tom"]:
    print(users["Tom"]["skype"])
else:
    print("skype is not found")
```


Python. Множества

Множества

```
users = {"Tom", "Bob", "Alice", "Tom"}
```

```
print(users)      # {"Tom", "Bob", "Alice"}
```

```
users2 = set(["Mike", "Bill", "Ted"])
```

```
users3 = set() #пустое множество
```

```
users4 = {} #пустой словарь
```

Множества. Функция len()

```
users = { "Tom", "Bob", "Alice" }
```

```
print(len(users) )    # 3
```

Множества. Добавление элементов

```
users = set()  
users.add("Sam")  
print(users)
```

Множества. Удаление элементов

```
users = {"Tom", "Bob", "Alice"}

user = "Tom"

if user in users:
    users.remove(user)

print(users)      # {"Bob", "Alice"}
```

Множества. Удаление элементов

```
user = "Tim"
```

```
#удаление без генерации исключения
```

```
users.discard(user)
```

```
#удаление всех элементов
```

```
users.clear()
```

Множества. Перебор элементов

```
users = { "Tom", "Bob", "Alice" }
```

```
for user in users:  
    print(user)
```

Множества. Операции с множествами

```
users = { "Tom", "Bob", "Alice" }  
users3 = users.copy()
```


Множества. Операции с множествами

```
users = { "Tom", "Bob", "Alice" }
```

```
users2 = { "Sam", "Kate", "Bob" }
```

```
users3 = users.union(users2)
```

```
print(users3)
```

```
# { "Bob", "Alice", "Sam", "Kate", "Tom" }
```

Множества. Операции с множествами

```
users = { "Tom", "Bob", "Alice" }  
users2 = { "Sam", "Kate", "Bob" }  
  
users3 = users.intersection(users2)  
print(users3)      # { "Bob" }
```

Множества. Операции с множествами

```
users = {"Tom", "Bob", "Alice"}  
users2 = {"Sam", "Kate", "Bob"}  
  
print(users & users2)      # {"Bob"}
```

Множества. Операции с множествами

```
users = { "Tom", "Bob", "Alice" }  
users2 = { "Sam", "Kate", "Bob" }  
  
users3 = users.difference(users2)  
print(users3)           # { "Tom", "Alice" }  
print(users - users2)   # { "Tom", "Alice" }
```

Множества. Отношения между множествами

```
users = {"Tom", "Bob", "Alice"}
superusers = {"Sam", "Tom", "Bob", "Alice",
              "Greg"}

print(users.issubset(superusers))    # True
print(superusers.issubset(users))    # False
```

Множества. Отношения между множествами

```
users = {"Tom", "Bob", "Alice"}  
superusers = {"Sam", "Tom", "Bob", "Alice",  
              "Greg"}  
  
print(users.issuperset(superusers))    # False  
print(superusers.issuperset(users))    # True
```

Множества. frozen set

```
users = frozenset({"Tom", "Bob", "Alice"})
```

Множества. frozen set.

Поддерживаемые операции

`len(s)` : возвращает длину множества

`x in s` : возвращает True, если элемент x присутствует в МНОЖЕСТВЕ s

`x not in s` : возвращает True, если элемент x отсутствует в МНОЖЕСТВЕ s

`s.issubset(t)` : возвращает True, если t содержит множество s

`s.issuperset(t)` : возвращает True, если t содержится в МНОЖЕСТВЕ s

Множества. frozen set.

Поддерживаемые операции

`s.union(t)` : возвращает объединение множеств `s` и `t`

`s.intersection(t)` : возвращает пересечение множеств `s` и `t`

`s.difference(t)` : возвращает разность множеств `s` и `t`

`s.copy()` : возвращает копию множества `s`