Московский авиационный институт (национальный исследовательский университет)

Основные структуры данных (коллекции) в Python

Множества (sets)

Множество — это структура данных, предназначенная для *неупорядоченного* хранения *уникальных* элементов

В Python множество представлено классом **set**

Способы создания множеств

```
# Создание множеств на основе итерируемых объектов 4
foo = set([100, 10, 20, 30, 30, 10])
print("foo:", foo)
bar = set("abracadabra")
print("bar:", bar)
spam = set([100, "hello", 42, 3.14])
```

print("spam:", spam)

foo: {10, 100, 20, 30}

bar: {'c', 'b', 'r', 'a', 'd'}

spam: {3.14, 42, 'hello', 100}

```
# Создание множеств с использованием фигурных скобок
                                                                         5
foo = \{100, 10, 20, 30, 30, 10\}
print("type(foo): ", type(foo))
print()
bar = {item + 1 for item in [100, 10, 20, 30, 30, 10]}
print("type(bar): ", type(bar))
print("bar: ", bar)
print()
# !!! Следующая строка создаст словарь
spam = \{\}
print("type(spam): ", type(spam))
type(foo): <class 'set'>
foo: {10, 100, 20, 30}
type(bar): <class 'set'>
bar:
    {21, 11, 101, 31}
type(spam): <class 'dict'>
```

```
# Создание пустого множества
foo = set()
bar = set([])

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

```
type(foo): <class 'set'>
foo: set()
bar: set()
```

print("bar:", bar)

```
# Не все типы могут входить в множество
# Ошибка!
foo = {1, 2, [100, 200]}
```

```
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

```
# Кортежи могут входить в множество
```

```
foo = {1, 2, (100, 200)}
print("foo:", foo)
```

foo: {1, 2, (100, 200)}

```
>>> foo = "Hello, world"
>>> bar = (1, 2, 3)
>>> baz = (1, 2, 3)
>>>  bam = (0, 2, 3)
>>> eggs = 13.5
>>> spam = 150
>>> hash(foo)
3386384232892847944
>>> hash(bar)
529344067295497451
>>> hash(baz)
529344067295497451
>>> hash(bam)
8477238830141211852
>>> hash(eggs)
1152921504606846989
>>> hash(spam)
150
```

```
>>> foo = [10, 20, 30]
>>> hash(foo)
Traceback (most recent call last):
  File "<...>", line 1, in <module>
    hash(foo)
    ~~~^^^^
TypeError: unhashable type: 'list'
```

```
>>> foo = (10, 20, [100, 200])
>>> hash(foo)
Traceback (most recent call last):
  File "<...>", line 1, in <module>
    hash(foo)
    ~~~^^^^
TypeError: unhashable type: 'list'
```

Операции со множествами

13

100 in foo: True

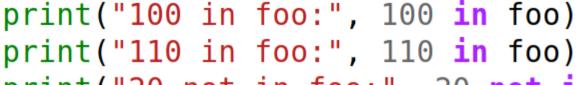
110 in foo: False

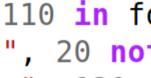
20 not in foo: False

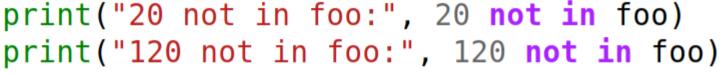
120 not in foo: True



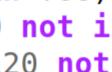












Проверка наличия элемента в множестве

```
# Перебор элементов множества foo = {100, 10, 20, 30, 30, 10}

for item in foo: print(item)
```

```
# Определение количества элементов множества foo = {100, 10, 20, 30, 30, 10}

print("foo:", foo) print("len(foo):", len(foo))
```

```
foo: {10, 100, 20, 30}
len(foo): 4
```

print(dir(set))

```
and ', ' class ', ' class getitem ', ' contains ', ' delattr ',
         ' doc ', ' eq ', ' format ', ' ge ', ' getattribute
  getstate ',' gt ',' hash ',' iand ',' init ',' init subclass
           isub ', ' iter ', ' ixor ', ' le ', ' len ', ' lt
           new ', ' or ', ' rand ', ' reduce ', ' reduce ex ',
  repr ', ' ror ', ' rsub ', ' rxor ', ' setattr ', ' sizeof ',
  str_', '_sub_', ' subclasshook ', ' xor ', 'add', 'clear', 'copy',
'difference', 'difference update', 'discard', 'intersection', 'intersection update',
'isdisjoint', 'issubset', 'issuperset', 'pop', 'remove', 'symmetric difference',
'symmetric difference update', 'union', 'update']
```

```
# Добавление элемента в множество foo = {100, 10, 20, 30, 30, 10}

foo.add(0)
print("foo:", foo)
```

```
foo: {0, 100, 10, 20, 30}
```

```
18
# Добавление элементов из других объектов
foo = \{10, 20, 30\}
bar = [10, 30, 40]
spam = \{40, 50, 60\}
print("1) foo:", foo)
foo.update(bar)
print("2) foo:", foo)
foo.update(spam)
print("3) foo:", foo)
1) foo: {10, 20, 30}
2) foo: {40, 10, 20, 30}
3) foo: {50, 20, 40, 10, 60, 30}
```

```
# Добавление элементов из других объектов foo = {10, 20, 30} bar = [10, 30, 40] spam = {40, 50, 60}
```

```
print("1) foo:", foo)

foo.update(bar, spam)
print("2) foo:", foo)
```

1) foo: {10, 20, 30}
2) foo: {50, 20, 40, 10, 60, 30}

```
# Объединение множеств
foo = {10, 20, 30}
bar = {10, 30, 40}
spam = foo | bar
```

```
spam: {20, 40, 10, 30}
```

print("spam:", spam)

```
# Объединение множеств
foo = {10, 20, 30}
bar = {10, 30, 40}
foo |= bar
```

```
foo: {20, 40, 10, 30}
```

```
# Удаление элемента из множества foo = {100, 10, 20, 30, 30, 10}

foo.remove(100) print("foo:", foo)
```

foo: {10, 20, 30}

```
# Попытка удалить несуществующий элемент foo = {100, 10, 20, 30, 30, 10}
# Ошибка!
```

```
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
KeyError: 42
```

foo.remove(42)

```
# Удаление элемента из множества с проверкой наличия элемента foo = {100, 10, 20, 30, 30, 10}
foo.discard(100)
```

foo.discard(0)
print("2) foo:", foo)

1) foo: {10, 20, 30} 2) foo: {10, 20, 30}

```
# Удаление всех элементов из множества foo = {100, 10, 20, 30, 30, 10}

foo.clear() print("foo:", foo)
```

foo: set()

```
# Получение разности множеств
foo = {10, 20, 30}
bar = {10, 30, 40}
spam = foo - bar
```

print("spam:", spam)

```
spam: {20}
```

```
# Получение разности множеств foo = {10, 20, 30} bar = {10, 30, 40}
```

```
foo -= bar
print("foo:", foo)
```

foo: {20}

```
# Нахождение пересечения множеств
foo = {10, 20, 30}
bar = {10, 30, 40}
```

```
spam: {10, 30}
```

spam = foo & bar

print("spam:", spam)

```
foo = \{10, 20, 30\}
bar = \{10, 30, 40\}
foo &= bar
```

foo: {10, 30}

```
# Получение симметричной разности множеств
foo = {10, 20, 30}
bar = {10, 30, 40}
```

```
spam: {40, 20}
```

spam = foo ^ bar

print("spam:", spam)

```
# Получение симметричной разности множеств foo = {10, 20, 30} bar = {10, 30, 40}
```

```
foo: {40, 20}
```

32

$foo = \{10, 20, 30\}$ $bar = \{10, 30, 40\}$ $spam = \{30, 10\}$ $baz = \{40, 10\}$ eggs = $\{30, 20, 10\}$

print("foo > spam: ", foo > spam)

print("foo > eggs: ", foo > eggs)

print("foo >= eggs: ", foo >= eggs)

print("foo == eggs: ", foo == eggs)

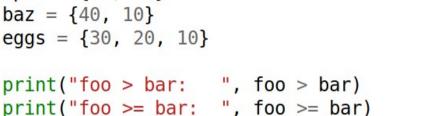
print("foo is eggs: ", foo is eggs)

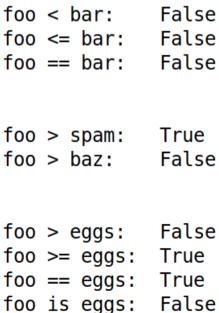
print("foo > baz: ", foo > baz)

print()

print()

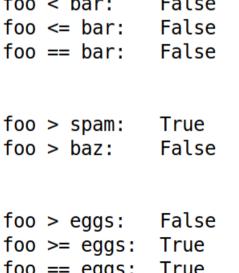
Сравнение множеств





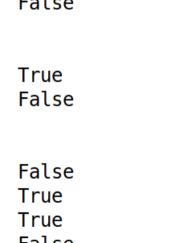
foo > bar:

foo >= bar:



False

False



Hеизменяемые множества. Класс frozenset

 $bar = frozenset(\{10, 20, 30, 10\})$

frozenset()

type(spam): <class 'frozenset'>

foo:

bar:

spam:

type(bar):

34

spam = frozenset([20, 30, 42, 42])

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

type(foo): <class 'frozenset'>

<class 'frozenset'>

frozenset({42, 20, 30})

frozenset({10, 20, 30})

print(dir(frozenset))

```
and ', ' class ', ' class getitem ', ' contains
  delattr ', ' dir ', ' doc ', ' eq ', ' format ', ' ge
  getattribute ', ' getstate ', ' gt ', ' hash ', '
  init subclass ', ' iter ', ' le ', ' len ', ' lt ', '
  new ', ' or ', ' rand ', ' reduce ', ' reduce ex ',
  repr ',' ror ',' rsub ',' rxor ',' setattr ',' sizeof ',
  str ',' sub ',' subclasshook ',' xor ','copy','difference',
'intersection', 'isdisjoint', 'issubset', 'issuperset',
'symmetric difference', 'union']
```

```
# Попытка "изменения" frozenset foo = frozenset({10, 20, 30}) bar = {10, 30, 40}
```

```
foo: frozenset({20, 40, 10, 30})
```

```
# Попытка "изменения" frozenset
foo = frozenset({10, 20, 30})
bar = {10, 30, 40}
foo = foo | bar
```

```
foo: frozenset({20, 40, 10, 30})
```

```
# Попытка "изменения" frozenset
foo = frozenset({10, 20, 30})
bar = {10, 30, 40}
```

```
spam = foo | bar
foo = spam
```

```
foo: frozenset({20, 40, 10, 30})
```

```
# Попытка "изменения" frozenset
foo = frozenset(\{10, 20, 30\})
bar = \{10, 30, 40\}
spam = foo
print("1) spam is foo:", spam is foo)
print(" foo: ", foo)
print(" spam: ", spam)
print(" id(foo):", id(foo))
foo |= bar
print("2) spam is foo:", spam is foo)
print(" foo: ", foo)
print(" spam: ", spam)
print(" id(foo):", id(foo))
```

```
1) spam is foo: True
            frozenset({10, 20, 30})
   foo:
   spam: frozenset({10, 20, 30})
   id(foo): 139786597260128
2) spam is foo: False
   foo:
            frozenset({20, 40, 10, 30})
            frozenset({10, 20, 30})
   spam:
   id(foo): 139786597257664
```

```
# frozenset - хэшируемый объект foo = frozenset({10, 20, 30}) bar = frozenset({20, 30, 40})

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

spam = {foo: "foo value", bar: "bar value"}

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

hash(foo): 1046241836650535896 hash(bar): 1287282707040401033 spam: {frozenset({10, 20, 30}): 'foo_value', frozenset({40, 20, 30}): 'bar_value'}