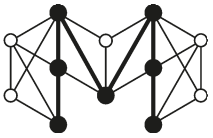
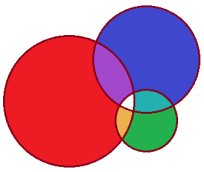
Программирование — 2026.



Множества



1. Описание/создание

In [1]:
```
A = {2, 4, 6, 8, 10, 12, 14, 16, 2}
B = {'Привет-привет!', 'Hello!', 'Привет', 3.1415}
print(A)
print(B)
print(type(A))
```

```
{2, 4, 6, 8, 10, 12, 14, 16}
{'Привет-привет!', 3.1415, 'Hello!', 'Привет'}
<class 'set'>
```

In [ ]:
```
empty = set() # пустое множество
```

1.1 Из строк или списков

In [2]:
```
C = set('Привет-привет!')
D = set([1,2,3,4,5,1,2,3,4,5])
print(C, D)
```

```
{'и', 'в', 'т', '-', '!', 'р', 'П', 'п', 'е'} {1, 2, 3, 4, 5}
```

1.2 Генераторы

In [3]:
```
E = set(i for i in range(1,10) if i%2!=0)
F = {i for i in range(1,10) if i%2!=0}
print(E, F)
```

```
{1, 3, 5, 7, 9} {1, 3, 5, 7, 9}
```

Важно: элементы множества – неизменяемые. Не бывает множества списков.

In [4]:
```
A = {[1,2], [3,4]} # Ошибка!
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 A = {[1,2], [3,4]} # Ошибка!

TypeError: cannot use 'list' as a set element (unhashable type: 'list')
```

2. Операции над множествами

Размер

In [5]:
```
# Размер (количество элементов)
n = len(A)
print(n)
```

```
8
```

In [6]:
```
# Добавление элемента
A.add(18)
print(A)
```

```
{2, 4, 6, 8, 10, 12, 14, 16, 18}
```

In [7]:
```
# Пример:
# альтернативное создание множества A
A = set()
for i in range(2,20,2):
    A.add(i)
print(A)
```

```
{2, 4, 6, 8, 10, 12, 14, 16, 18}
```

In [8]:
```
# Принадлежность элемента множеству
print(2 in A)
print('Привет' in B)
```

```
True
True
```

In [9]:
```
# создадим два множества
# для иллюстрации остальных операций
A = {i for i in range(11)}
B = {2**i for i in range(6)}
print(f'A = {A}\nB = {B}')
```

```
A = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
B = {32, 1, 2, 4, 8, 16}
```

In [10]:
```
# Объединение множеств
# способ 1
AB1 = A | B
# способ 2
AB2 = A.union(B)
print(f'A U B  = {AB1}\nA U B  = {AB2}')
```

```
A U B  = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 32, 16}
A U B  = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 32, 16}
```

In [11]:
```
# Пересечение множеств
# способ 1
AB1 = A & B
# способ 2
AB2 = A.intersection(B)
print(f'A ∩ B  = {AB1}\nA ∩ B  = {AB2}')
```

```
A ∩ B  = {8, 1, 2, 4}
A ∩ B  = {8, 1, 2, 4}
```

In [13]:
```
# Разность множеств
# способ 1
AB1 = A - B
# способ 2
AB2 = A.difference(B)
print(f'A \\ B  = {AB1}\nA \\ B  = {AB2}')
```

```
A \ B  = {0, 3, 5, 6, 7, 9, 10}
A \ B  = {0, 3, 5, 6, 7, 9, 10}
```

In [14]:
```
# Аналоги операций a += b, a -= b  и т.п.

# удаляем из множества A все элементы,
# входящие в множество B
A -= B
# или так
A.difference_update(B)
print(A)

# добавляем в множество A
# всех элементов из множества B
A |= B
# или так
A.update(B)
print(A)

# оставляем в множестве A только
# те элеменнты, которые есть в B
A &= B
# или так
A.intersection_update(B)
print(A)
```

```
{0, 3, 5, 6, 7, 9, 10}
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16, 32}
{32, 1, 2, 4, 8, 16}
```

In [15]:
```
A = {1,2,3,4,5}
B = {5,4,3,2,1}
# Проверка на вложение
print(A <= B)
# или так:
print(A.issubset(B))
print(A < B)
B = {5,4,3,2,1}
print(A < B)

# а можно и так:
A >= B
A.issuperset(B)
```

```
True
True
False
False
```

Out[15]:
```
True
```

Множество vs Список: скорость поиска

Как время поиска элемента в списке и множестве зависит от их размера?

In [16]:
```
# создадим большой список и большое множество
from time import time
n = 2*10**7
print('Создаем список...')
a = [i**2 for i in range(n)]
print('Создаем множество...')
b = set(a)
k1 = (n//2)**2 # этот элемент есть в списке/множестве
k2 = k1 + 1     # такого элемента нет
print('Готово!')
```

```
Создаем список...
Создаем множество...
Готово!
```

In [17]:
```
# Поиск в списке
start = time()
res1 = k1 in a
finish = time()
print(f'Поиск в списке: ответ {res1} получен за {finish-start} сек')
start = time()
res2 = k2 in a
finish = time()
print(f'Поиск в списке: ответ {res2} получен за {finish-start} сек')
```

```
Поиск в списке: ответ True получен за 0.15868830680847168 сек
Поиск в списке: ответ False получен за 1.2608308792114258 сек
```

In [19]:
```
# Поиск в множестве
start = time()
res1 = k1 in b
finish = time()
print(f'Поиск в множестве: ответ {res1} получен за {finish-start} сек')
start = time()
res2 = k2 in b
finish = time()
print(f'Поиск в множестве: ответ {res2} получен за {finish-start} сек')
```

```
Поиск в множестве: ответ True получен за 8.654594421386719e-05 сек
Поиск в множестве: ответ False получен за 8.630752563476562e-05 сек
```

Как так?

Иллюстрация на примере простого кода:

```
s = {"яблоко", "банан", "вишня", "груша", "слива"}
print("банан" in s)
```

См. файл `Механика множества Python.pdf`

In [20]:
```
hash("яблоко")
```

Out[20]:
```
4747133628603962472
```

__ВЫВОД:__ Поиск в множестве выполняется за константное (ну почти) время

Задачи по теме "Множества"

pythontutor.ru

In [21]:
```
# Для организации входных данных
from random import randint
```

Задача 1. Количество различных чисел

Дан список чисел. Определите, сколько в нем встречается различных чисел.

Примечание. Эту задачу на Питоне можно решить в одну строчку.

In [24]:
```
# Дано
a = [randint(-5,5) for i in range(12)]
print(a)
# Решение
print(len(set(a)))
```

```
[0, -1, -2, -2, 3, -4, 0, -3, 2, -5, -2, 2]
8
```

Задача 2. Количество различных чисел

Даны два списка чисел. Посчитайте, сколько чисел содержится одновременно как в первом списке, так и во втором.

Примечание. Эту задачу на Питоне можно решить в одну строчку.

In [25]:
```
# Дано
a = [randint(-5,5) for i in range(12)]
print(a)
b = [randint(-5,5) for i in range(12)]
print(b)
# Решение
print(len(set(a) & set(b)))
```

```
[5, -3, 3, 4, -4, -4, 3, -3, 5, 0, -2, -2]
[-4, 0, 0, -1, -2, 3, -3, 4, -2, -4, 3, -1]
6
```

Задача 3. Пересечение множеств

Даны два списка чисел. Найдите все числа, которые входят как в первый, так и во второй список и выведите их в порядке возрастания.

Примечание. И даже эту задачу на Питоне можно решить в одну строчку.

In [26]:
```
# Дано
a = [randint(-5,5) for i in range(12)]
print(a)
b = [randint(-5,5) for i in range(12)]
print(b)
# Решение
print(sorted(set(a) & set(b)))
```

```
[3, -2, 3, -4, -4, -1, -2, 5, -4, -1, -5, -4]
[0, -3, 5, 5, 4, 2, 4, -3, 1, 4, 3, 1]
[3, 5]
```

Задачи с сайта



1. Один ряд клавиш. Дан список слов `words`. Выберите из него те слова, которые могут быть напечатаны, используя ровно один ряд русской клавиатуры (допускается также нажатие клавиш `Shift` или `CapsLock`)



In [27]:
```
def find_words(words):
    r1 = set('йцукенгшщзхъ')
    r2 = set('фывапролджэ')
    r3 = set('ячсмитьбю')
    res = []
    for w in words:
        s = set(w.lower())
        if s<=r1 or s<=r2 or s<=r3:
            res.append(w)
    return res
```

In [28]:
```
w1 = ['флора', 'форма', 'право', 'вектор',
      'Митя', 'Витя', 'Паша', 'Юля',
      'чит',  'бить', 'порка', 'цена',
      'фара', 'плод', 'жить', 'ворон']
```

In [29]:
```
find_words(w1)
```

Out[29]:
```
['флора', 'право', 'Митя', 'чит', 'бить', 'фара', 'плод']
```

2. Сумма уникальных элементов. Дан список целых чисел `nums`. Найдите сумму тех элементов списка, которые встречаются в нем только один раз

__Плохое правильное решение__

In [30]:
```
def sum_of_unique(nums):
    s = 0
    for x in nums:
        if nums.count(x)==1:
            s += x
    return s
```

In [31]:
```
assert sum_of_unique([1,5,3,5])==4, 'Ошибка тестирования 1'
assert sum_of_unique([1,1,1,1])==0, 'Ошибка тестирования 2'
assert sum_of_unique([1,2,3,4])==10, 'Ошибка тестирования 3'
```

In [32]:
```
import random
N = 10**4
random.seed(123456)
nums = [random.randint(-N,N) for i in range(N)]
```

In [33]:
```
from time import time
t = time()
assert sum_of_unique(nums)==191442, 'Ошибка тестирования 4'
dt = time()-t
print(dt)
assert dt<1, "Превышен допустимый интервал времени"
```

```
1.2173142433166504
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[33], line 6
      4 dt = time()-t
      5 print(dt)
----> 6 assert dt<1, "Превышен допустимый интервал времени"

AssertionError: Превышен допустимый интервал времени
```

__Хорошее правильное решение__

In [34]:
```
def sum_of_unique(nums):
    a = set() # все элементы
    b = set() # точно дубли
    for x in nums:
        if x in a:
            b.add(x)
        else:
            a.add(x)
    return sum(a-b)
```

In [35]:
```
assert sum_of_unique([1,2,3,2])==4, 'Ошибка тестирования 1'
assert sum_of_unique([1,1,1,1])==0, 'Ошибка тестирования 2'
assert sum_of_unique([1,2,3,4])==10, 'Ошибка тестирования 3'
```

In [36]:
```
t = time()
assert sum_of_unique(nums)==191442, 'Ошибка тестирования 4'
dt2 = time()-t
print(dt2)
assert dt2<1, "Превышен допустимый интервал времени"
```

```
0.0021245479583740234
```

In [37]:
```
# Во сколько раз ускорили работу?
dt/dt2
```

Out[37]:
```
572.9756480754124
```

In [ ]: