MAXIMUM
ОБРАЗОВАТЕЛЬНЫЕ ПРОГРАММЫ И ТЕХНОЛОГИИ

DS
программирование

Множества

План

План

Узнаем что такое множества и как добавлять и удалять данных из множества

План

Узнаем что такое множества и как добавлять и удалять данных из множества

Изучим ограничения, связанные с множествами

План

- Узнаем что такое множества и как добавлять и удалять данных из множества
- Изучим ограничения, связанные с множествами
- Узнаем об операциях: пересечение, объединение и вычитание множеств

План

- Узнаем что такое множества и как добавлять и удалять данных из множества
- Изучим ограничения, связанные с множествами
- Узнаем об операциях: пересечение, объединение и вычитание множеств
- Посмотрим на реальные кейсы применения множеств

Поехали!

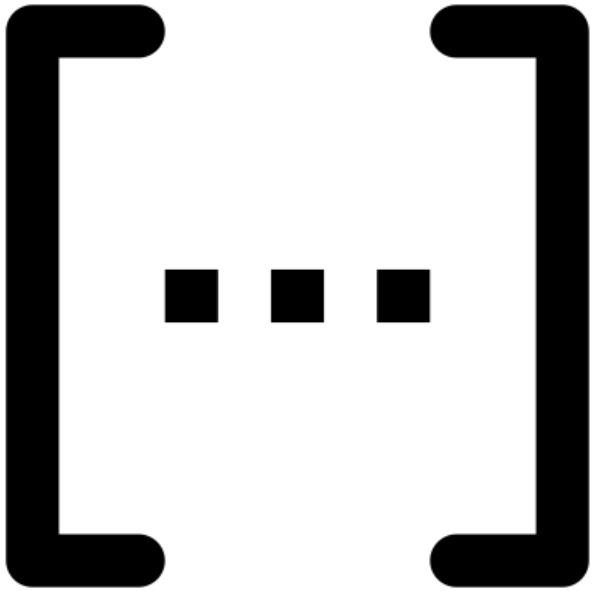
Что такое множество

Что такое множество

Что такое множество в математике?

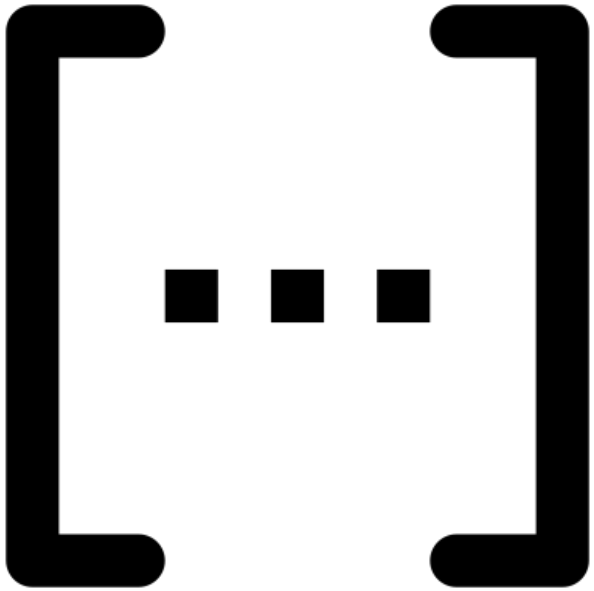
Что такое

Множество —



Что такое

Множество — одно из ключевых понятий математики, представляющее собой набор, совокупность каких-либо объектов — элементов этого множества.



Что такое

Натуральные числа								

Что такое

Натуральные числа	числа, которые мы используем при счете							

Что такое

Натуральные числа	числа, которые мы используем при счете					1		

Что такое

Натуральные числа	числа, которые мы используем при счете					1	2	

Что такое

Натуральные числа	числа, которые мы используем при счете					1	2	3

Что такое

Натуральные числа	числа, которые мы используем при счете					1	2	3
Целые числа	натуральные числа + 0 + отрицательные числа							

Что такое

Натуральные числа	числа, которые мы используем при счете					1	2	3
Целые числа	натуральные числа + 0 + отрицательные числа					1	2	3

Что такое

Натуральные числа	числа, которые мы используем при счете					1	2	3
Целые числа	натуральные числа + 0 + отрицательные числа			-1	0	1	2	3

Что такое

Натуральные числа	числа, которые мы используем при счете					1	2	3
Целые числа	натуральные числа + 0 + отрицательные числа		-2	-1	0	1	2	3

Что такое

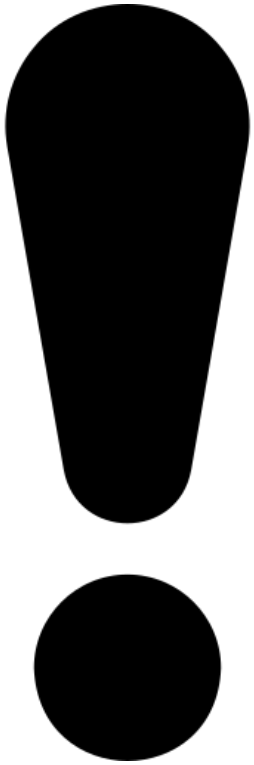
Натуральные числа	числа, которые мы используем при счете					1	2	3
Целые числа	натуральные числа + 0 + отрицательные числа	-3	-2	-1	0	1	2	3

Что такое

Натуральные числа	числа, которые мы используем при счете					1	2	3
Целые числа	натуральные числа + 0 + отрицательные числа	-3	-2	-1	0	1	2	3

Множество – это просто набор каких-то элементов!

Что такое



Элементы в множестве представлены **один раз**, **то есть не повторяются**

Что такое множество

Какие операции можно производить с множествами?

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B				
Объединение A и B				

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B		2		
Объединение A и B				

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B		2	3	
Объединение A и B				

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B		2	3	
Объединение A и B	1			

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B		2	3	
Объединение A и B	1	2		

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B		2	3	
Объединение A и B	1	2	3	

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B		2	3	
Объединение A и B	1	2	3	4

Что такое

Множество A	1	2	3	
Множество B		2	3	4
Пересечение A и B		2	3	
Объединение A и B	1	2	3	4

Элементы в множестве никак **не упорядочены**

Структура данных set

Структура данных set

```
my_set = set()
```

Структура данных set

```
my_set = set()  # создаем пустое множество
```

Структура данных set

```
my_set = set()  # создаем пустое множество
```

set()

«набор|множество»

Функция создает пустое множество

Структура данных set

```
my_set = set() # создаем пустое множество
my_set_2 = {1, 2, 3} # заполненное множество
```

Структура данных set

```
my_set = set() # создаем пустое множество
my_set_2 = {1, 2, 3} # заполненное множество
```

Обязательно перечисляем элементы множества.

Структура данных set

```
my_set = set() # создаем пустое множество
my_set_2 = {1, 2, 3} # заполненное множество
```

Обязательно перечисляем элементы множества.

Если оставим пустые фигурные скобки – получим словарь

Структура данных set

```
my_set = set() # создаем пустое множество
my_set_2 = {1, 2, 3} # заполненное множество
```

Обязательно перечисляем элементы множества.

Если оставим пустые фигурные скобки – получим словарь

как мы можем проверить тип данных?

Структура данных set

использовать функцию type()

```
my_set = set() # создаем пустое множество
my_set_2 = {1, 2, 3} # заполненное множество
```

Обязательно перечисляем элементы множества.

Если оставим пустые фигурные скобки – получим словарь

как мы можем проверить тип данных?

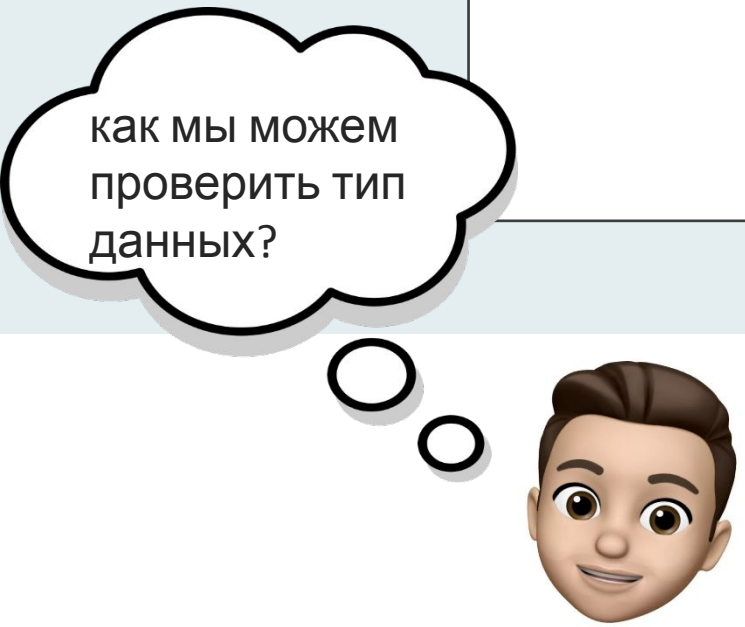
Структура данных set

использовать функцию type()

```
my_set = set() # создаем пустое множество
my_set_2 = {1, 2, 3} # заполненное множество
print(type(my_set_2))
```

Обязательно перечисляем элементы множества.

Если оставим пустые фигурные скобки – получим словарь



Структура данных set

использовать функцию type()

```
my_set = set() # создаем пустое множество
my_set_2 = {1, 2, 3} # заполненное множество
print(type(my_set_2)
)

>>> <class 'set'>
```

Обязательно перечисляем элементы множества.

Если оставим пустые фигурные скобки – получим словарь

как мы можем проверить тип данных?

Добавление и удаление данных из множества

Метод add()

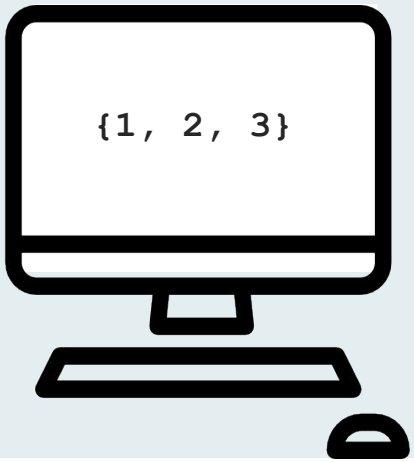
```
my_set.add(1)
my_set.add(2)
my_set.add(3)
print(my_set)
```

Метод add()

```
my_set.add(1)
my_set.add(2)
my_set.add(3)
print(my_set)
```

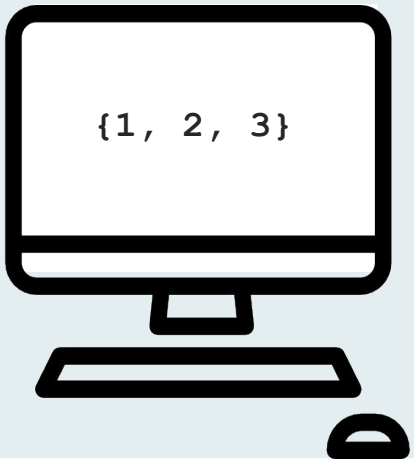
Метод add()

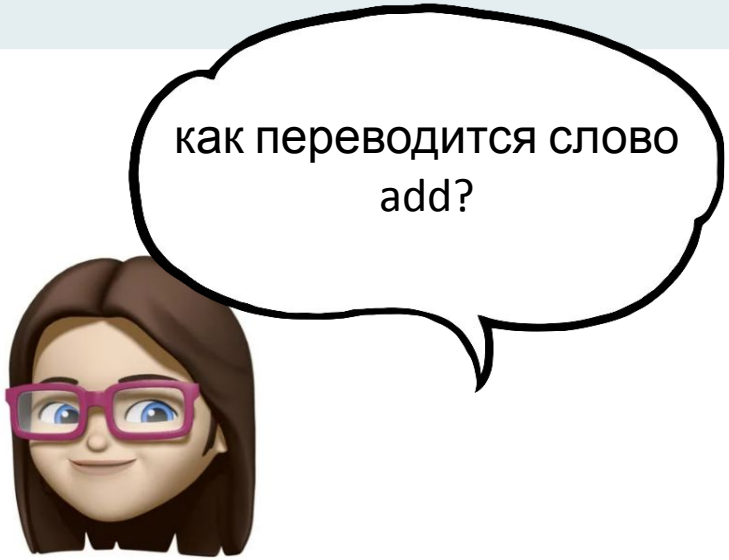
```
my_set.add(1)
my_set.add(2)
my_set.add(3)
print(my_set)
```



Метод add()

```
my_set.add(1)
my_set.add(2)
my_set.add(3)
print(my_set)
```

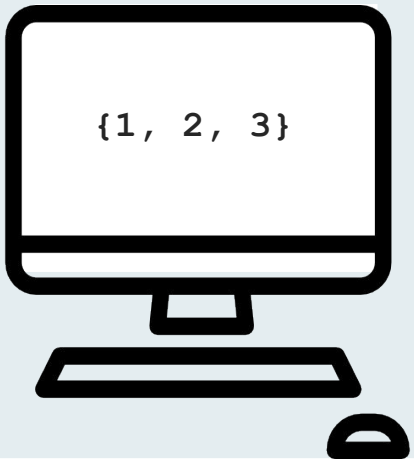


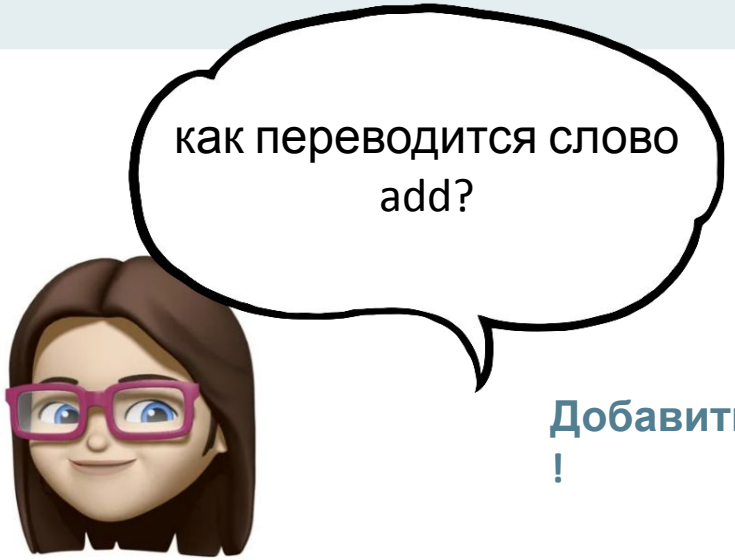


Добавление и удаление

Метод add()

```
my_set.add(1)
my_set.add(2)
my_set.add(3)
print(my_set)
```





Добавление и удаление

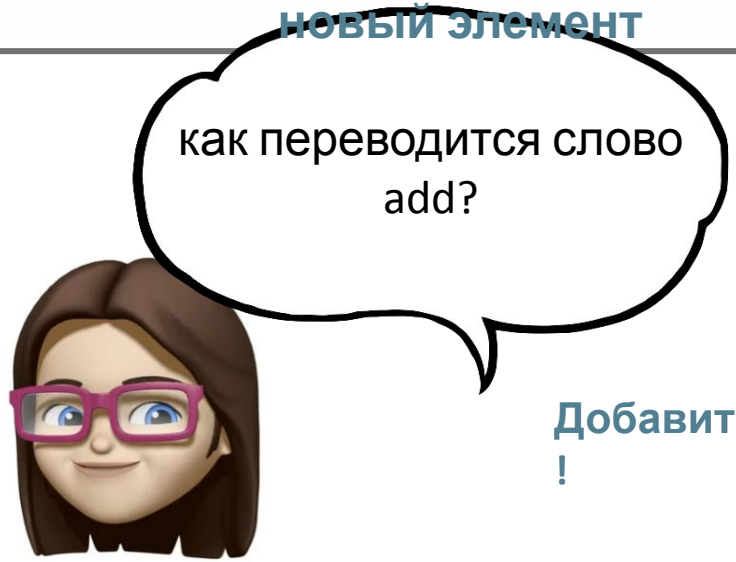
Метод add()

```
my_set.add(1)
my_set.add(2)
my_set.add(3)
print(my_set)
```

add()

метод, позволяющий добавить в множество новый элемент

```
{1, 2, 3}
```



Добавление и удаление

```
my_set.discard(1)
```

Добавление и удаление

```
my_set.discard(1)
```

discard()

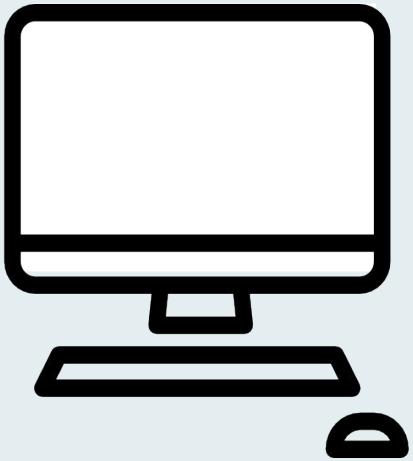
метод, позволяющий удалить из множества указанный элемент

Заполнение через цикл

```
for i in range(100):
    my_set.add(i)
```

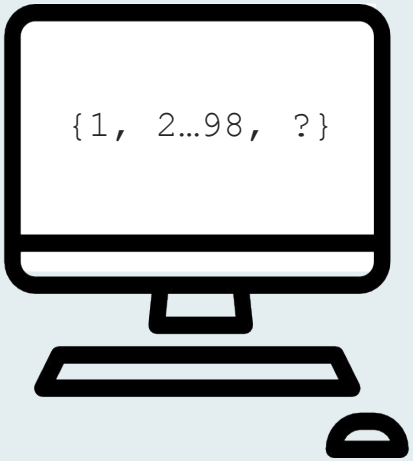
Заполнение через цикл

```
for i in range(100):
    my_set.add(i)
```



Заполнение через цикл

```
for i in range(100):
    my_set.add(i)
```



Заполнение через цикл

```
for i in range(100):
    my_set.add(i)
```

Добавление и удаление

```
my_set.clear()
```

Добавление и удаление

```
my_set.clear()
```

clear()

метод, позволяющий удалить все элементы из множества

Ограничения, связанные с множествами

Ограничения

Что мы не можем делать с множествами

Ограничения

Что мы не можем делать с множествами

Мы не можем обращаться к элементам по индексам.

Ограничения

Что мы не можем делать с множествами

Мы не можем обращаться к элементам по индексам.

```
my_set = {1, 2, 3}
```

Ограничения

Что мы не можем делать с множествами

Мы не можем обращаться к элементам по индексам.

```
my_set = {1, 2, 3}
print(my_set[0])
```

Ограничения

Что мы не можем делать с множествами

Мы не можем обращаться к элементам по индексам.

```
my_set = {1, 2, 3}
print(my_set[0])
```

TypeError: 'set' object is not subscriptable

Ограничения

Что мы не можем делать с множествами

Мы не можем обращаться к элементам по индексам.

В множестве могут храниться только неизменяемые типы данных

Ограничения

Что мы не можем делать с множествами

Мы не можем обращаться к элементам по индексам.

В множестве могут храниться только неизменяемые типы данных

```
my_set = {1, 2, 3}
my_set.add([4, 5])
```

Ограничения

Что мы не можем делать с множествами

Мы не можем обращаться к элементам по индексам.

В множестве могут храниться только неизменяемые типы данных

```
my_set = {1, 2, 3}
my_set.add([4, 5])
```

TypeError: unhashable type: 'list'

Ограничения

Что мы не можем делать с множествами

- Мы не можем обращаться к элементам по индексам.
- В множестве могут храниться только неизменяемые типы данных
- В множестве не могут храниться списки

Пересечение, объединение и вычитание множеств

```
my_grades = {3, 4, 5}
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}
```

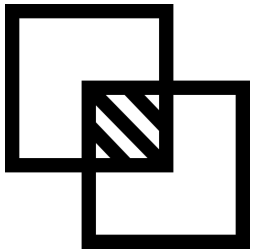
```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))

>>> {4, 5}
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))
print(your_grades.intersection(my_grades))
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))
print(your_grades.intersection(my_grades))

>>> {4, 5}
```

Пересечение, объединение и вычитание множеств

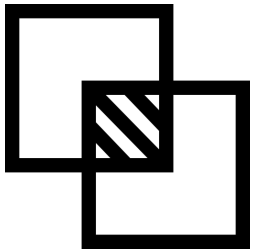


intersection() — метод, который принимает один параметр – другое множество, и находит пересечение этих множеств.

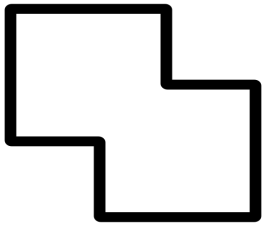
```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))
print(your_grades.intersection(my_grades))
print(my_grades.union(your_grades))
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))
print(your_grades.intersection(my_grades))
print(my_grades.union(your_grades))

>>> {2, 3, 4, 5}
```

Пересечение, объединение и вычитание множеств



intersection() — метод, который принимает один параметр – другое множество, и находит пересечение этих множеств.

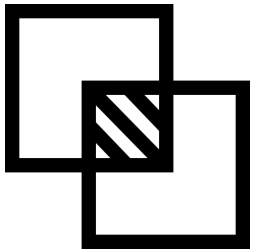


union() — метод, который принимает один параметр – другое множество, и объединяет элементы этих двух множеств.

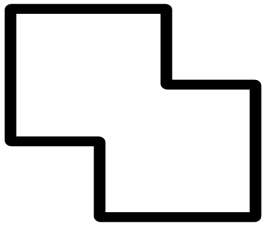
```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))
print(your_grades.intersection(my_grades))
print(my_grades.union(your_grades))
print(my_grades.difference(your_grades))
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


print(my_grades.intersection(your_grades))
print(your_grades.intersection(my_grades))
print(my_grades.union(your_grades))
print(my_grades.difference(your_grades))

>>> {3}
```

Пересечение, объединение и вычитание множеств



intersection() — метод, который принимает один параметр – другое множество, и находит пересечение этих множеств.



union() — метод, который принимает один параметр – другое множество, и объединяет элементы этих двух множеств.

difference() — метод, который принимает один параметр – другое множество, и вычитает элементы этих множеств.
В результирующем множестве окажутся элементы первого множества, которых изначально не было во втором

Пересечение, объединение и вычитание множеств

Эти операции не обновляют текущее множество, а создают новое. Если нам нужно именно обновить текущее множество, то для этого следует использовать те же методы, но с дополнением в конце **«_update»**

update() – метод, работающий как union(), только обновляет исходное множество

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)

>>> {4, 5}
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)

my_grades.update(your_grades)
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)

my_grades.update(your_grades)
print(my_grades)
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)

my_grades.update(your_grades)
print(my_grades)

>>> {2, 3, 4, 5}
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)

my_grades.update(your_grades)
print(my_grades)

my_grades.difference_update(your_grades)
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)

my_grades.update(your_grades)
print(my_grades)

my_grades.difference_update(your_grades)
print(my_grades)
```

```
my_grades = {3, 4, 5}
your_grades = {2, 4, 5}


my_grades.intersection_update(your_grades)
print(my_grades)

my_grades.update(your_grades)
print(my_grades)

my_grades.difference_update(your_grades)
print(my_grades)

>>> {3}
```

Кейсы применения множеств

Заполнение через цикл

```
my_list = [1, 2, 1, 3, 4, 2,
4]
```

Заполнение через цикл

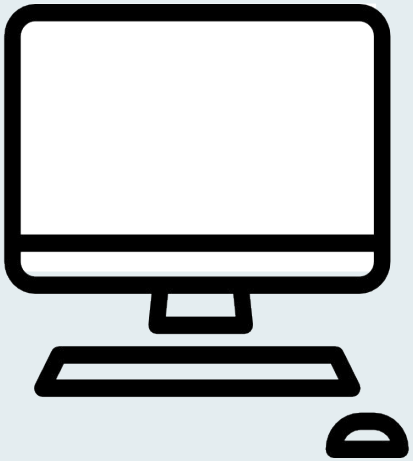
```
my_list = [1, 2, 1, 3, 4, 2,
4]

for i in set(my_list):
      print(i)
```

Заполнение через цикл

```
my_list = [1, 2, 1, 3, 4, 2,
4]

for i in set(my_list):
      print(i)
```

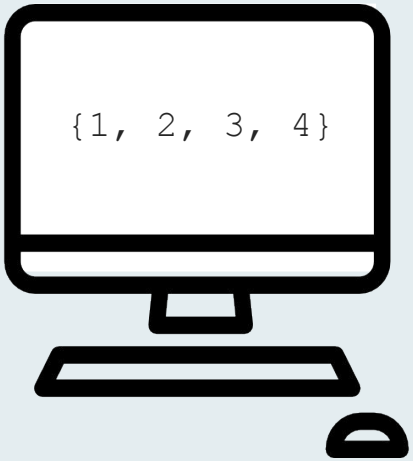
Заполнение через цикл

```
my_list = [1, 2, 1, 3, 4, 2,
4]

for i in set(my_list):
      print(i)
```



Заполнение через цикл

```
my_list = [1, 2, 1, 3, 4, 2,
4]

for i in set(my_list):
      print(i)
```



Является ли строка 'c a t s' подстрокой для 'I love cats'?

```
str1 = 'I love cats'
str2 = 'c a t s'
```

```
str1 = 'I love cats'
str2 = 'c a t s'
```

issubset()

Метод проверяющи
является ли текуще
множество
подмножеством дл
другого множества.

```
str1 = 'I love cats'
str2 = 'c a t s'
if set(str2).issubset(str1):
```

issubset()

Метод проверяющи
является ли текуще
множество
подмножеством дл
другого множества.

```
str1 = 'I love cats'
str2 = 'c a t s'
if set(str2).issubset(str1):
      print('Да, является')
```

issubset()

Метод проверяющи
является ли текуще
множество
подмножеством дл
другого множества.

```
str1 = 'I love cats'
str2 = 'c a t s'
if set(str2).issubset(str1):
      print('Да, является')
else:
      print('Не является')
```

issubset()

**Метод проверяющи
является ли текуще
множество
подмножеством дл
другого множества.**

```
str1 = 'I love cats'
str2 = 'c a t s'
if set(str2).issubset(str1):
      print('Да, является')
else:
      print('Не является')


>>> Да, является
```

issubset()

**Метод проверяющи
является ли текуще
множество
подмножеством дл
другого множества.**

Петя и Ваня играют в игру. Петя загадал натуральное число в **диапазоне от 1 до n** включительно. Ваня пытается угадать это число, для этого он называет некоторые множества натуральных чисел. Петя отвечает Ване YES, если среди названных чисел есть задуманное или NO в противном случае.

После нескольких заданных вопросов Ваня запутался в том, какие вопросы он задавал и какие ответы получил, в связи с чем просит вас помочь ему определить, какие числа мог загадать Петя.

Формат входных данных

Первая строка входных данных содержит число натуральное **n** — **наибольшее** число, которое мог загадать Петя. Далее идут строки, которые являются вариантами ответа Вани. Каждая строка представляет собой набор натуральных чисел, разделенных пробелами. После каждой строки с вопросом вводится с клавиатуры ответ Пети: YES или NO. Наконец, последняя строка входных данных содержит одно слово HELP.

Формат выходных данных

Вы должны вывести все числа, которые мог бы загадать Петя, исходя из указанных входных данных.

Пример входных данных:

10
1 2 3 4 5
YES
2 4
NO
HELP

Пример выходных данных:

1 3 5

```
n = int(input())  #считаем с клавиатуры число n
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
        for elem in ask.split():#убираем пробелы, перебираем элементы
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
        for elem in ask.split():#убираем пробелы, перебираем элементы
            ask_set.add(int(elem)) #преобразуем элемент в целое число
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
        for elem in ask.split():#убираем пробелы, перебираем элементы
            ask_set.add(int(elem)) #преобразуем элемент в целое число
    answer = input() #получаем ответ от Пети
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
        for elem in ask.split():#убираем пробелы, перебираем элементы
            ask_set.add(int(elem)) #преобразуем элемент в целое число
    answer = input() #получаем ответ от Пети
    if answer == 'YES': #если «Да»
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
        for elem in ask.split():#убираем пробелы, перебираем элементы
            ask_set.add(int(elem)) #преобразуем элемент в целое число
    answer = input() #получаем ответ от Пети
    if answer == 'YES': #если «Да»
        result.intersection_update(ask_set) #ищем пересечение
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
        for elem in ask.split():#убираем пробелы, перебираем элементы
            ask_set.add(int(elem)) #преобразуем элемент в целое число
    answer = input() #получаем ответ от Пети
    if answer == 'YES': #если «Да»
        result.intersection_update(ask_set) #ищем пересечение
    elif answer == 'NO': #если «Нет»
```

```
n = int(input()) #считаем с клавиатуры число n
result = set() #сформируем пустое множество
for i in range(1, n + 1): #цикл для прохода по числам от 1 до n
    result.add(i) #добавляем числа в множество
while True: #запускаем бесконечный цикл
    ask = input() #считаем с клавиатуры ответ Вани
    if ask == 'HELP': #проверка, нужна ли помощь Ване
        print(result) #если просит, выводим на экран результат
        break #выходим из цикла
    else: #если Ваня не просит о помощи, значит он ввел числа
        ask_set = set() #создадим новое пустое множество
        for elem in ask.split():#убираем пробелы, перебираем элементы
            ask_set.add(int(elem)) #преобразуем элемент в целое число
    answer = input() #получаем ответ от Пети
    if answer == 'YES': #если «Да»
        result.intersection_update(ask_set) #ищем пересечение
    elif answer == 'NO': #если «Нет»
        result.difference_update(ask_set) #ищем вычитание
```

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
Нет

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
Нет

2. Можно ли обратиться к элементу множества по индексу?

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
   Нет

2. Можно ли обратиться к элементу множества по индексу?
   Нет

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
   Нет

2. Можно ли обратиться к элементу множества по индексу?
   Нет

3. Можно ли добавить список в множество?

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
   Нет

2. Можно ли обратиться к элементу множества по индексу?
   Нет

3. Можно ли добавить список в множество?
   Нет

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
   Нет

2. Можно ли обратиться к элементу множества по индексу?
   Нет

3. Можно ли добавить список в множество?
   Нет

4. Какой метод помогает найти пересечение множеств?

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
   Нет

2. Можно ли обратиться к элементу множества по индексу?
   Нет

3. Можно ли добавить список в множество?
   Нет

4. Какой метод помогает найти пересечение множеств?
   intersection()

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
   Нет
2. Можно ли обратиться к элементу множества по индексу?
   Нет
3. Можно ли добавить список в множество?
   Нет
4. Какой метод помогает найти пересечение множеств?
   intersection()
5. Какой метод позволяет объединить два множества, изменив при этом исходное множество?

MAXIMUM ОБРАЗОВАТЕЛЬНЫЕ ПРОГРАММЫ И ТЕХНОЛОГИИ

Проверка понимания.

Вопросы:

1. Множество — это упорядоченная структура данных?
   Нет
2. Можно ли обратиться к элементу множества по индексу?
   Нет
3. Можно ли добавить список в множество?
   Нет
4. Какой метод помогает найти пересечение множеств?
   intersection()
5. Какой метод позволяет объединить два множества, изменив при этом исходное множество?
   update

MAXIMUM EDUCATION

DS
Программирование
Python

Спасибо за внимание!