

# Data structures in Python.

2	Data structures in Python.	2.1 Sets.
		2.2 Strings.
		2.3 Lists.
		2.4 Tuples.

# Data structures

- Data structures are “containers” that organize and group data according to type. Each of the data structures is unique in its own way.
- The data structures differ based on mutability and order.
- **Mutability** refers to the ability to change an object after its creation. Mutable objects can be modified, added, or deleted after they’ve been created, while immutable objects cannot be modified after their creation.
- **Order**, in this context, relates to whether the position of an element can be used to access the element.

# Sets

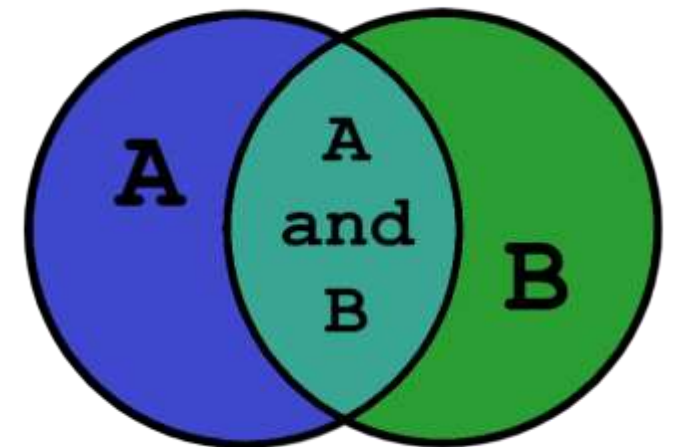
11.2.1.1 create sets;

11.2.1.2 use operations of adding, deleting, counting the number of elements;

11.2.1.3 apply operations to sets: union, intersection, subtraction, symmetric difference;

11.2.1.4 compare sets;

11.4.3.2 solve applied problems of various subject areas.



# Sets

A set is a complex data type that represents several values (elements of a set) under one name. A set is an unordered collection of unique items.

```
color_set = {"red", "yellow", "green"}  
print(color_set)
```

# Set's characteristics

- Sets are **unordered**.
- Set elements are **unique**. Duplicate elements are not allowed.
- A set itself may be modified, but the elements contained in the set must be of an **immutable** type.
- Sets are **unindexed**

A set is an **unordered** collection of unique items.

```
fruits = set() # empty set fruits
fruits = {"apple", "orange", "lemon", "banana"}
print(fruits) # output the set to the screen
```

What the program will output?

A set is an **unordered** collection of unique items.

$A = \{1, 2, 3\}$

$B = \{3, 2, 3, 1\}$

```
print(A == B)
```

What the program will output?

True



The set only stores **unique** elements, i.e. those that do not repeat others.

```
fruits = {"apple", "orange", "lemon", "orange",  
"banana", "orange", "apple"}  
  
print(fruits)
```

What the program will output?

The elements of a set can have **different** types.

```
fruits_and_price = {"apple", 300, "orange", 700, "lemon", 900}
```

Is it possible?

- *Set* in Python is a data structure equivalent to sets in mathematics. It may consist of various elements; the order of elements in a set is undefined. You can add and delete elements of a set, you can iterate the elements of the set, you can perform standard operations on sets (union, intersection, difference). Besides that, you can check if an element belongs to a set.
- Unlike arrays, where the elements are stored as ordered list, the order of elements in a set is undefined (moreover, the set elements are usually not stored in order of appearance in the set; this allows checking if an element belongs to a set faster than just going through all the elements of the set).
- Any immutable data type can be an element of a set: a number, a string, a tuple. Mutable (changeable) data types cannot be elements of the set. In particular, list cannot be an element of a set (but tuple can), and another set cannot be an element of a set. The requirement of immutability follows from the way how do computers represent sets in memory.

# Set operations

**len(set\_variable)** - Calculation of the number of elements in a set

```
fruits = {"apple", "orange", "lemon", "banana"}  
print(len(fruits))
```

```
fruits_and_price = {"apple", 300, "orange", 700,  
"lemon", 900, "banana", 550}  
print(len(fruits_and_price))
```

What the program will output?

4

8

```
for item in my_set:
```

Looping through the elements of the set

```
fruits = {"apple", "orange", "lemon", "banana"}  
for item in fruits:  
    print(item)
```

What the program will output?

Search for an item in a set  
if item **in** my\_set:

```
my_set = {"apple", "orange", "lemon", "banana"}  
item = "aple"
```

```
if item in my_set:  
    print('The item is in the set')  
else:  
    print('The item is not in set')
```

What the program will output?

The item is  
not in set

## Deleting an item

**discard** - removes the given element, if it is in the set, and does nothing if it is not;

**remove** - removes the given element, and throws a KeyError if not;

```
my_set = {'one', 'two', 'three'}
```

```
my_set.discard('two')
```

```
print(my_set)
```

```
my_set.discard('four')
```

```
my_set = {'one', 'two', 'three'}
```

```
my_set.remove('two')
```

```
print(my_set)
```

```
my_set.remove('four')
```

What the program will output? `{'one', 'three'}`  
`{'one', 'three'}` Error



Adding an element to the set

```
my_set.add(item)
```

```
fruits = {"apple", "orange", "lemon", "banana"}  
fruits.add("melon")  
fruits.add("lemon")  
print(len(fruits))
```

What the program will output?

Clear set

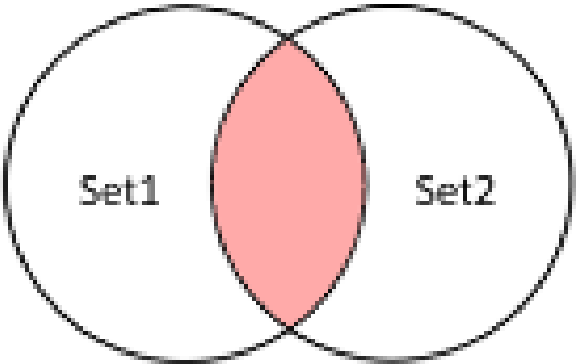
```
my_set.clear ()
```

```
my_set = {'one', 'two', 'three'}  
my_set.clear()  
print(my_set)
```

What the program will output?

set()

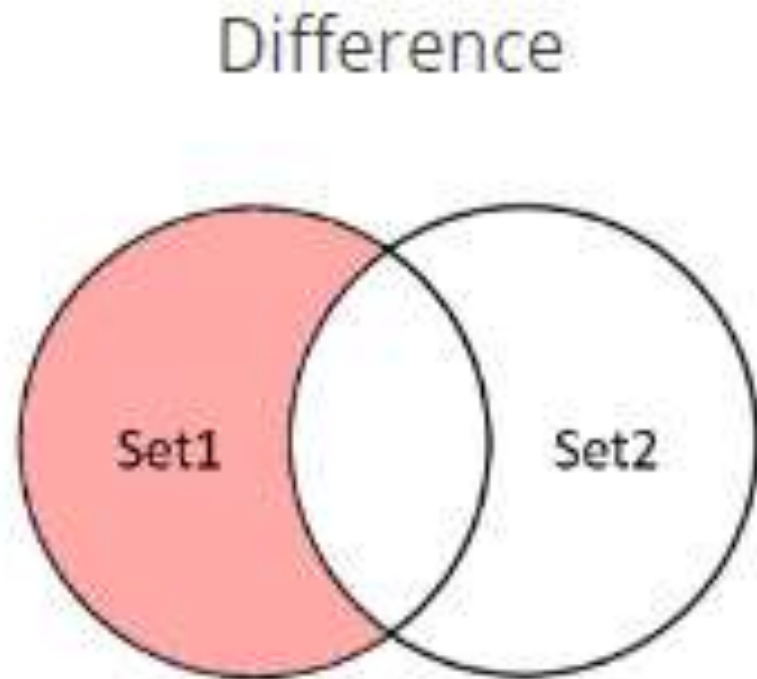
## Operations on two sets - Intersection

Operation	Usage example
<p data-bbox="359 496 639 544">Intersection</p>  <p data-bbox="282 803 372 839">Set1</p> <p data-bbox="619 803 708 839">Set2</p>	<pre data-bbox="1009 511 1768 882">set1 = {5, 8, 4, 6} set2 = {1, 2, 8, 5} new_set = set1 &amp; set2 print(new_set) new_set = set1.intersection(set2) print(new_set)</pre>

What the program will output?

{5, 8}

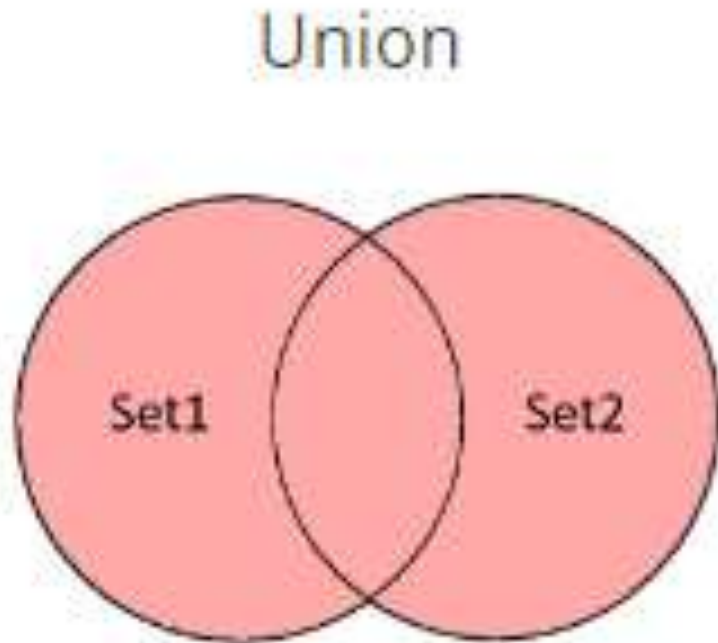
## Operations on two sets - **Difference**



```
set1 = {5, 8, 4, 6}
set2 = {1, 2, 8, 5}
new_set = set1 - set2
print(new_set)
new_set = set1.difference(set2)
print(new_set)
```

What the program will output? {4, 6}

## Operations on two sets - **Union**

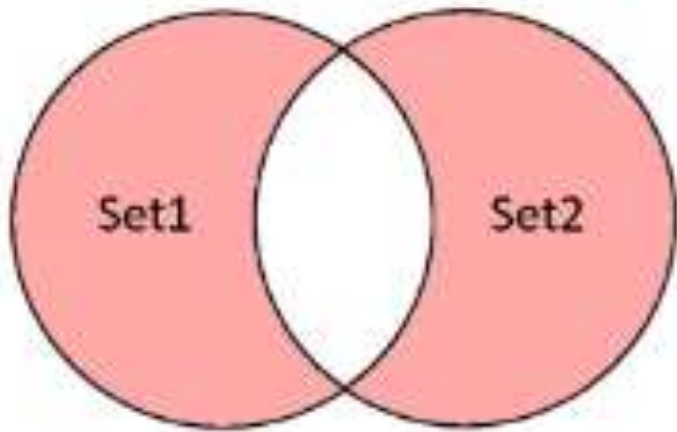


```
set1 = {5, 8, 4, 6}
set2 = {1, 2, 8, 5}
new_set = set1 | set2
print(new_set)
new_set = set1.union(set2)
print(new_set)
```

What the program will output? **{1, 2, 4, 5, 6, 8}**

## Operations on two sets - Symmetric Difference

Symmetric Difference



```
set1 = {5, 8, 4, 6}
```

```
set2 = {1, 2, 8, 5}
```

```
new_set = set1 ^ set2
```

```
print(new_set)
```

```
new_set = set1.symmetric_difference(set2)
```

```
print(new_set)
```

What the program will output? {1, 2, 4, 6}

# Operations on sets

This is how you perform the well-known [operations on sets](#) in Python:

<b>A   B</b> <b>A.union(B)</b>	Returns a set which is the union of sets <b>A</b> and <b>B</b> .
<b>A  = B</b> <b>A.update(B)</b>	Adds all elements of array <b>B</b> to the set <b>A</b> .
<b>A &amp; B</b> <b>A.intersection(B)</b>	Returns a set which is the intersection of sets <b>A</b> and <b>B</b> .
<b>A &amp;= B</b> <b>A.intersection_update(B)</b>	Leaves in the set <b>A</b> only items that belong to the set <b>B</b> .
<b>A - B</b> <b>A.difference(B)</b>	Returns the set difference of <b>A</b> and <b>B</b> (the elements included in <b>A</b> , but not included in <b>B</b> ).
<b>A -= B</b> <b>A.difference_update(B)</b>	Removes all elements of <b>B</b> from the set <b>A</b> .
<b>A ^ B</b> <b>A.symmetric_difference(B)</b>	Returns the symmetric difference of sets <b>A</b> and <b>B</b> (the elements belonging to either <b>A</b> or <b>B</b> , but not to both sets simultaneously).
<b>A ^= B</b> <b>A.symmetric_difference_update(B)</b>	Writes in <b>A</b> the symmetric difference of sets <b>A</b> and <b>B</b> .
<b>A &lt;= B</b> <b>A.issubset(B)</b>	Returns <b>true</b> if <b>A</b> is a subset of <b>B</b> .
<b>A &gt;= B</b> <b>A.issuperset(B)</b>	Returns <b>true</b> if <b>B</b> is a subset of <b>A</b> .
<b>A &lt; B</b>	Equivalent to <b>A &lt;= B and A != B</b>
<b>A &gt; B</b>	Equivalent to <b>A &gt;= B and A != B</b>

$A \cup B$ <code>A.union(B)</code>	Возвращает множество, являющееся объединением множеств <code>A</code> и <code>B</code> .
$A \cup= B$ <code>A.update(B)</code>	Добавляет в множество <code>A</code> все элементы из множества <code>B</code> .
$A \& B$ <code>A.intersection(B)</code>	Возвращает множество, являющееся пересечением множеств <code>A</code> и <code>B</code> .
$A \&= B$ <code>A.intersection_update(B)</code>	Оставляет в множестве <code>A</code> только те элементы, которые есть в множестве <code>B</code> .
$A - B$ <code>A.difference(B)</code>	Возвращает разность множеств <code>A</code> и <code>B</code> (элементы, входящие в <code>A</code> , но не входящие в <code>B</code> ).
$A -= B$ <code>A.difference_update(B)</code>	Удаляет из множества <code>A</code> все элементы, входящие в <code>B</code> .
$A \wedge B$ <code>A.symmetric_difference(B)</code>	Возвращает симметрическую разность множеств <code>A</code> и <code>B</code> (элементы, входящие в <code>A</code> или в <code>B</code> , но не в оба из них одновременно).
$A \wedge= B$ <code>A.symmetric_difference_update(B)</code>	Записывает в <code>A</code> симметрическую разность множеств <code>A</code> и <code>B</code> .
$A \leq B$ <code>A.issubset(B)</code>	Возвращает <code>true</code> , если <code>A</code> является подмножеством <code>B</code> .
$A \geq B$ <code>A.issuperset(B)</code>	Возвращает <code>true</code> , если <code>B</code> является подмножеством <code>A</code> .
$A < B$	Эквивалентно <code>A &lt;= B and A != B</code>
$A > B$	Эквивалентно <code>A &gt;= B and A != B</code>



## Comparison of sets

```
print({1, 2, 3, 4} == {4, 3, 1, 2})
```

True

```
print({1, 3, 2, 4} != {4, 3, 1, 2})
```

False

```
print({"one", "two", "three"} >= {"one", "three"})
```

True

False

```
print({2, 4} < {1, 3, 4})
```

False

```
print({"one", "more", "time"} < {"time", "one", "more"})
```

What the program will output?

## Subset and superset

The operation `s1 < s2` means "s1 is a **subset** of s2"  
"s2 is a **superset** of s1"

```
s1 = {'a', 'b', 'c'}
```

```
print(s1 <= s1)
```

```
s2 = {'a', 'b'}
```

```
print(s2 <= s1)
```

```
s3 = {'a'}
```

```
print(s3 <= s1)
```

```
s4 = {'a', 'z'}
```

```
print(s4 <= s1)
```

True

True

True

False

What the program will output?

# Python - Set Exercises

## Exercise:

Check if "apple" is present in the `fruits` set.

```
fruits = {"apple", "banana", "cherry"}  
if "apple" in fruits:  
    print("Yes, apple is a fruit!")
```

[Submit Answer »](#)

[https://www.w3schools.com/python/python\\_sets\\_exercises.asp](https://www.w3schools.com/python/python_sets_exercises.asp)

# Set Exercises

1. Write how to create an empty **flowers** set.

2. What will be displayed after executing the following program:

```
myset = {4, 5, 8, 5}
```

```
print (myset)
```

3. What will be displayed after executing the following program:

```
myset = {"to", "have", "to", 5, 8, 5}
```

```
print (len (myset))
```

# Exercises 2-4

Ex. 2 "Compare sets"



Compare sets

`{1, 2, 3, 4, 5, 6} == {6, 5, 3,`

`{2, 4, 6, 8, 10} != {10, 2, 6,`

`{'one', 'three'} <= {'one', 'two'`

`{1, 3, 5, 7, 9, 11} > {3, 5, 7, 11, 13}`

`{'let', 'it', 'be'} < {'be', 'it', 'let'}`

**Задание**

Define logic result of conditions.

OK

# Tasks

<https://stepik.org/lesson/449973/step/3?unit=440356>

**The problem - "How many elements?"**

Difficulty: \*

Create a set with the following values: carrot, potato, tomato, onion and display the number of elements in this set.

[Help: Sets](#)

**Output:**

Integer - the number of elements in the set.

---

**Sample Input:**

---

**Sample Output:**

4

### The task - "Add and count"

Difficulty: \*\*

Write a program that adds n elements to a set and prints out how many elements are in the set.

[Help: Sets](#)

#### Input data:

The first line contains an integer n.

On the next lines, there are n elements to add to the set.

#### Output:

Integer - the number of elements in the set.

---

#### Sample Input:

```
4
carrot
onion
potato
tomato
```

---

#### Sample Output:

```
4
```

**The task "The sum of the elements of a set"**

Difficulty: \*\*

Write a program that adds n integers to a set and outputs the sum of the elements in the set.

[Help: Sets](#)

**Input data:**

The first line contains an integer n.

The next lines contain n integers to add to the set.

**Output:**

The sum of the elements of the set.

---

**Sample Input:**

```
3
10
11
12
```

---

**Sample Output:**

```
33
```



### Task "Sports sections"

Difficulty: \*\*

In a specialized sports school, each student in the class plays either basketball, volleyball, or both.

The class teacher has a list of students involved in each sport. Write a program that allows the class teacher to quickly figure out how many students are in only one sport.

[Help: Sets](#)

#### Input data:

The first two lines indicate the number of students playing basketball and volleyball (**B** and **V**). Then there are **B** lines - the names of the students who play basketball; and **V** lines with the names of the students involved in the volleyball section. It is guaranteed that there are no namesakes among the students.

#### Output:

The number of students who practice only one sport. If there are no such ones, you need to output "there are none".

---

#### Sample Input:

```
2
3
Kataev
Romanov
Valiev
Sartaev
Romanov
```

#### Sample Output:

```
3
```

Активация Windows  
Чтобы активировать Windows, перейдите в раздел "Параметры".

### Task "Summer Literature"

Difficulty: \*\*\*

Arseny received a list of summer references at the end of the school year. Now he needs to find out which books from this list he has and which he does not. Fortunately, Alexey has a text document on his computer that contains all the books from his home library in random order. Determine which books are from the list for the summer Alexei has and which ones are not.

[Help: Sets](#)

#### Input data:

The first line contains an integer  $n$  - the number of books in the home library.

The second line contains the number  $m$  - the number of books on the list for the summer. The home library and the list of books contain at least one book ( $n \geq 1$  and  $m \geq 1$ ).

Then there are  $n$  lines with the titles of books from the home library and  $m$  lines of titles from the summer list. It is guaranteed that all words in the book titles are separated by one space, and after the last word, there is a line feed immediately (that is, there are no "invisible" spaces).

#### Output:

Output data:  $m$  lines containing the word "YES" if the book is found in the library and "NO" if not.

---

#### Sample Input:

```
4
2
The hobbit
Alice in Wonderland
Tom Sawyer
Treasure Island
Tom Sawyer
Lord of the Rings
```

#### Sample Output:

```
YES
NO
```

Активация Winc  
Чтобы активировать  
раздел "Параметры".

# Tasks

1. Given a list of integers. Determine how many distinct numbers there are.
2. Given two lists of numbers. Count how many unique numbers occur in both of them.
3. Given two lists of numbers. Find all the numbers that occur in both the first and the second list and print them in ascending order.
4. Given a sequence of numbers, determine if the next number has already been encountered. For each number, print the word **YES** (in a separate line) if this number has already been encountered, and print **NO**, if it has not already been encountered.

# Task Polyglots

## ***Statement***

Each student at a certain school speaks a number of languages. We need to determine which languages are spoken by all the students, which languages are spoken by at least one student.

Given, the number of students, and then for each student given the number of languages they speak followed by the name of each language spoken, find and print the number of languages spoken by all the students, followed by a list the languages by name, then print the number of languages spoken by at least one student, followed by the list of the languages by name. Print the languages in alphabetical order.

# ЗАДАЧИ

1. Дан список чисел. Определите, сколько в нем встречается различных чисел.
2. Даны два списка чисел. Посчитайте, сколько чисел содержится одновременно как в первом списке, так и во втором.
3. Даны два списка чисел. Найдите все числа, которые входят как в первый, так и во второй список и выведите их в порядке возрастания.
4. Во входной строке записана последовательность чисел через пробел. Для каждого числа выведите слово YES (в отдельной строке), если это число ранее встречалось в последовательности или NO, если не встречалось.

# Python Sets Quiz

<https://realpython.com/quizzes/python-sets/>