Операции над множествами, сравнение множеств

На этом занятии мы с вами будем говорить об операциях над ними – это:

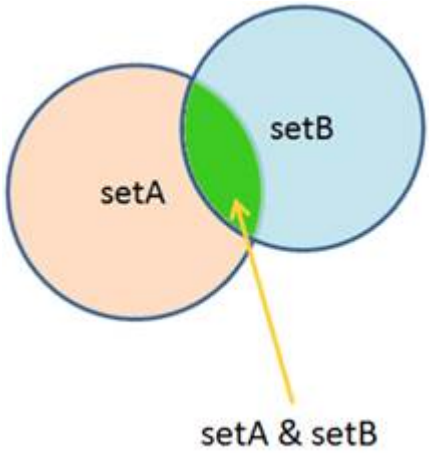
- пересечение множеств;
- объединение множеств;
- вычитание множеств;
- вычисление симметричной разности.

Для простоты зададим два множества с числами:

```
setA = {1, 2, 3, 4}
setB = {3, 4, 5, 6, 7}
```

На их основе можно построить третье множество, как результат пересечения этих двух: `setA & setB`

На выходе получаем новое множество, состоящее из значений, общих для множеств setA и setB. Это и есть операция пересечения двух множеств. При этом, исходные множества остаются без изменений. Здесь именно создается новое множество с результатом выполнения этой операции.



Разумеется, чтобы в дальнейшем в программе работать с полученным результатом, его нужно сохранить через какую-либо переменную, например, так: `res = setA & setB`

Также допустима запись в виде: `setA &= setB`

это будет эквивалент строчки: `setA = setA & setB`

То есть, мы вычисляем результат пересечения и сохраняем его в переменной seta и, как бы, меняем само множество setA.

А вот если взять множество:

```
setC = {9, 10, 11}
```

которое не имеет общих значений с другим пересекающимся множеством, то результатом: `setA & setC`

будет пустое множество.

Обычно, на практике используют оператор пересечения &, но вместо него можно использовать специальный метод:

```
setA = {1, 2, 3, 4}
setB = {3, 4, 5, 6, 7}
setA.intersection(setB)
```

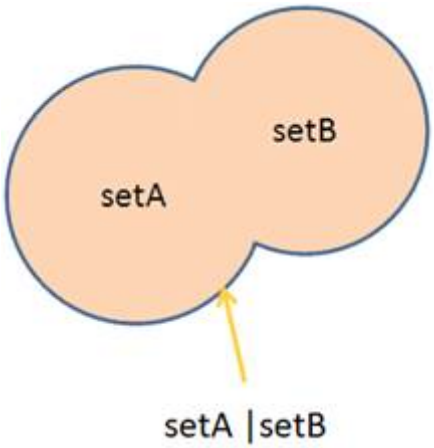
Результат будет тем же, на выходе получим третье множество, состоящее из общих значений множеств setA и setB. Если же мы хотим выполнить аналог операции: `setA &= setB`

то для этого следует использовать другой метод: `setA.intersection_update(setB)`

В результате, меняется само множество setA, в котором хранятся значения пересечения двух множеств.

Следующая операция – это объединение двух множеств. Она записывается, так:

```
setA = {1, 2, 3, 4}
setB = {3, 4, 5, 6, 7}
setA | setB
```



На выходе также получаем новое множество, состоящее из всех значений обоих множеств, разумеется, без их дублирования.

Или же можно воспользоваться эквивалентным методом: `setA.union(setB)`

который возвращает множество из объединенных значений. При этом, сами множества остаются без изменений.

Операцию объединения можно записать также в виде: `setA |= setB`

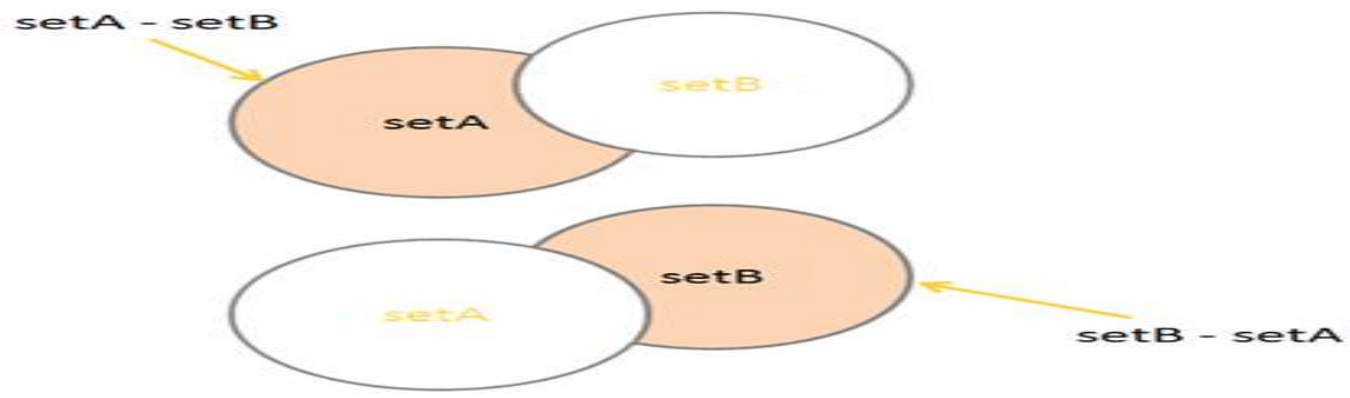
Тогда уже само множество setA будет хранить результат объединения двух множеств, то есть, оно изменится. Множество setB останется без изменений.Следующая операция – это **вычитание** множеств. Например, для множеств:

```
setA = {1,2,3,4}
setB = {3,4,5,6,7}
```

операция `setA - setB`

возвратит новое множество, в котором из множества setA будут удалены все значения, повторяющиеся в множестве setB:

{1, 2}



Или, наоборот, если из множества setB вычесть множество setA: `setB - setA`

то получим значения из которых исключены величины, входящие в множество setA.

Также здесь можно выполнять эквивалентные операции:

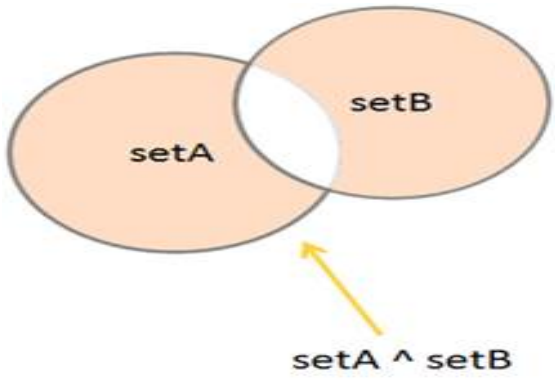
```
setA -= setB
setB -= setA
```

В этом случае переменные setA и setB будут ссылаться на соответствующие результаты вычитаний.

Наконец, последняя операции над множествами – симметричная разность, возвращает множество, состоящее из неповторяющихся значений обоих множеств. Реализуется она следующим образом:

```
setA = {1,2,3,4}
setB = {3,4,5,6,7}
setA ^ setB
```

На выходе получим третье, новое множество, состоящее из значений, не входящих одновременно в оба множества.



В заключение этого занятия я хочу показать вам, как можно сравнивать множества между собой.

Предположим, имеются два таких множества:

```
setA = {7, 6, 5, 4, 3}
setB = {3, 4, 5, 6, 7}
```

И мы хотим узнать, равны они или нет. Для этого, используется уже знакомый нам оператор сравнения на равенство: `setA == setB`

Увидим значение True (истина). Почему? Дело в том, что множества считаются равными, если все элементы, входящие в одно из них, также принадлежат другому и мощности этих множеств равны (то есть они содержат одинаковое число элементов). Наши множества setA и setB удовлетворяют этим условиям.

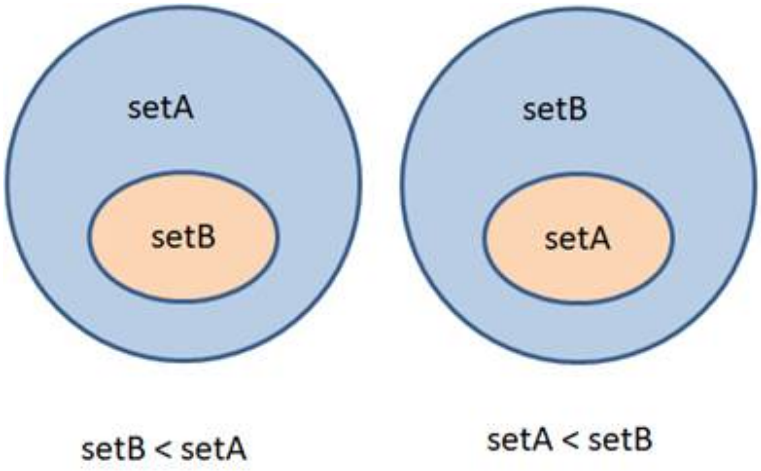
Соответственно, противоположное сравнение на не равенство, выполняется, следующим образом: `setA != setB`

Следующие два оператора сравнения на больше и меньше, фактически, определяют, входит ли одно множество в другое или нет.

Например, возьмем множества

```
setA = {7, 6, 5, 4, 3}
setB = {3, 4, 5}
```

тогда операция `setB < setA` вернет True, а операция `setB > setA` значение False. Но, если хотя бы один элемент множества setB не будет принадлежать множеству setA:



setB < setA setA < setB

`setB.add(22)` то обе операции вернут False. Для равных множеств

```
setA = {7, 6, 5, 4, 3}
setB = {3, 4, 5, 6, 7}
```

обе операции также вернут False. Но вот такие операторы:

```
setA <= setB
setA >= setB
```

вернут True.