МНОЖЕСТВО В PYTHON

ВВЕДЕНИЕ

Множества в Python – это структура данных, которые содержат неупорядоченные элементы. Элементы также не является индексированным. Как и список, множество позволяет внесение и удаление элементов. Однако, есть ряд особенных характеристик, которые определяют и отделяют множество от других структур данных:

- Множество не содержит дубликаты элементов;
- Элементы множества являются неизменными (их нельзя менять), однако само по себе множество является изменяемым, и его можно менять;
- Так как элементы не индексируются, множества не поддерживают никаких операций **среза** и **индексирования**.

СОЗДАНИЕ МНОЖЕСТВ

Мы можем создать множество путем передачи всех элементов множества внутри фигурных скобок {} и разделить элементы при помощи запятых (,). Множество может содержать любое количество элементов и элементы могут быть разных типов, к примеру, целые числа, строки, кортежи, и т. д. Однако, множество не поддерживает изменяемые элементы, такие как списки, словари, и так далее.

- A=set()
- A=set('foxford')

>> {'d', 'f', 'o', 'r', 'x'}

- A = {4, 5, 6}
- A = {} – не использовать

ДОСТУП К ЭЛЕМЕНТАМ МНОЖЕСТВ

Python не предоставляет прямой способ получения значения к отдельным **элементам множества**. Однако, мы можем использовать цикл для итерации через все элементы множества. Например:

```
months = set(["Jan", "Feb", "March", "Apr", "May",
"June", "July", "Aug", "Sep", "Oct", "Nov", "Dec"])

for m in months:
    print(m)
```

ДОБАВЛЕНИЕ ЭЛЕМЕНТОВ ВО МНОЖЕСТВО

Python позволяет нам вносить новые элементы во множество при помощи функции **add().** Например:

```
months = set(["Jan", "March", "Apr", "May", "June", "July", "Aug", "Sep", "Oct", "Nov", "Dec"])
months.add("Feb")
print(months)
```

Или

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

УДАЛЕНИЕ ЭЛЕМЕНТА ИЗ МНОЖЕСТВ

Python позволяет нам удалять элемент из множества, но не используя индекс, так как множество элементов не индексированы. Элементы могут быть удалены при помощи обоих методов discard() и remove().

Помните, что метод discard() не будет выдавать ошибку, если элемент не был найден во множестве. Однако, если метод remove() используется и элемент не был найден, возникнет ошибка.

Продемонстрируем как удалять элемент при помощи метода discard():

```
num_set = {1, 2, 3, 4, 5, 6}
num_set.discard(3)
print(num_set)
```

Результат:

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

УДАЛЕНИЕ ЭЛЕМЕНТА ИЗ МНОЖЕСТВ

С методом pop(), мы можем удалить и вернуть элемент. Так как элементы находятся в произвольном порядке, мы не можем утверждать или предсказать, какой элемент будет удален.

Например:

```
num_set = {1, 2, 3, 4, 5, 6}
print(num_set.pop())
```

Результат:

```
1
```

Вы можете использовать тот же метод при удалении элемента и возврате элементов, которые остаются во множестве. Например:

```
num_set = {1, 2, 3, 4, 5, 6}
num_set.pop()
print(num_set)
```

Результат:

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

ФУНКЦИИ ДЛЯ РАБОТЫ СО МНОЖЕСТВАМИ

- **len(set)**: возвращает количество элементов множества
- **min(set)**: возвращает наименьший элемент множества
- **max(set)**: возвращает наибольший элемент множества
- **x in set**: проверяет принадлежность множеству
- **set.clear()**: очищает множество

ОБЪЕДИНЕНИЕ МНОЖЕСТВ

Предположим, у нас есть два множества, **A** и **B**. Объединение этих двух множеств — это множество со всеми элементами обеих множеств. Такая операция выполняется при помощи функции Python под названием union().

Рассмотрим пример:

```
months_a = set(["Jan", "Feb", "March", "Apr", "May", "June"])
months_b = set(["July", "Aug", "Sep", "Oct", "Nov", "Dec"])
all_months = months_a.union(months_b)
print(all_months)
```

Результат:

```
{'Oct', 'Jan', 'Nov', 'May', 'Aug', 'Feb', 'Sep', 'March', 'Apr', 'Dec', 'June',  'July'}
```

ОБЪЕДИНЕНИЕ МНОЖЕСТВ

При выполнении операции объединения, дубликаты игнорируются, так что только один из двух элементов дубликатов будет отображаться.
Например:

```
x = {1, 2, 3}
y = {4, 3, 6}
z = {7, 4, 9}
output = x.union(y, z)
print(output)
```

Результат:

{1, 2, 3, 4, 6, 7, 9}

ПЕРЕСЕЧЕНИЕ МНОЖЕСТВ

Предположим, у вас есть два множества: **A** и **B**. Их пересечение представляет собой множество элементов, которые являются общими для **A** и для **B**.

Операция пересечения во множествах может быть достигнута как при помощи оператора &, так и метода intersection(). Рассмотрим пример:

```
x = {1, 2, 3}
y = {4, 3, 6}
print(x & y) # Результат: 3
```

В обеих множествах 3 является **общим элементом**. То же самое может быть достигнуто при использовании метода intersection():

```
x = {1, 2, 3}
y = {4, 3, 6}
z = x.intersection(y)
print(z)    # Результат: 3
```

РАЗНИЦА МЕЖДУ МНОЖЕСТВАМИ

Предположим, у вас есть два множества: **A** и **B**. Разница между A и B (A — B) — это множество со всеми элементами, которые содержатся в A, но не в B. Соответственно, (B — A) — это множество со всеми элементами в B, но не в A.

Для определения разницы между множествами в Python, мы можем использовать как функцию difference(), так и оператор — . Рассмотрим пример:

```
set_a = {1, 2, 3, 4, 5}
set_b = {4, 5, 6, 7, 8}
diff_set = set_a.difference(set_b)
print(diff_set)
```

Результат:

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

СИММЕТРИЧНАЯ РАЗНИЦА

Симметричная разница между множествами A и B — это множество с элементами, которые находятся в **A** и **B**, за исключением тех элементов, которые **являются общими** для обеих множеств. Это определяется использованием метода Python под названием symmetric_difference(), или оператора ^. Посмотрим на пример:

```
set_a = {1, 2, 3, 4, 5}
set_b = {4, 5, 6, 7, 8}
symm_diff = set_a.symmetric_difference(set_b)
print(symm_diff)
```

Результат:

```
{1, 2, 3, 6, 7, 8}
```

МНОЖЕСТВА В PYTHON

A | B (A.union(B)). Возвращает множество, являющееся объединением множеств A и B.

A & B (A.intersection(B)). Возвращает множество, являющееся
пересечением множеств A и B.

A - B (A.difference(B)). Возвращает разность множеств A и B.

A ^ B (A.symmetric_difference(B)). Возвращает симметрическую
разность множеств A и B.

A < B Возвращает true, если A является подмножеством B и не равно B.
A > B Возвращает true, если A является подмножеством B и не равно B.

МНОЖЕСТВА В PYTHON

A |= B (A.update(B)). Добавляет в множество A все элементы множества B.
A &= B (A.intersection_update(B)). Оставляет в множестве A только элементы множества B.
A -= B (A.difference_update(B)). Удаляет из множества A все элементы, входящие в B.
A ^= B (A.symmetric_difference_update(B)). Записывает в A симметрическую разность множеств A и B.
A <= B (A.issubset(B)). Возвращает true, если A является подмножеством B.
A>=B (A.issuperset(B)). Возвращает true, если A является подмножеством B.

МЕТОДЫ МНОЖЕСТВ

Метод copy()

Этот метод возвращает копию множества. Например:

```
string_set = {"Nicholas", "Michelle", "John", "Mercy"}
x = string_set.copy()
print(x)
```

Результат:

```
{'John', 'Michelle', 'Nicholas', 'Mercy'}
```

МЕТОДЫ МНОЖЕСТВ

Метод isdisjoint()

Этот метод проверяет, является ли множество пересечением или нет. Если множества **не содержат общих элементов**, метод возвращает True, в противном случае — False. Например:

```
names_a = {"Nicholas", "Michelle", "John", "Mercy"}
names_b = {"Jeff", "Bosco", "Teddy", "Milly"}
x = names_a.isdisjoint(names_b)
print(x)
```

Результат:

```
True
```

Оба множества **не имеют общих элементов**, что делает выдачу True.

ЗАДАЧА №1

Дано два списка.
Узнать:
1. количество уникальных элементов списка;
2. максимальный из них;
3. минимальный из них ;
4. вывести общие элементы для двух списков

РЕШЕНИЕ

```
A = set(input().split())
B = set(input().split())
print(len(A), len(B))
print(max(max(A),max(B)))
print(min(min(A),min(B)))
print(A.intersection(B))
```

ЗАДАЧА №2

В строке записан текст. Словом считается последовательность непробельных символов, идущих подряд. Слова разделены одним или большим числом пробелов или символами конца строки.
Определите, сколько различных слов содержится в тексте

РЕШЕНИЕ

```
a=set()
f = open('test.txt')
for w in f.read().split():
a.add(w)
print(len(a))
```

Или:

```
print(len(set(w for w in open('test.txt').read().split())))
```