Множества в Python

Множества (set) это неупорядоченный набор уникальных элементов.
Элементами множества могут быть числа, строки, кортежи, логические константы, но не могут быть списки.

Примеры множеств

```
a = {1, 2, 4, 89, -56, 0, -34}
b = {'one', 'two', 3, 4, 'five', (5, 7, 8), True}
```

Способы создания множества

1. Создание пустого множества `b = set()`
2. Создание множества на основе итерируемого (перечисляемого) объекта

`sl = 'ABCDABACABA'` `b = set(sl)`	`b={'A', 'B', 'C', 'D'}`	Каждый символ – отдельный элемент списка, повторяющихся символов нет
`a = [1, 2, 2, 3, 1]` `b = set(a)`	`b = {1, 2, 3}`	Каждый уникальный элемент списка становится элементом множества.

3. Создание множества с помощью генератора

```
b = set([x % 10 for x in range(1, 100) if x % 2 == 0])
----------------
b = {8, 0, 2, 4, 6}
```

4. Добавление элемента в сет с помощью метода add

```
b = set()
b.add(3)
b.add(2)
b.add(3)
----------
b = {3, 2}
```

Функции и методы множеств

`s = len(b)`	Находит длину множества
`x = min(b)` `x = max(b)`	Находит минимальный (максимальный) элемент множества, если элементы списка сравнимые элементы
`x = sum(b)`	Находит сумму элементов множества, если все они числа
`a = sorted(b)`	Создание нового списка a, в котором элементы множества b будут отсортированы по возрастанию
`x in b`	Проверка вхождения элемента x во множество b
`b.add(x)`	Добавление элемента x во множество b
`b.remove(x)`	Удаление элемента x из множества b
`print(*b)`	Вывод элементов множества через пробел в одну строку

Просмотр элементов множества

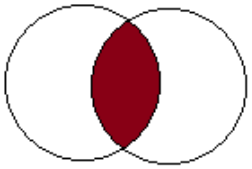
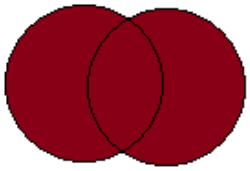
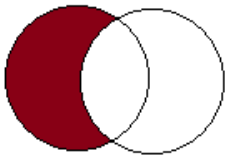
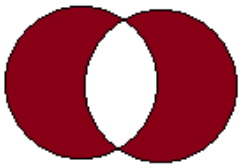
```
for x in b:
    print(x, end = ' ')
```

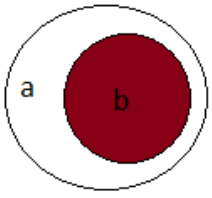
В отличие от списков по множествам нельзя пройтись, перебирая индексы элементов, т.к. множество – это неупорядоченный набор данных.

Множества в Python. Операции над множествами

Над множествами в Python можно выполнять операции аналогичные операциями над множествами в математике.
Пусть заданы два множества:

```
a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7}
```

Логическая операция	Запись на Python	Результат выполнения	Математическое представление
Пересечение множеств: войдут только элементы, которые есть в обоих множествах	`c = a & b`	`c = {4, 5}`	
Объединение множеств: войдут элементы, которые есть хотя бы в одном множестве	`c = a | b`	`c={1, 2, 3, 4, 5, 6, 7}`	
Разность множеств: из одного множества удаляют все элементы, которые есть и во втором множестве	`c = a - b`	`c = {1, 2, 3}`	
Симметричная разность множеств: в новое множество войду элементы, которые принадлежат одному из множеств, но не обоим одновременно.	`c = a ^ b`	`c = {1, 2, 3, 6, 7}`	

Сравнение множеств: Одно множество меньше другого, когда оно является подмножеством другого. Множества равны, если все их элементы одинаковые.	`a < b`
	`a <= b`
	`a != b`
	`a == b`
		`{1, 2, 3} <={0, 1, 2, 3, 4}`
		`{1, 2, 3} != {1, 2, 4}`
		`{1, 2, 3} == {3, 2, 1}`
			 b < a

Примеры операций над множествами:

```
a = set('elephant')
b = set('telephone')
a & b = {'p', 'n', 'h', 'e', 'l', 't'}
a | b = {'p', 'n', 'e', 'h', 'l', 'a', 't', 'o'}
a ^ b = {'o', 'a'}
a - b = {'a'}
b - a = {'o'}
a > {'e', 'l', 't'} == True
```

Примеры решения задач с помощью множеств

№1. Создайте множество, состоящее из заглавных букв английского алфавита.

Решение:

Для решения задачи воспользуемся генератором. Коды символов английского алфавита идут подряд с 65 по 90. Переберем с помощью range все числа от 65 до 90 и к каждому применим функцию chr(x) – преобразования кода в символ.

Код решения:

```
b = set([chr(x) for x in range(65, 91)])
print(*b)
```

Подумайте, что нужно добавить в код программы, чтобы символы выводились в алфавитном порядке.

№2. На вход программе подается строка. Выведите все цифры, встречающиеся в строке в том порядке, как они идут в заданной строке.

Входные данные	Выходные значения
As234G78k94	2347894

Решение:

С помощью генератора создадим множество, состоящее из всех цифр. Пройдем по символам строки и проверим, принадлежит ли символ множеству цифр, если да, то выедем его на экран.

Код решения:

```
sl = input()
b = {str(x) for x in range(0, 10)}
for x in sl:
    if x in b:
        print(x, end = '')
```

№3. На вход программе подается символьная строка. Определите, сколько различных цифр встречается в строке.

Входные данные	Выходные значения
As234G78k94	6
11A1A!s!***1	1

Решение:

Эту задачу можно решить несколькими способами. Рассмотрим один из них.

Будем перебирать все символы исходной строки. Для каждого символа будем определять, является ли он цифрой (см. предыдущую задачу). Если да, то добавим его в сет. В ответ нужно вывести длину полученного сета.

Frozenset в Python

Frozenset (замороженное множество) – это структура с характеристиками множества, однако, как только элементы становятся назначенными, их нельзя менять. Кортежи могут рассматриваться как неизменяемые списки, в то время как frozenset - как неизменные множества. Frozenset можно использовать как элемент сета.