Множество - это неупорядоченный набор уникальных элементов. Множество часто применяют для удаления дублирующей информации. В множестве нельзя к элементам обратиться по индексу, т.к. это неупорядоченная коллекция.

Создание множества в Python

Раздаточный материал № 95

```
 # целые числа
b = {1, 3, 5, 7, 1, 3, 5, 7}
print(b)
{1, 3, 5, 7}

# строки
c = {'ok', 'no', 'yes', 'ok', 'no', 'yes'}
print(c)
{'ok', 'no', 'yes'}
```

Преобразование строки в множество

Функция set() разбивает строку на символы, а из символов формирует множество. Символы на выходе располагаются в случайном порядке.

Раздаточный материал № 96

```
 d = set('множество')
print(d)
{'о', 'ж', 'т', 'н', 'с', 'м', 'е', 'в'}
```

Преобразование списка в множество

Раздаточный материал № 97

```
 f = set([11, 12, 13, 14, 12, 13])
print(f)
{11, 12, 13, 14}

e = set(['list', 'set', 'and', 'set'])
print(e)
{'and', 'list', 'set'}
```

Создание пустого множества

Раздаточный материал № 98

```
q = set()
```

Добавление элемента в множество - метод add. При попытке добавить дублирующий элемент, ничего не изменится.

Раздаточный материал № 99

```
 w = {3, 4, 9}
w.add(11)
print(w)
{11, 9, 3, 4}
```

Если нужно добавить в множество одновременно несколько элементов, то используют метод update.

Раздаточный материал № 100

```
 t = {5, 6, 10}
t.update([12, 15, 17])
print(t)
{5, 6, 10, 12, 15, 17}
```

Удаление элемента из множества - метод discard удаляет указанный элемент в скобках.

Раздаточный материал № 101

```
y = {20, 21, 22}
y.discard(21)
print(y)
{20, 22}
```

Метод remove делает тоже самое.

```
p = {27, 28, 29}
p.remove(27)
print(p)
{28, 29}
```

При попытке удаления несуществующего элемента методом discard никакой ошибки не будет. А при удалении с помощью метода remove, возникнет ошибка.

Метод pop удаляет случайный элемент, аргументы внутри скобок не передаются.
Раздаточный материал № 102

```
g = {27, 28, 29}
g.pop()
print(g)
{28, 29}
```

Метод clear очищает все элементы множества.
Раздаточный материал № 103

```
h = {31, 32, 33}
h.clear()
print(h)
set()
```

Функция len подсчитает количество элементов в множестве.

Раздаточный материал № 104

```
k = {34, 35, 36, 37}
print(len(k))
4
```

Пересечение множеств в Python

Оператор амперсанд (&) ищет одинаковые элементы в нескольких множествах и формирует новое множество, состоящее из пересекаемых элементов. Если пересекаемых элементов нет, то выводится пустое множество.

Раздаточный материал № 105

```
l = {38, 39, 40, 41}
z = {42, 39, 40, 43}
print(l & z)
{40, 39}

x = {44, 45, 46, 47}
c = {48, 49, 50, 51}
print(x & c)
set()
```

Объединение множеств в Python

Оператор вертикальная черта (|) объединяет элементы нескольких множеств в одно, удаляя дубли.

Раздаточный материал № 106

```
v = {52, 53, 54, 55}
b = {55, 56, 57, 58}
print(v | b)
{52, 53, 54, 55, 56, 57, 58}
```

Метод union является аналогичным способом объединения множеств.

```
 n = {59, 60, 61}
m = {62, 63, 64}
print (n.union(m))
{64, 59, 60, 61, 62, 63}
```

Разность двух множеств – это множество, в которое входят все элементы первого множества, не входящие во второе множество.

Раздаточный материал № 107

```
one = {71, 72, 73}
two = {71, 72, 75}
print(one - two)
{73}
```

Сравнение множеств в Python

Результат сравнения вернет True, если элементы одного множества, идентичны элементам другого множества.

Раздаточный материал № 108

```
one = {71, 72, 73}
two = {71, 72, 73}
```

```
print(one == two)
```
True

```
q = {65, 66, 67}
z = {68, 69, 70}
print(q == z)
```
False

Элементы множества можно перебрать с помощью цикла for
Раздаточный материал № 109

```
colors = {"red", "green", "blue"}
for color in colors:
    print(color)
```

Результат (порядок может быть другим):
red
green
blue