# Генераторы множеств и генераторы словарей

На этом занятии мы с вами поговорим о генераторах множеств и словарей. Ранее, я вам уже рассказывал о генераторах списков, когда мы их создавали, используя синтаксис:

[<способ формирования значения> for <счетчик> in <итерируемый объект>]

Например:

```
a = [x ** 2 for x in range(1, 5)]
print(a)
```

Далее, я буду полагать, что вы помните и знаете этот материал. Так вот, те же самые конструкции можно определять и для множеств со словарями. Для этого достаточно вместо квадратных скобок прописать фигурные:

```
a = {x ** 2 for x in range(1, 5)}
```

и вместо списка получим уже множество. Почему было сформировано именно множество, а не словарь? Как мы помним, множество представляет собой набор отдельных значений, а в

словаре дополнительно еще прописываются ключи. Здесь же, при генерации мы получаем серию значений, поэтому, такая коллекция в Python воспринимается именно как множество.

А вот если мы будем генерировать последовательность с ключами, например, так:

```
a = {x: x ** 2 for x in range(1, 5)}
```

то получим уже словарь.

Где такие генераторы могут использоваться? Давайте представим, что нам нужно выделить уникальные значения из списка:

```
d = [1, 2, '1', '2', -4, 3, 4]
```

Причем, все элементы следует привести к целым числам перед записью в множество. Для решения этой задачи удобно воспользоваться генератором множества:

```
a = {int(x) for x in d}
```

Видите, как легко и просто, в одну строчку мы это сделали. Мало того, генераторы работают быстрее циклов. Поэтому реализация задачи в виде:

```
set_d = set()
for x in d:
    set_d.add(int(x))
print(set_d)
```

будет и более громоздкой и более медленной. Поэтому там, где это возможно, лучше использовать соответствующие генераторы.

То же самое и со словарем. Допустим, нам нужно все его ключи записать заглавными буквами, а значения представить целыми числами:

```
m = {"неудовл.": 2, "удовл.": 3, "хорошо": '4', "отлично":
'5'}
```

Опять же используем генератор:

```
a = {key.upper(): int(value) for key, value in m.items()}
```

и на выходе получаем искомый словарь. Видите, как элегантно это можно сделать с использованием генераторов.

Также как и со списками, в генераторах множеств и словарей можно использовать условия. Например, мы хотим в множество добавить только положительные числа из списка:

```
d = [1, 2, '1', '2', -4, 3, 4]
```

Также предварительно их преобразовываем в числа и делаем проверку:

```
a = {int(x) for x in d if int(x) > 0}
```

Или, со словарем. Поменяем в нем местами ключи и значения и поместим в него только отмети от 2 до 5:

```
m = {"безнадежно": 0, "убого": 1, "неудовл.": 2, "удовл.": 3,
"хорошо": '4', "отлично": '5'}
a = {int(value): key for key, value in m.items() if
int(value) >= 2 and int(value) <= 5}
```

Вот так просто и легко можно использовать генераторы множеств и словарей в своих программах. Основное их преимущество перед обычными циклами – более высокая скорость работы и наглядность текста программы. Поэтому, если циклы можно относительно легко заменить генераторами, то стоит это делать. Также в генераторах множеств и словарей можно применять и вложенные схемы, когда один генератор вложен в другой.