Теоретический материал к занятию Генераторы и итераторы 2 занятие.

Генераторы словарей во многом аналогичны генераторам списков. В простых случаях они состоят из выражения, цикла и итерируемого объекта, которые заключаются в фигурные скобки.

```
dict1 = {x: x**2 + 1 for x in range(5)}
print(dict1)
```

Давайте разберем выражение x: x**2, которое состоит из двух выражений: первое x - это выражение которое создает ключи элементов, второе x**2 - создает значения элементов. Казалось бы, что благодаря этому можно использовать две разные переменные и создавать очень сложные словари, но из-за того что добавление элементов с одинаковыми ключами приводит к перезаписи значений, подобные трюки не получаются:

```
dict1 = {x: y for x in 'ABC' for y in 'XYZ'}
print(dict1)
```

Данный генератор выдает такой результат, потому что каждому ключу из 'ABC' может соответствовать только одно значение. Поэтому его аналогичная запись будет выглядеть так:

```
dict1 = {x: 'Z' for x in 'ABC'}
print(dict1)
```

А также можно для получения данного результата воспользоваться методом .fromkeys():

```
dict1 = {}.fromkeys('ABC', 'Z')
print(dict1)
```

но, если все-таки требуется использовать две переменные, то можно воспользоваться данным методом:

```
dict1 = {x: y for x, y in [('A', 0), ('B', 1), ('C', 2)]}

print(dict1)
```

Генераторы словарей могут содержать внутри другие генераторы. Частенько можно встретить примеры, где создаются словари у которых значениями являются списки:

```
dict1 = {x: [y for y in range(x, x + 3)] for x in range(4)}

print(dict1)
```

```
dict1 = {x: [y % 2 for y in range(10)] for x in 'ABC'}
print(dict1)
```

```
dict1 = {'ABCDE'[i]: [i % 2]*5 for i in range(5)}
print
```

```
dict1 = {x: {y: 0 for y in 'XYZ'} for x in 'ABC'}
print(dict1)
```

```
dict1 = {x: {y: x for y in 'XYZ'} for x in 'ABC'}
print(dict1)
```

Создаваемые генераторы словарей могут иметь и более сложный синтаксис:

```
dict1 = {x: {y: [z for z in range(z, z+ 2)] for y in 'XYZ'} for x, z in
zip('ABC', range(3))}
print(dict1)
```

Вложенные генераторы множеств, могут быть использованы для хранения уникальных элементов. Например, вот так можно посмотреть на все остатки от деления квадратов чисел на определенный список делителей:

```
dict1 = {i: {j**2 % i for j in range(1, 100)} for i in [4, 5, 8, 9]}
print(dict1)
```

Условия в генераторах словарей применяются точно также

```
dict1 = {x: {y: 3 for y in 'ABCD' if y != x} for x in 'ABCD'}
print(dict1)
```

Условное выражение if... else... указывается перед объявлением цикла for:

```
dict1 = {x: 1 if x in 'ACE' else 0 for x in 'ABCDEF'}

print(dict1)
```

Самый простой генератор множеств состоит из выражения, объявления цикла и итерируемого объекта, которые заключены в фигурные скобки:

```
set1 =  {i**2 % 4 for i in range(10)}
print(set1)
```

Условие if указывается после объявления цикла и итерируемого объекта:

```
set1 =   {i for i in ['ab_1', 'ac_2', 'bc_1', 'bc_2'] if 'a' not in i}
print(set1)
```

Условное выражение if... else... указывается до объявления цикла:

```
set1 =   {'A' + i[1:] if i[0] == 'a' else 'B' + i[1:] for i in ['ab_1',
'ac_2', 'bc_1', 'bc_2']}
print(set1)
```

Конструкции if и if... else... могут быть использованы одновременно:

```
set1 =  {'A' + i[1:] if i[0] == 'a' else 'B' + i[1:] for i in ['ab_1',
'ac_2', 'bc_1', 'bc_2'] if i[1] == 'c'}
print(set1)
```