Генераторы

ЛЕКЦИЯ 6

Содержание

- Генераторы
- Выражения-генераторы
- Функции-генераторы
- next() и send()

Генераторы

Генератор – это специальный инструмент, позволяющий выдавать результаты по требованию, а не все сразу.

- Генераторы не хранят выдаваемые значения

Генераторы бывают двух типов:

- **Функции-генераторы** – функции, возвращающие по одному значению за раз используя инструкцию yield, которая приостанавливает выполнение функции.
- **Выражения-генераторы** – выражения, которые не конструируют список с результатами, а возвращают объект, который будет воспроизводить результаты по требованию.

Выражения - генераторы

Генератор списков:


```
N = [i**2 for i in range(10) if i % 2 == 0]

print(N)
```

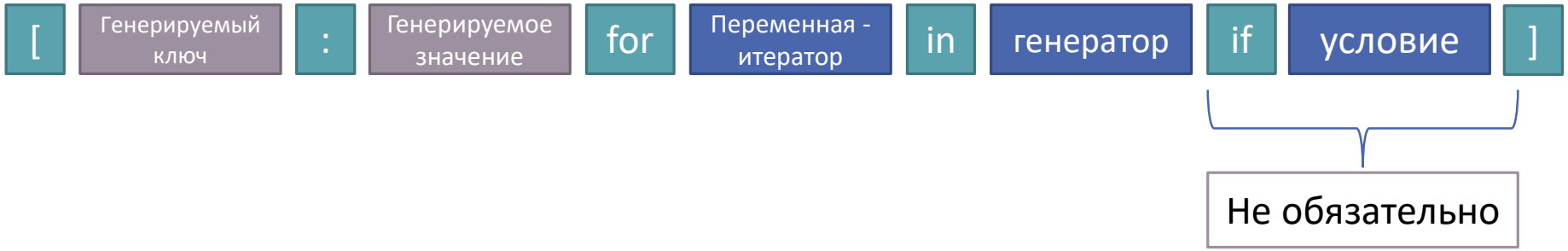
```
[0, 4, 16, 36, 64]

Process finished with exit code 0
```

При генерации кортежей квадратные скобки меняются на круглые

Выражения-генераторы

Генерация словарей:




```
N = {i: i**2 for i in range(10) if i % 2 == 0}

print(N)
```

```
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64}

Process finished with exit code 0
```

Функции-генераторы

В функциях-генераторах, вместо ключевого слово **return** используется слово **yield**.

- **yield** приостанавливает выполнение функции
- При следующем вызове функция возобновит свою работу сохранив все предыдущие значения переменных. Функция возобновляется с ключевого слова **yield**
- При использовании функции-генератора в выражениях-генераторах функция выполняется до тех пор пока не выполнит последнюю инструкцию, возвращая столько значений, сколько **yield** она встретит

```
1  def my_gen(n):
2      for i in range(n):
3          yield i**2
4
5
6  N = [i for i in my_gen(4)]
7
8  print(N)
```

```
[0, 1, 4, 9]

Process finished with exit code 0
```

next()

next() запускает функцию генератор и позволяет получить одно значение из функции генератора:

```
1   def my_gen():
2       i = 0
3       while True:
4           yield i**2
5           i += 3
6
7
8   gen = my_gen()
9
10  print(next(gen))
11  print(next(gen))
12  print(next(gen))
```

```
0
9
36

Process finished with exit code 0
```

send()

Метол **send**() запускает функцию генератор, передавая за место **yield** использованного последний раз, свой аргумент. После этого **send**() работает как **next**()

- Однако для того чтобы использовать **send**() функция-генератор должна достигнуть хотя бы первого **yield**. Поэтому до использования **send**() следует вызвать **next**()

```
1   def my_gen():
2       i = 0
3       while True:
4           i = yield i ** 2
5
6
7   gen = my_gen()
8   next(gen)
9
10  print(gen.send(4))
11  print(gen.send(5))
12  print(gen.send(2))
```

```
16
25
4

Process finished with exit code 0
```