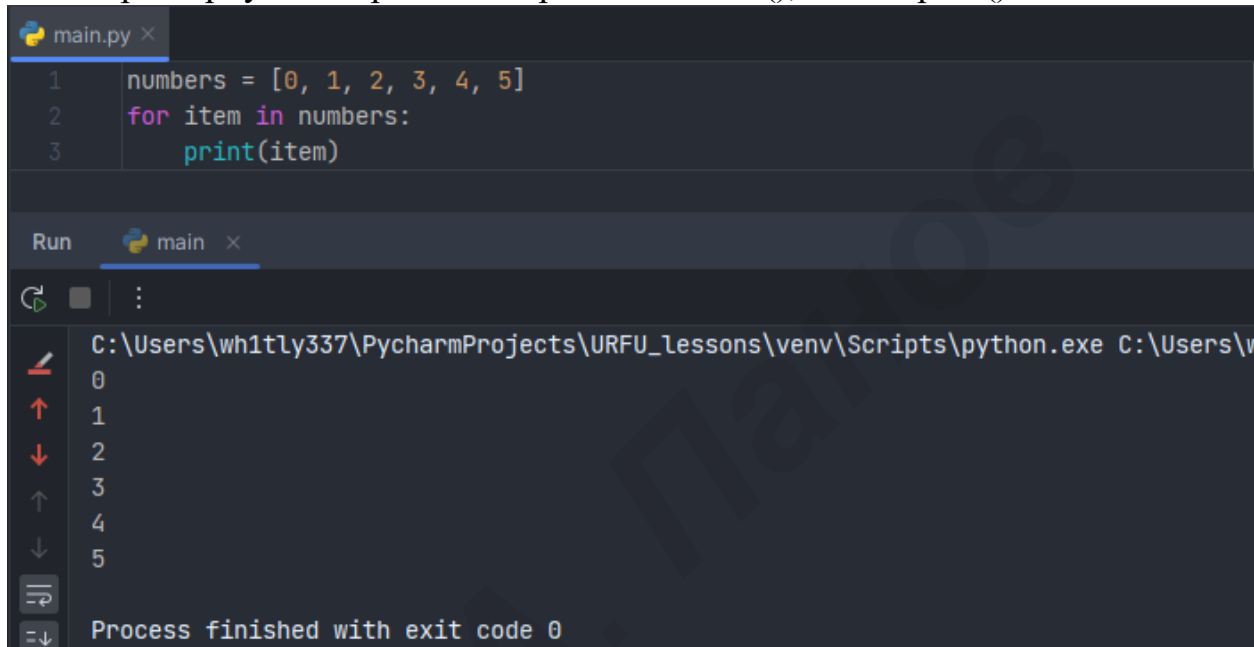


# Лабораторная работа 11

## ТЕМА 11. Итераторы и генераторы

### Лабораторные задания:

- 1) Простой итератор, но у него нет гибкой настройки, например его нельзя развернуть. Он работает просто как `next()`, но нет `prev()`



```
main.py x
1 numbers = [0, 1, 2, 3, 4, 5]
2 for item in numbers:
3     print(item)
```

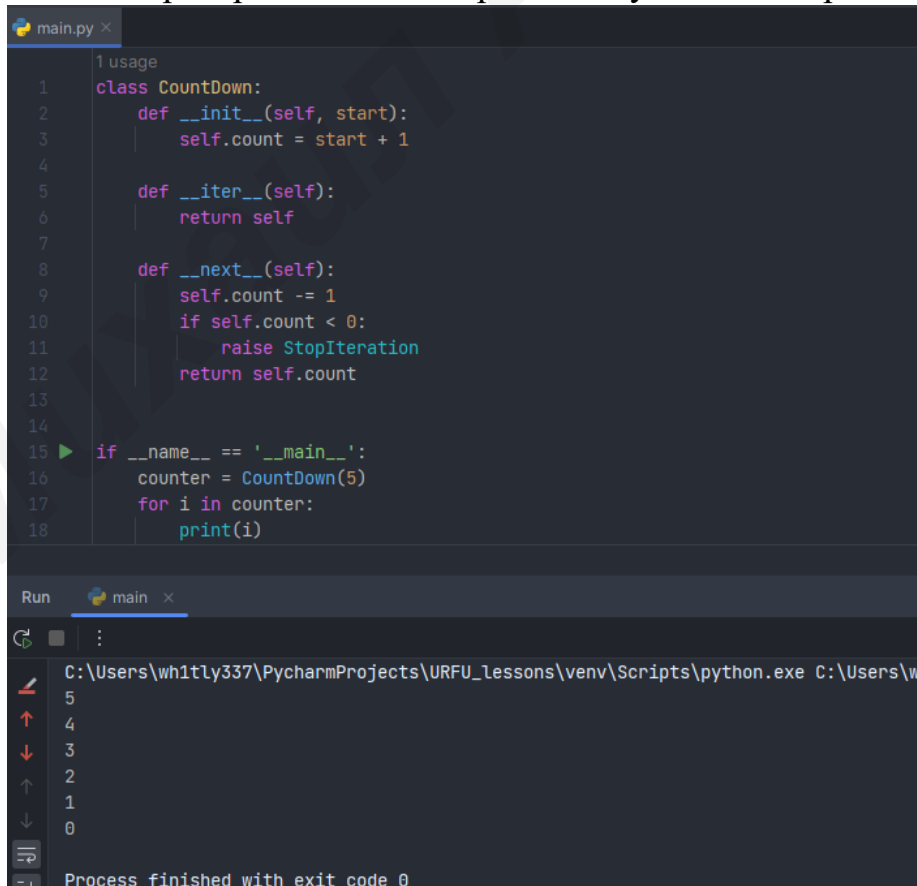
Run main x

C:\Users\wh1tly337\PycharmProjects\URFU\_lessons\venv\Scripts\python.exe C:\Users\w

0  
1  
2  
3  
4  
5

Process finished with exit code 0

- 2) Класс итератор с гибкой настройкой и удобным применением



```
main.py x
1 usage
2 class Countdown:
3     def __init__(self, start):
4         self.count = start + 1
5
6     def __iter__(self):
7         return self
8
9     def __next__(self):
10        self.count -= 1
11        if self.count < 0:
12            raise StopIteration
13        return self.count
14
15 if __name__ == '__main__':
16     counter = Countdown(5)
17     for i in counter:
18         print(i)
```

Run main x

C:\Users\wh1tly337\PycharmProjects\URFU\_lessons\venv\Scripts\python.exe C:\Users\w

5  
4  
3  
2  
1  
0

Process finished with exit code 0

- 3) Генератор списка

```
main.py x
1 a = [i ** 2 for i in range(1, 5)]
2
3 print('a - ', a)
4 for i in a:
5     print(i)
6
7 print('iter(a) - ', iter(a))
8 for i in a:
9     print(i)
```

Run main x

C:\Users\wh1tly337\PycharmProjects\URFU\_lessons\venv\Scripts\python.exe C:\Users\w  
a - [1, 4, 9, 16]  
1  
4  
9  
16  
iter(a) - <list\_iterator object at 0x0000028ECB0F9390>  
1  
4  
9  
16  
Process finished with exit code 0

#### 4) Выражения генераторы

```
main.py x
1 b = (i ** 2 for i in range(1, 5))
2 print(b) # вывод не такой, как у генератора списков
3 print('first')
4 for i in b:
5     print(i)
6 print('second')
7 # из-за особенностей выражений генераторов,
8 # они не будут выводиться больше одного раза
9 for i in b:
10    print(i)
```

Run main x

C:\Users\wh1tly337\PycharmProjects\URFU\_lessons\venv\Scripts\python.exe C:\Users\w  
<generator object <genexpr> at 0x000001E35CB09220>  
first  
1  
4  
9  
16  
second  
Process finished with exit code 0

- 5) Такой же счетчик, как и в первом задании, только это генератор и использует yield



The screenshot shows a PyCharm IDE window with a file named `main.py`. The code defines a generator function `countdown` and uses it in the `__main__` block. The output console shows the numbers 5 through 0, indicating the generator yields values in descending order.

```
1 usage
2 def countdown(count):
3     while count >= 0:
4         yield count
5         count -= 1
6
7 if __name__ == '__main__':
8     counter = countdown(5)
9     for i in counter:
10        print(i)
```

Run main x

C:\Users\wh1tly337\PycharmProjects\URFU\_lessons\venv\Scripts\python.exe C:\Users\w  
5  
4  
3  
2  
1  
0  
Process finished with exit code 0