Итераторы и Генераторы

Итератор - это объект, который поддерживает функцию next() для перехода к следующему элементу, либо бросает исключение, если элементов больше нет.

Объекты, элементы которых можно перебирать в цикле *for*, содержат в себе объект итератор, для того, чтобы его получить необходимо использовать функцию ***iter()***, а для извлечения следующего элемента из итератора – функцию ***next(),*** а когда эти элементы заканчиваются, генерируется исключение ***StopIteration****.*

```
num_list = [1, 2, 3, 4, 5] - получаем итератор из любого итерируемого объекта
itr = iter(num_list)
print(next(itr))
1
print(next(itr))
2

num_list = (x**2 for x in ranga(5)) - получаем итератор с помощью выражения-генератора
print(next(num_list))
0
print(next(num_list))
1
```

Создание собственных итераторов

```
class SimpleIterator:
    def __iter__(self):
            return self

    def __init__(self, limit):
        self.limit = limit
        self.counter = 0

    def __next__(self):
        if self.counter < self.limit:
            self.counter += 1
            return 1
        else:
            raise StopIteration

iter1 = SimpleIterator(3)
for i in iter1:
    print(i)
```

Генератор – это итератор, элементы которого можно перебирать только один раз. Вместо ключевого слова *return* в генераторе используется ***yield****.*

При вызове ***yield*** функция не прекращает свою работу, а “замораживается” до очередной итерации, запускаемой функцией *next().*

```
def simple_generator(val):
   while val > 0:
       val -= 1
       yield 1

gen_iter = simple_generator(5)
```