

ИТЕРАТОРЫ, ГЕНЕРАТОРЫ



КИРИЛЛ ТАБЕЛЬСКИЙ



КИРИЛЛ ТАБЕЛЬСКИЙ

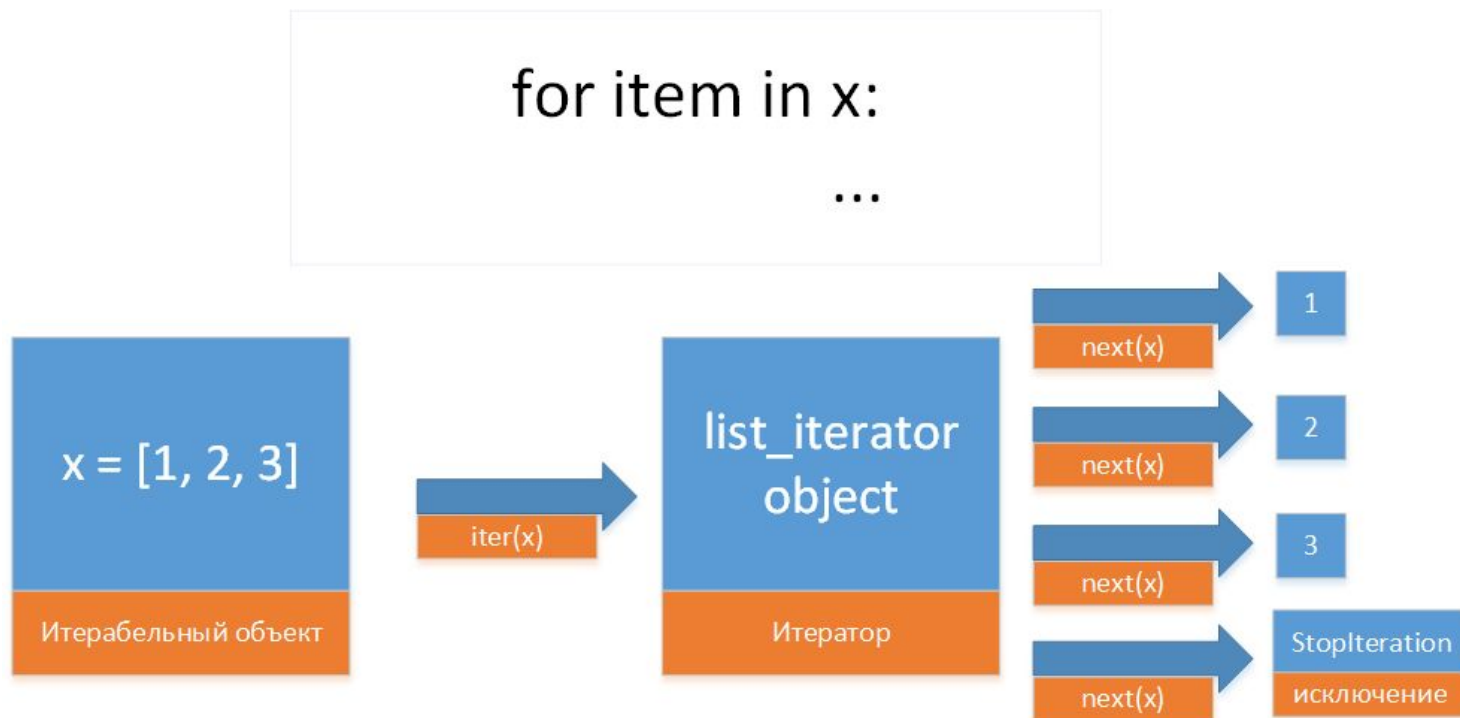
Lightmap



План занятия

1. [list и память](#)
2. [Как работает цикл for](#)
3. [Итератор](#)
4. [Генератор и yield](#)

Что скрывает цикл for



Делаем итератор

```
class HelloWorldIterator:
```

```
    def __iter__(self):  
        print('Цикл начинается')  
        return self
```

Выполнится перед
первой итерацией

```
    def __next__(self):
```

```
        return 'Hello World'
```

```
hello_world_iterator = HelloWorldIterator()
```

```
for item in hello_world_iterator:  
    print(item)
```

Делаем итератор

```
class HelloWorldIterator:

    def __init__(self, n):
        self.n = n

    def __iter__(self):
        print('Цикл начинается')
        self.counter = 0
        return self

    def __next__(self):
        self.counter += 1

        if self.counter > self.n:
            print('Цикл завершается')
            raise StopIteration


        return 'Hello World'
```

Условие выхода из
цикла

```
for item in HelloWorldIterator(n=3):
```

Генератор

```
def hello_world():  
    while True:  
        yield 'Hello World'
```



```
hello_world_generator = hello_world()  
  
for item in hello_world_generator:  
    print(item)
```

Генератор

```
def hello_world(n):  
    for i in range(n):  
        yield 'Hello World'  
  
for item in hello_world(n=3):  
    print(item)
```

Функции с yield возвращают **generator object**

```
hello_world_generator = hello_world(n=3)
```



У **generator object** есть методы `__iter__` и `__next__`,
которые формируются автоматически

Генератор

```
def my_generator(*args, **kwargs):  
    ...  
    while True:
```

```
        ...  
        print('Код до yield')
```

```
        yield item
```

```
        ...  
        print('Код после yield')
```

```
for item in my_generator(...):
```

Первая итерация

```
print('Код до yield')
```

```
yield item
```

Остальные итерации

```
print('Код после yield')
```

```
print('Код до yield')
```

```
yield item
```

Итераторы, генераторы и коллекции

```
class HelloWorld:

    def __init__(self, n):
        ...
    def __iter__(self):
        ...
    def __next__(self):
        ...

helloworld_tuple = tuple(HelloWorld(n=3))
// ('HelloWorld', 'HelloWorld', 'HelloWorld')

helloworld_list = list(HelloWorld(n=3))
// ['HelloWorld', 'HelloWorld', 'HelloWorld']

helloworld_set = set(HelloWorld(n=3))
// {'HelloWorld'}
```

Потребление памяти

Итератор и генератор знают текущий **item** и как вычислить следующий

Коллекция, например **list**, хранит в себе все **item**

```
from pymppler.sizeof import sizeof

range_1_million = range(1 000 000)
print(f'range_1_million size is: {sizeof(range_1_million) / 1024 } kb')

list_1_million = list(range(1 000 000))
print(f'list_1_million size is: {sizeof(list_1_million) / 1024} kb')
```

range_1_million size is: 0.046875 kb

list_1_million size is: 39062.546875 kb

comprehension

```
list_1_hundred_x2 = [i * 2 for i in range(100)]
```

что сделать с *i*

откуда брать *i*

```
gen_1_hundred_x2 = (i * 2 for i in range(100))
```

```
tuple_1_hundred_x2 = tuple(i * 2 for i in range(100))
```

```
set_1_hundred_x2 = set(i * 2 for i in range(100))
```

comprehension

```
list_1_hundred_x2 = [i * 2 for i in range(100) if i % 2 == 0]
```

что сделать с *i*

откуда брать *i*

брать только эти *i*

```
gen_1_hundred_x2 = (i * 2 for i in range(100) if i % 2 == 0)
```

```
tuple_1_hundred_x2 = tuple(i * 2 for i in range(100) if i % 2 == 0)
```

```
set_1_hundred_x2 = set(i * 2 for i in range(100) if i % 2 == 0)
```

Трудности перевода

В русскоязычной литературе **list comprehension** часто переводят как генератор списков, что иногда путает **generator object**

```
[... for ... in ... if ...]    list comprehension (генератор списков)
```

```
def yield_function(*args, **kwargs):  
    ...  
    while True:  
        ...  
        yield item  
    ...
```

```
generator = yield_function(...)    generator object (генератор)
```

```
generator = (... for ... in ... if ...)    тоже генератор
```



Дополнительные материалы

- <https://tproger.ru/translations/implementing-zip-list-comprehensions/>
- <https://pythonworld.ru/moduli/modul-itertools.html>
- <https://docs.python.org/3/library/itertools.html>
- <https://pythonist.ru/map-filter-i-reduce-funkczii-v-python/>
- <https://more-itertools.readthedocs.io/en/stable/index.html>



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задаём в чате вашей группы!
- Задачи можно сдавать по частям.
- Зачёт по домашней работе проставляется после того, как приняты **все задачи**.

 нетология

Задавайте вопросы и напишите отзыв о лекции!

КИРИЛЛ ТАБЕЛЬСКИЙ