Генераторы

Основы Python

План урока

- Обсудим генераторы

ГЕНЕРАТОРЫ

это функции, которые возвращают итератор по значениям, которые они сами производят

Пример

ooo | код

```
def get_fitst_3_numbers():
    yield 1
    yield 2
    yield 3

g = get_fitst_3_numbers()

print(next(g))

print(next(g))

print(next(g))

print(next(g))
```

ooo | вывод

```
>>> 1
>>> 2
>>> 3
>>> StopIteration
```

3 раза используем слово “yield”

Пример

ooo | код

```
def get_fitst_3_numbers():
    yield 1
    yield 2
    yield 3

g = get_fitst_3_numbers()

print(next(g))

print(next(g))

print(next(g))

print(next(g))
```

ooo | вывод

```
>>> 1
>>> 2
>>> 3
>>> StopIteration
```

Присваиваем вызов генератора переменной “g”

Пример

ooo | код

```
def get_fitst_3_numbers():
    yield 1
    yield 2
    yield 3

g = get_fitst_3_numbers()

print(next(g))

print(next(g))

print(next(g))

print(next(g))
```

ooo | вывод

```
>>> 1
>>> 2
>>> 3
 >>> StopIteration
```

При помощи оператора “next” получаем следующие значения

Пример

ooo | код

```
def get_fitst_3_numbers():
    yield 1
    yield 2
    yield 3

g = get_fitst_3_numbers()

print(next(g))

print(next(g))

print(next(g))

print(next(g))
```

ooo | вывод

```
>>> 1
>>> 2
>>> 3
>>> StopIteration
```

Получаем исключение “StopIteration”

Пример

ooo | код

```
def get_fitst_3_numbers():
    yield 1
    yield 2
    yield 3

g = get_fitst_3_numbers()

for num in get_fitst_3_numbers():
    print(num)
```

ooo | вывод

```
>>> 1
>>> 2
>>> 3
```

Пример

ooo | код

```
def get_fitst_3_numbers():
    yield 1
    yield 2
    yield 3

g = get_fitst_3_numbers()

first_3_num = (num for num in get_fitst_3_numbers())

print(first_3_num)
```

ooo | вывод

```
>>> <generator object <genexpr> at 0x7fbec6b61fc0>
```

Переменной “first_3_num” присваивается генераторное выражение, с которым можно работать как с итератором

Пример

ooo | код

```
def get_fitst_3_numbers():
    yield 1
    yield 2
    yield 3

g = get_fitst_3_numbers()

first_3_num = (num for num in get_fitst_3_numbers())

first_3_num_list = list(first_3_num)

print(first_3_num_list)
```

ooo | вывод

```
>>> [1, 2, 3]
```

Пример

ooo | код

```
def fibonacci_gen():
    first = 0
    second = 1
    while True:
        yield first
        first, second = second, first + second

fib_gen = fibonacci_gen()

print(next(fib_gen))
print(next(fib_gen))
print(next(fib_gen))
print(next(fib_gen))
print(next(fib_gen))
print(next(fib_gen))
infinite_sequence = (y for y in fibonacci_gen())
```

ooo | вывод

```
>>> 0
>>> 1
>>> 1
>>> 2
>>> 3
>>> 5
```

Пример

ooo код

```
gen_expr = (num for num in [1, 2, 3])

print(gen_expr)
```

ooo вывод

```
>>> <generator object <genexpr> at 0x7fde708cc2e0>
```

Пример

КОД

```
def print_hello_name():
    name = None
    i = 0
    while True:
        i += 1
        print(f'Hello, {yield name}!', i)

hello_generator = print_hello_name()

print(next(hello_generator))

print(hello_generator.send('Peter'))

print(next(hello_generator))

print(hello_generator.send('Steven'))

print(next(hello_generator))

print(next(hello_generator))
```

ВЫВОД

```
>>> None
>>> Hello, Peter! 1
>>> None
>>> Hello, None! 2
>>> None
>>> Hello, Steven! 3
>>> None
>>> Hello, None! 4
>>> None
>>> Hello, None! 5
>>> None
```

Пример

ooo | код

```
def get_numbers_catch_exception():
    num = 1
    while True:
        try:
            yield num
            num += 1
        except AttributeError:
            print('OMG! This is an AttributeError')

nums = get_numbers_catch_exception()
print(next(nums))

print(next(nums))

nums.throw(AttributeError)
print(next(nums))
```

ooo | вывод

```
>>> 1
>>> 2
>>> OMG! This is an AttributeError
>>> 3
```

Пример

ooo | код

```
def manage_progression():
    num = 1
    step = 1
    while True:
        next_step = (yield num)
        if next_step is not None:
            step = next_step
        num += step
        if next_step is not None:
            yield num

mp = manage_progression()
print(next(mp))
print(next(mp))
print(next(mp))
print(next(mp))
print(next(mp))
print(next(mp))
print(next(mp))
mp.send(10)
print(next(mp))
print(next(mp))
```

ooo | вывод

```
>>> 1
>>> 2
>>> 3
>>> 4
>>> 5
>>> 6
>>> 16
>>> 26
```

Пример

ooo | КОД

```
numbers = [number for number in range(10)]
print(numbers, type(numbers))

character_code_table = {character: ord(character) for character in 'abcd'}
print(character_code_table, type(character_code_table))
```

ooo | ВЫВОД

```
>>> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] <class 'list'>
>>> {'a': 97, 'b': 98, 'c': 99, 'd': 100} <class 'dict'>
```