Генераторы

Генераторы | Функции, которые возвращают итератор по значениям, которые они сами производят

Главное отличие генераторов от обычных функций в Python заключается в использовании ключевого слова "yield" вместо "return". Когда функция содержит оператор "yield", она становится генератором, который может возвращать значение и приостанавливать свою работу до момента, когда будет вызван снова

Пример

```
def generator_func():
    for i in range(5):
        yield i
generator = generator_func()  # Создание объекта генератора
for item in generator:
    print(item)
```

В этом примере функция "generator_func" является генератором, который при каждом вызове оператора "yield" возвращает значение из списка [0, 1, 2, 3, 4]. При каждой итерации по объекту генератора значение сохраняется и печатается

Генераторы могут быть использованы в циклах, списковых выражениях или других операциях, где ожидается последовательность данных. Они позволяют экономить память и улучшают производительность программы

Генераторы также могут принимать параметры и изменять свое поведение в зависимости от них. Например, генератор, который генерирует последовательность чисел от 0 до n:

Пример

```
def number_generator(n):
    for i in range(n):
        yield i

generator = number_generator(10)  # Создание объекта генератора

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

В этом примере функция "number_generator" принимает параметр "n" и использует его для определения длины последовательности чисел, которые будут сгенерированы

Генераторы могут быть бесконечными, что означает, что они могут продолжать генерировать значения до бесконечности. Например, генератор, который генерирует бесконечную последовательность чисел Фибоначчи

Пример

```
Пример:
def fibonacci_generator():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

generator = fibonacci_generator()  # Создание объекта генератора

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

В этом примере функция "fibonacci_generator" использует бесконечный цикл while True для генерации чисел Фибоначчи. Каждый раз, когда оператор "yield" выполняется, следующее число в последовательности Фибоначчи возвращается, и цикл продолжает работу

Итог

В этой теме мы разобрали понятие функции и ее синтаксис, рассмотрели процесс возвращения и передачи аргументов, изучили глобальные и локальные переменные, познакомились с понятием рекурсии, а также поговорили о замыканиях и и декораторах