Итераторы и генераторы

Содержит счётное количество элементов (коллекцию)

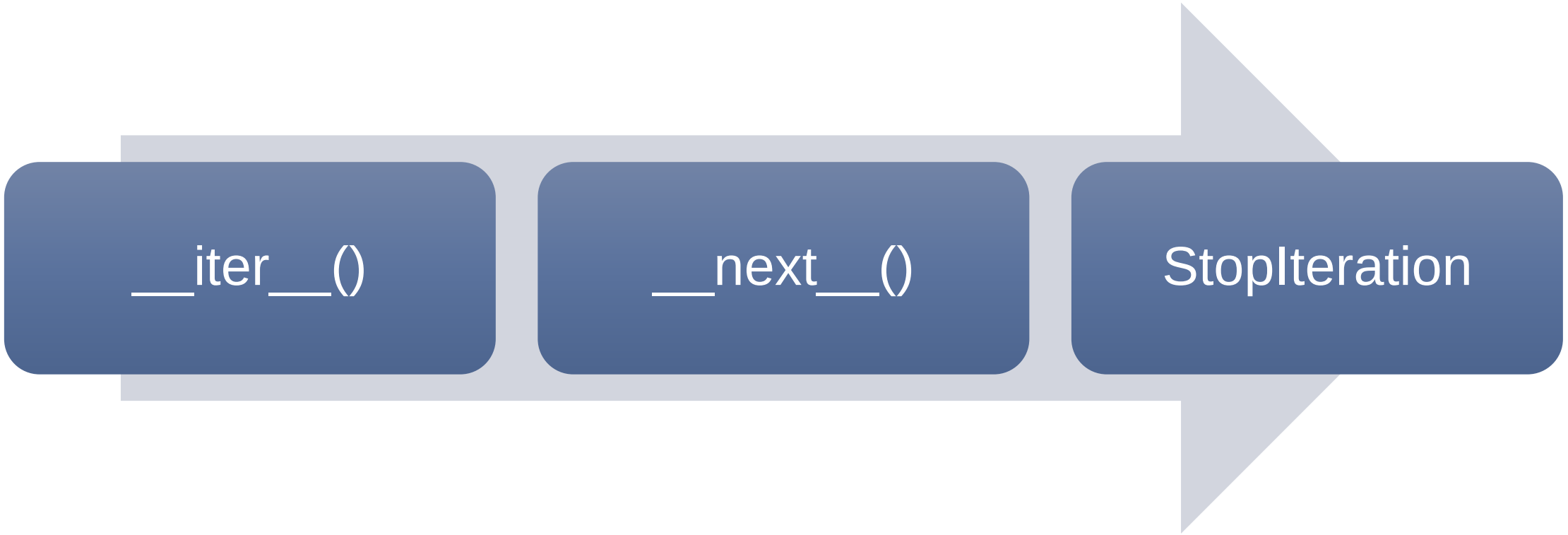
Реализует протокол итерации

Структура коллекции и порядок обхода могут быть любыми

Итератор

Специальный класс, который позволяет обойти некоторую коллекцию элементов в определённом порядке

Протокол итерации



Итераторы

Может выдавать любое количество значений, каждый вызов функции при этом возвращает **одноразовый** генератор

Неявно реализует протокол итерации

Может генерировать последовательность элементов «на лету» с помощью инструкции **yield**

Генератор

Функция, выдающая последовательность значений

Схема работы генератора

Генераторы

Преимущества генераторов

- Простота имплементации и использования
- Экономия памяти определена структурой генератора
- Представление бесконечного потока данных как объекта

Генераторные выражения

Декораторы

```
def my_decorator(func):
        def wrapper():
                #do something
                func()
                #do something else
        return wrapper

@my_decorator
def hello ():
print("hello")

'''
def hello():
print(“hello")

hello = my_decorator(hello)
'''
```

Декоратор

Функция, «оборачивающая» другую функцию тем самым добавляя некоторую логику

Преимущества использования декораторов

Явный синтаксис, привлекающий внимание

Декоратор применяется один раз сразу после создания функции

Одним и тем же декоратором можно изменять поведения множества функций

Декоратор для вычисления времени выполнения функции