Асинхронный итератор

Для начала вспомним, что такое итератор. **Итераторы** — это объекты Python, которые обладают магическими методами **__iter__** и **__next__**. Метод **__next__** на последней итерации должен вызывать исполючение **StopIteration .**

Пример простого итератора, который при каждом вызове метода __next__ будет возвращать текущее число вызовов функции до тех пор, пока мы не достигнем ограничения 5 раз.

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

    def __iter__(self):
        return self

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


iter = Iterator(5)
for i in iter:
    print(i)
```

```
1
2
3
4
5
```

Существует такой синтаксис асинхронной итерации, как **async for item in aiter**.

```
async for item in aiter:
    pass
```

Асинхронные итераторы — это объекты Python, которые обладают магическими методами **__aiter__** и **__anext__**. Метод **__anext__** на последней итерации должен вызывать исполючение **StopAsyncIteration**. Отличается от обычного итератора возможностью работать асинхронно.

документация

Небольшой пример:

Допустим у нас есть список людей и мы хотим окликнуть каждого из них. После того, как мы называет имя из скиска - есть некая задержка, пока этот человек ответит.

```
import asyncio
import random

class Crowd:
    def __init__(self, people: list[str]):
        self._people = people
        self._i = 0

    def __aiter__(self):
        return self

    async def __anext__(self):
        if self._i >= len(self._people):
            raise StopAsyncIteration

        human = self._people[self._i]
        self._i += 1
        delay = random.randint(1,5)
        await asyncio.sleep(delay)

        return f"wait for {human} {delay}c"


async def main():
    crowd = Crowd(["Andrey", "Alex", "Artem", "Igor"])
    async for i in crowd:
        print(i)


asyncio.run(main())
```

```
Hey, Andrey!
I an here! delay=4c
Hey, Alex!
I an here! delay=1c
Hey, Artem!
I an here! delay=4c
Hey, Igor!
I an here! delay=2c
```

Пример использования:

При работе с БД мы хотим запрашивать данные по частям, с передачей управления другим подпрограммам. Например MotorCursor в библиотеке для работы с MongoDB или ResultCursor в sqlalchemy. Позже в курсе подробнее обсудим подобные реализации.