?????????? ?????????? ????????

Асинхронный контекстный менеджер.

Это класс, реализующий два специальных метода-сопрограммы: __aenter__, который асинхронно захватывает ресурс, и __aexit__, который закрывает ресурс. Сопрограмма __aexit__ принимает несколько аргументов, относящихся к обработке исключений. Пример для сокетов:

```
import asyncio
import socket
from types import TracebackType
from typing import Optional, Type
class ConnectedSocket:
    def __init__(self, server_socket):
        self._connection = None
        self._server_socket = server_socket
    async def __aenter__(self):
        print('Вход в контекстный менеджер, ожидание подключения')
        loop = asyncio.get_event_loop()
        connection, address = await loop.sock_accept(self._server_socket)
        self._connection = connection
        print('Подключение подтверждено')
        return self._connection
    async def __aexit__(self,
                        exc_type: Optional[Type[BaseException]],
                        exc_val: Optional[BaseException],
                        exc_tb: Optional[TracebackType]):
        print('Выход из контекстного менеджера')
        self._connection.close()
        print('Подключение закрыто')
async def main():
    loop = asyncio.get_event_loop()
    server_socket = socket.socket()
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_address = ('127.0.0.1', 8000)
```

```
    server_socket.setblocking(False)
    server_socket.bind(server_address)
    server_socket.listen()
    async with ConnectedSocket(server_socket) as connection:
        data = await loop.sock_recv(connection, 1024)
        print(data)
asyncio.run(main())
```

Revision #3
Created 5 May 2025 09:14:02 by Admin
Updated 24 June 2025 14:28:25 by Admin