Лабораторная работа №7

Тема: «Асинхронное программирование на Python»

Цель: Изучить основы асинхронного программирования на языке Python.

Научиться использовать библиотеку asyncio для написания асинхронного кода.

Ознакомиться с концепцией корутин и задач (tasks).

Требования к выполнению лабораторной работы №7

1. Изучите теоретическую часть к седьмой лабораторной работе:
    a. Теоретическая часть к седьмой лабораторной работе (notebook_7.ipynb).
    b. Лекция №7.
2. Создайте новый проект.
3. Запустите примеры из лабораторной работы.
4. Выполните задание и отправьте решение в ОРИОКС.

Формат защиты лабораторных работ:

1. Продемонстрируйте выполненные задания.
2. Ответьте на вопросы по вашему коду.
3. При необходимости выполните дополнительное (*дополнительные*) задание от преподавателя.
4. Ответьте (*устно*) преподавателю на контрольные вопросы.

Список вопросов

1. Что такое асинхронное программирование? В чем его отличие от синхронного? Приведите примеры, где асинхронное программирование предпочтительнее синхронного.
2. Объясните ключевые понятия: Future, Task, Event Loop. Как они взаимодействуют в асинхронных программах на Python? Приведите примеры использования этих понятий.
3. Что такое корутины (coroutines)? Как они связаны с асинхронностью? В чем их отличие от обычных функций?
4. В чем отличие корутин (coroutines) от генераторных функций?
5. Что такое async и await? Для чего они используются? Как они влияют на выполнение кода?
6. Как обрабатывать исключения в асинхронном коде? Какие особенности у обработки исключений в асинхронных функциях?
7. Как организовать асинхронную обработку нескольких задач?
8. Как управлять временем выполнения асинхронных задач?
9. Какие есть паттерны проектирования для асинхронных приложений?
10. Что такое asyncio.CancelledError и когда оно возникает?
11. Какие асинхронные функции для работы с файлами предоставляет стандартная библиотека Python?
12. В чем преимущества использования aiohttp по сравнению с синхронными библиотеками, такими как requests?
13. В чем разница между параллельным и асинхронным программированием?
14. Что такое asyncio.create_task и как его использовать?

Задание 1. Асинхронная обработка файлов

Напишите асинхронную функцию, которая читает содержимое файла и возвращает его в виде строки. Используйте библиотеку aiofiles.

Задание 2. Асинхронная обработка данных

Напишите асинхронную функцию, которая выполняет обработку данных (задачу обработки данных придумайте самостоятельно). Используйте библиотеку asyncio.

Задание 3. Асинхронное выполнение задач

Создайте корутину, которая будет принимать имя и выводить приветствие через 2 секунды. Создайте список имен и с помощью asyncio.gather() выполните корутину для каждого имени одновременно.

Задание 4. Асинхронный HTTP-запросы

Используйте библиотеку aiohttp для выполнения асинхронных HTTP-запросов.

Напишите корутину, которая запрашивает данные с нескольких веб-сайтов (список и количество веб-сайтов выберите самостоятельно) и выводит статус ответа.

Задание 5. Обработка исключений

Дополните предыдущую задачу обработкой возможных исключений, связанных с сетевыми ошибками или неверными URL.