3 модуль. Задания

Глава 3.1. Цикл событий и основы asyncio

1. Что происходит, если на этапе обработки ошибок цикла событий возникает ошибка?
– Цикл событий начинается сначала
– Цикл событий продолжает работу, игнорируя ошибку
– Цикл событий завершается и вызывается ошибка
– Цикл событий переходит к следующему этапу
Правильный ответ: Цикл событий завершается и вызывается ошибка

2. Что делает следующий код?
– Выполняет синхронную функцию sync_func() внутри асинхронной функции async_func и выводит результат 42
– Выполняет асинхронную функцию async_func() внутри синхронной функции sync_func и выводит результат 42
– Выдает ошибку, так как синхронная функция не может быть выполнена внутри асинхронной функции
– Создает бесконечный цикл событий
Правильный ответ: Выполняет синхронную функцию sync_func() внутри асинхронной функции async_func и выводит результат 42

3. Что такое цикл событий (event loop)?
– Механизм, который создает новые потоки для выполнения задач
– Механизм, который автоматически распределяет задачи между процессорами
– Механизм, который позволяет выполнять асинхронные задачи параллельно
– Механизм, который последовательно выполняет задачи в одном потоке
Правильный ответ: Механизм, который последовательно выполняет задачи в одном потоке

4. Можно ли вызывать asyncio.run() в том же потоке, где уже запущен другой цикл событий?
– Зависит от версии Python
– Да, можно
– Нет правильного ответа
– Нет, нельзя
Правильный ответ: Нет, нельзя

Глава 3.2. Оператор await

1. Что происходит, когда интерпретатор встречает await перед Awaitable объектом?
– Начинается выполнение новой корутины
– Завершается выполнение текущей корутины
– Продолжается выполнение текущей корутины
– Приостанавливается выполнение текущей корутины до получения результата
Правильный ответ: Приостанавливается выполнение текущей корутины до получения результата

2. Какие преимущества имеет оператор await?
– Предотвращение блокировки
– Предотвращение конкурентного выполнения
– Ухудшение читаемости кода
– Обеспечение конкурентного выполнения
Правильный ответ: Предотвращение блокировки; обеспечение конкурентного выполнения

Глава 3.3. Future и конкурентность

1. Что такое Future?
– Модуль для работы с асинхронными задачами
– Тип данных для асинхронного программирования
– Тип объекта с результатом выполненной операции
– Тип объекта, представляющий результат задачи, которая еще не завершена
Правильный ответ: Тип объекта, представляющий результат задачи, которая еще не завершена

2. Какое утверждение верно относительно await?
– Он предотвращает конкурентное выполнение
– Он ухудшает читаемость кода
– Он используется только в синхронных функциях
– Он позволяет писать асинхронный код, похожий на синхронный
Правильный ответ: Он позволяет писать асинхронный код, похожий на синхронный

Глава 3.4. Корутины

1. Что такое корутины?
– Функции, которые не могут быть приостановлены
– Функции, выполняемые только синхронно
– Функции, которые могут быть приостановлены и возобновлены
– Функции, вызываемые один раз
Правильный ответ: Функции, которые могут быть приостановлены и возобновлены

2. Что делает ключевое слово async def?
– Определяет синхронную функцию
– Определяет переменную
– Определяет класс
– Определяет асинхронную функцию
Правильный ответ: Определяет асинхронную функцию

Глава 3.5. asyncio.create_task

1. Что такое asyncio.create_task()?
– Функция для создания асинхронных задач
– Функция для создания потоков
– Функция для создания синхронных задач
– Функция для создания процессов
Правильный ответ: Функция для создания асинхронных задач

2. Какой объект возвращает asyncio.create_task()?
– Future
– Coroutine
– Task
– Thread
Правильный ответ: Task

Глава 3.6. Future и Task

1. Какой объект представляет результат асинхронной операции, которая еще не завершена?
– Thread
– Task
– Coroutine
– Future

Правильный ответ: Future

2. Какой метод используется для проверки, завершился ли объект Future?
– has_result()
– is_completed()
– is_done()
– done()
Правильный ответ: done()