ВЫЗОВ ФУНКЦИЙ ПО РАСПИСАНИЮ

ВВЕДЕНИЕ

Задачи урока

01 Разобраться, как работают задачи по расписанию

02 Понять, в чем отличие у call_soon и call_later

Метод call_soon()

Метод call_soon() является одним из основных инструментов в библиотеке asyncio для планирования и запуска асинхронных задач в event loop (цикле событий)

Он позволяет добавлять вызовы функций или корутин в очередь на выполнение в ближайшее время

Основным параметром этого метода является **callback** — функция или метод, которую нужно выполнить

Остальные параметры args представляют собой дополнительные аргументы, которые будут переданы в функцию при ее вызове. **Context** — это необязательный параметр, который может использоваться для указания контекста выполнения функции

Когда вызывается метод call_soon(), задача добавляется в очередь выполнения **event loop**, и она будет выполнена при следующей итерации цикла событий. Таким образом, эта функция позволяет планировать асинхронные задачи в будущем, а не выполнять их немедленно

Пример использования **call_soon()**

```python
import asyncio

async def my_task():
    print("Асинхронная задача выполняется")

def callback():
    print("Callback функция выполнена")

loop = asyncio.get_event_loop()
loop.call_soon(callback)
loop.run_until_complete(my_task())
```

В этом примере мы создаем асинхронную задачу **my_task()**, которая будет выполняться с помощью цикла событий. Затем мы используем call_soon() для добавления **callback** функции **callback()** в очередь выполнения. При запуске цикла событий метод call_soon() будет вызывать **callback()** и выводить соответствующее сообщение

> Метод call_soon() полезен, когда нам нужно запланировать выполнение функции или корутины в **event loop** в ближайшем будущем. Он позволяет управлять порядком и временем выполнения асинхронных задач в asyncio

> Метод call_later() является еще одним полезным инструментом в библиотеке asyncio для планирования асинхронных задач. Он позволяет задать отложенное выполнение функции или корутины в **event loop** (цикле событий), через определенное количество времени

Первый параметр

delay
представляет собой время задержки в секундах, перед выполнением задачи

Второй параметр

callback
это функция или метод, которую нужно выполнить

Остальные параметры args представляют собой дополнительные аргументы, которые будут переданы в функцию при ее вызове

Context
это необязательный параметр, который может использоваться для указания контекста выполнения функции

При вызове метода call_later(), задача добавляется в очередь выполнения **event loop**, и она будет выполнена после указанной задержки времени. Такое отложенное выполнение позволяет планировать задачи на выполнение в будущем

Пример использования **call_later()**

```
import asyncio

async def my_task():
    print("Асинхронная задача выполняется")

def callback():
    print("Callback функция выполнена")

loop = asyncio.get_event_loop()
loop.call_later(2, callback)
loop.run_until_complete(my_task())
```

В этом примере мы создаем асинхронную задачу **my_task()**, которая будет выполняться с помощью цикла событий. Затем, с помощью **call_later()**, мы добавляем функцию **callback()** в очередь выполнения с задержкой в 2 секунды. При запуске цикла событий, метод **call_later()** ожидает указанное время и после этого вызывает **callback()**, выводя соответствующее сообщение.

Метод **call_later()** предоставляет удобный способ планирования выполнения задач в **asyncio**. Он позволяет управлять задержкой времени перед выполнением функции или корутины, что может быть полезно во многих сценариях, например, для установки таймера или задержки между операциями

ИТОГИ

- Если мы хотим выполнить задачу с задержкой стоит использовать **call_later**
- Если мы хотим выполнить задачу в следующем шаге **event loop** стоит использовать **call_later**