ПАРАЛЛЕЛЬНЫЙ ЗАПУСК

ВВЕДЕНИЕ

Задачи урока

01 Разобраться, как с помощью параллельного выполнения в асинхронных методах можно ускорить работу

asyncio.as_completed является функцией в библиотеке asyncio в Python, которая возвращает итератор, позволяющий ожидать завершения набора корутин или заданий и возвращать их по мере готовности

Эта функция полезна, когда вы имеете набор асинхронных операций и хотите получить результаты, как только они станут доступными, вместо ожидания завершения всех операций. Вместо того, чтобы ждать каждой операции отдельно, **asyncio.as_completed** позволяет эффективно ожидать завершения любой операции и получать результаты по мере их готовности

```
import time
from turtle import pen
import random

import asyncio
import functools

async def get_all_pages():
    await asyncio.sleep(3)
    return range(0, 3)

async def get_page_data(site, page):
    print(site, page)
    await asyncio.sleep(2)
    return random.randint(0, 20)


async def parser2(site_name):
    await asyncio.sleep(2)
    pages = await get_all_pages()
```

```
    for thread_page in asyncio.as_completed([get_page_data(site_name,
page) for page in pages]):
        print(await thread_page)

if __name__ == '__main__':
    start_time = time.time()
    jobs = []
    for site in ["site 1", "site 2", "site 3"]:
        jobs.append(asyncio.ensure_future(parser2(site)))
    event_loop = asyncio.get_event_loop()
    event_loop.run_until_complete(asyncio.gather(*jobs))
    event_loop.close()
    print("--- %s seconds ---" % (time.time() - start_time))
```

ИТОГИ

- **as_completed** используем, когда необходимо получать данные по мере выполнения асинхронных методов