

MUHAMMAD AL-XORAZMIY
NOMIDAGI TATU FARG'ONA FILIALI
FERGANA BRANCH OF TUIT
NAMED AFTER MUHAMMAD AL-KHORAZMI

“AL-FARG'ONIY AVLODLARI”

ELEKTRON ILMIY JURNALI | ELECTRONIC SCIENTIFIC JOURNAL

TA'LIMDAGI ILMIY, OMMABOP VA ILMIY TADQIQOT ISHLARI



2-SON 1(2)
2023-YIL

TATU, FARG'ONA
O'ZBEKISTON



O'ZBEKISTON RESPUBLIKASI RAQAMLI TEXNOLOGIYALAR VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI
TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
FARG'ONA FILIALI

Muassis: Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti Farg'ona filiali.

Chop etish tili: O'zbek, ingliz, rus. Jurnal texnika fanlariga ixtisoslashgan bo'lib, barcha shu sohadagi matematika, fizika, axborot texnologiyalari yo'nalishida maqolalar chop etib boradi.

Учредитель: Ферганский филиал Ташкентского университета информационных технологий имени Мухаммада ал-Хоразми.

Язык издания: узбекский, английский, русский.

Журнал специализируется на технических науках и публикует статьи в области математики, физики и информационных технологий.

Founder: Fergana branch of the Tashkent University of Information Technologies named after Muhammad al-Khorazmi.

Language of publication: Uzbek, English, Russian.

The magazine specializes in technical sciences and publishes articles in the field of mathematics, physics, and information technology.

2023 yil, Tom 1, №2
Vol.1, Iss.2, 2023 y

ELEKTRON ILMIY JURNALI

ELECTRONIC SCIENTIFIC JOURNAL

«Al-Farg'oniylar avlodlari» («The descendants of al-Fargani», «Potomki al-Fargani») O'zbekiston Respublikasi Prezidenti administratsiyasi huzuridagi Axborot va ommaviy kommunikatsiyalar agentligida 2022-yil 21 dekabrda 054493-son bilan ro'yxatdan o'tgan.

Tahririyat manzili:

151100, Farg'ona sh., Aeroport ko'chasi 17-uy, 201A-xona

Tel: (+99899) 998-01-42

e-mail: info@al-fargoniy.uz

Qo'lyozmalar taqrizlanmaydi va qaytarilmaydi.

FARG'ONA - 2023 YIL

TAHRIR HAY'ATI

Maxkamov Baxtiyor Shuxratovich,

Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti rektori, iqtisodiyot fanlari doktori, professor

Muxtarov Farrux Muhammadovich,

Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti Farg'ona filiali direktori, texnika fanlari doktori

Arjannikov Andrey Vasilevich,

Rossiya Federatsiyasi Sibir davlat universiteti professori, fizika-matematika fanlari doktori

Satibayev Abdugani Djunosovich,

Qirg'iziston Respublikasi, Osh texnologiyalari universiteti, fizika-matematika fanlari doktori, professor

Rasulov Akbarali Maxamatovich,

Axborot texnologiyalari kafedrasida professori, fizika-matematika fanlari doktori

Yakubov Maksadxon Sultaniyazovich,

TATU «Axborot texnologiyalari» kafedrasida professori, t.f.d., professor, xalqaro axborotlashtirish fanlari Akademiyasi akademigi

Bo'taboyev Muhammadjon To'ychiyevich,

Farg'ona politexnika instituti, Iqtisod fanlari doktori, professor

Abdullayev Abdujabbor,

Andijon mashinosozlik instituti, Iqtisod fanlari doktori, professor

Qo'ldashev Abbasjon Hakimovich,

O'zbekiston milliy universiteti huzuridagi Yarimo'tkazgichlar fizikasi va mikroelektronika ilmiy-tadqiqot instituti, texnika fanlari doktori, professor

Ergashev Sirojiddin Fayazovich,

Farg'ona politexnika instituti, elektronika va asbobsozlik kafedrasida professori, texnika fanlari doktori, professor

Qoraboyev Muhammadjon Qoraboievich,

Toshkent tibbiyot akademiyasi Farg'ona filiali fizika matematika fanlari doktori, professor, BMT ning maslahatchisi maqomidagi xalqaro axborotlashtirish akademiyasi akademigi

Naymanboyev Raxmonali,

TATU FF Telekommunikatsiya kafedrasida faxriy dotsenti

Polvonov Baxtiyor Zaylobiddinovich,

TATU FF Ilmiy ishlar va innovatsiyalar bo'yicha direktor o'rinbosari

Zulunov Ravshanbek Mamatovich,

TATU FF «Dasturiy injiniringi» kafedrasida dotsenti, fizika-matematika fanlari nomzodi

Saliyev Nabijon,

O'zbekiston jismoniy tarbiya va sport universiteti Farg'ona filiali dotsenti

G'ulomov Sherzod Rajaboyevich,

TATU Kiberxavfsizlik fakulteti dekani, Ph.D., dotsent

G'aniyev Abduxalil Abdujalioviyevich,

TATU Kiberxavfsizlik fakulteti, Axborot xavfsizligi kafedrasida t.f.n., dotsent

Zaynidinov Hakimjon Nasritdinovich,

TATU Kompyuter injiniringi fakulteti, Sun'iy intellekt kafedrasida texnika fanlari doktori, professor

Abdullaev Temurbek Marufovich,

Kafedra mudiri, texnika fanlar bo'yicha falsafa doktori

Bilolov Inomjon O'ktamovich,

Kafedra mudiri, pedagogika fanlar nomzodi

Daliev Baxtiyor Sirojiddinovich,

Fakultet dekani, fizika-matematika fanlari bo'yicha falsafa doktori

Zokirov Sanjar Ikromjon o'g'li,

Kafedra mudiri, fizika-matematika fanlari bo'yicha falsafa doktori

Ibroximov Nodirbek Ikromjonovich,

Dasturiy injiniring va raqamli iqtisodiyot fakulteti dekani, fizika-matematika fanlari bo'yicha PhD

Kochkorova Gulnora Dexkanbaevna,

Kafedra mudiri, falsafa fanlari nomzodi

Kadirov Abdumalik Matkarimovich,

Yoshlar masalalari va ma'naviy-ma'rifiy ishlar bo'yicha direktor o'rinbosari, falsafa fanlar bo'yicha falsafa doktori

Nurdinova Raziya Abdixalikovna,

Ilmiy tadqiqotlar, innovatsiyalar va ilmiy-pedagogik kadrlar tayyorlash bo'limi boshlig'i, texnika fanlari bo'yicha falsafa doktori

Otakulov Oybek Hamdamovich,

Kompyuter injiniringi fakulteti dekani, texnika fanlar nomzodi, dotsent

Obidova Gulmira Kuziboevna,

Kafedra mudiri, falsafa fanlari doktori

Rayimjonova Odina Sodiqovna,

Kafedra mudiri, texnika fanlari bo'yicha falsafa doktori (PhD), dotsent

Sabirov Salim Satiyevich,

Kafedra mudiri, fizika-matematika fanlari nomzodi, dotsent

Teshaboev Muhiddin Ma'rufovich,

Ta'lim sifatini nazorat qilish bo'limi boshlig'i, falsafa fanlari bo'yicha falsafa doktori

To'xtasinov Dadaxon Farxodovich,

Kafedra mudiri, pedagogika fanlari bo'yicha falsafa doktori (PhD)

Jurnal quyidagi bazalarda indekslanadi:



MUNDARIJA | ОГЛАВЛЕНИЕ | TABLE OF CONTENTS

Farrux Muxtarov, MAXSUS AXBOROT ALMASHUV KANALLARIGA BO'LADIGAN XAVF-XATARLARNI ANIQLASH, VAHOLASH VA BOSHQARISH HAMDA ULARNI BARTARAF ETISH USULLARINI ISHLAB CHIQUISH	5-8
Muhammadmullo Asrayev, 0-TARTIBLI BIR JINSLI FUNKSIONALLAR KO'RINISHIDAGI SODDA MEZONLAR UCHUN 1 INFORMATIV BELGILAR MAJMUASINI ANIQLASH USULLARI	9-12
Musoxon Dadaxonov, Muhammadmullo Asrayev, BERILGAN TASVIR SIFATINI VAHOLASH	13-16
Узоков Бархаёт Мухаммадиевич, АДАПТАЦИЯ МОДЕЛЕЙ ОПЕРАТИВНОГО УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ ПО ТЕХНИКО-ЭКОНОМИЧЕСКИМ ПОКАЗАТЕЛЯМ	17-22
Mirzakarimov Baxtiyor Abdusalomovich, Kayumov Ahror Muminjonovich, THE CHALLENGES OF TEACHING JAVA PROGRAMMING LANGUAGE IN EDUCATIONAL SYSTEMS	23-26
Якубов М.С., Хошимов Б.М., АНАЛИЗ СОВРЕМЕННЫХ МЕТОДОВ ОПРЕДЕЛЕНИЕ ПОКАЗАТЕЛЕЙ КАЧЕСТВА НЕФТЕПРОДУКТОВ	27-32
Mirzakarimov Baxtiyor Abdusalomovich, Hayitov Azizjon Mo'minjon o'g'li, THE USE OF BIOMETRIC AUTHENTICATION TECHNIQUES FOR SAFEGUARDING DATA IN COMPUTER SYSTEMS AGAINST UNAUTHORIZED ACCESS OR BREACHES	33-36
Zulunov Ravshan Mamatovich, Kayumov Ahror Muminjonovich, THE LIMITATIONS OF TEACHING JAVA PROGRAMMING LANGUAGE IN EDUCATIONAL SYSTEMS	37-40
D.X.Tojimatov, KIBER TAHDIDLARNI BASHORAT QILISH VA XAVF-XATARLARDAN NIHOYALANISHDA SUN'IY INTELEKT IMKONIYATLARIDAN FOYDALANISH	41-44
Хаджаев С.И., АСИНХРОННАЯ БИБЛИОТЕКА PYTHON ASYNCIO: ПРЕИМУЩЕСТВА И ПРИМЕРЫ ПРИМЕНЕНИЯ	45-48
Kayumov Ahror Muminjonovich, CREATING AN EXPERT SYSTEM-BASED PROGRAM TO EVALUATE TEXTILE MACHINE EFFECTIVENESS	49-52
Zulunov Ravshanbek Mamatovich, Mahmudova Muqaddasxon Abdubannob qizi, TIBBIYOT MUASSASALARIDA ELEKTRON NAVBAT TIZIMI	53-57
Зулунов Равшанбек Маматович, Гуламова Диёра Ифтихар қизи, РЕЧЕВОЙ СИГНАЛ И ЕГО НОРМАЛИЗАЦИЯ	58-60
Солиев Баҳромжон Набижоновиҷ, ГЕНЕРАЦИЯ АВТОМАТИЧЕСКОЙ ДОКУМЕНТАЦИИ API В DJANGO REST FRAMEWORK С ПРИМЕНЕНИЕМ DRF SPECTACULAR	61-66
Эрматова Зарина Кахрамоновна, АЛЬТЕРНАТИВНЫЕ ПОДХОДЫ К ОБРАБОТКЕ ОШИБОК: СРАВНЕНИЕ EXCEPTIONS И STD::EXPECTED В C++	67-73

АСИНХРОННАЯ БИБЛИОТЕКА PYTHON ASYNCIO: ПРЕИМУЩЕСТВА И ПРИМЕРЫ ПРИМЕНЕНИЯ

Хаджаев С.И.,
ассистент кафедры «Программный инжиниринг»
Ферганский филиал ТУИТ имени Мухаммада ал-Хорезми

Аннотация: Статья описывает асинхронную библиотеку Python asyncio и ее использование в асинхронном программировании. Рассматриваются основные принципы работы asyncio, включая корутины, сопрограммы и событийный цикл. Обсуждаются преимущества использования asyncio в сетевых приложениях и операциях ввода-вывода. Также рассматриваются ограничения asyncio и возможные альтернативы, такие как многопоточность и параллельное программирование. В целом статья предоставляет обзор асинхронного программирования в Python и может быть полезна для разработчиков, работающих с масштабируемыми приложениями.

Ключевые слова: Python, asyncio, асинхронное программирование, сетевое программирование, многопоточность, параллельное программирование, ввод-вывод, корутины, сопрограммы, событийный цикл, масштабируемость

Введение. Python - один из самых популярных языков программирования в мире благодаря своей простоте, удобству использования и богатой экосистеме библиотек. Однако, в некоторых случаях, Python может быть не самым быстродействующим языком, особенно если мы имеем дело с блокирующими операциями, такими как чтение/запись данных из/в базу данных или сетевые операции.

Для улучшения производительности и скорости выполнения задач, связанных с вводом/выводом, в Python была разработана асинхронная библиотека asyncio.

В настоящее время асинхронное программирование является важным инструментом для разработки современных сетевых приложений и операций ввода-вывода. В Python для этой цели используется библиотека asyncio, которая предоставляет возможности для создания асинхронных сетевых приложений и операций ввода-вывода.

Целью данной статьи является представление методологии асинхронного программирования с использованием библиотеки asyncio в Python. В статье будут рассмотрены основные принципы асинхронного программирования, а также приведены примеры использования asyncio для создания сетевых приложений и операций ввода-вывода.

В целом, понимание основ асинхронного программирования и его использование в Python может помочь разработчикам создавать более эффективные и масштабируемые сетевые приложения и операции ввода-вывода. Кроме того, в статье будет рассмотрен литературный обзор, который поможет читателям найти дополнительную информацию и ресурсы для изучения асинхронного программирования и asyncio в Python.

Методология и литературный обзор. Асинхронное программирование основывается на идее, что несколько задач могут выполняться одновременно, без ожидания окончания предыдущей. В традиционном синхронном подходе, операции выполняются последовательно, то есть каждая операция начинается только после того, как предыдущая закончена.

В асинхронном программировании, задачи разбиваются на небольшие подзадачи, которые могут выполняться параллельно. Когда одна подзадача завершается, другая начинает выполняться. Это позволяет увеличить производительность приложения и снизить время ответа.

Библиотека asyncio предоставляет инструменты для асинхронного программирования, включая event loop, coroutine и Future.[1]

Event loop - это основной компонент asyncio. Он управляет запуском и остановкой корутин, а также координирует выполнение их задач.

Coroutine - это функция, которая может приостанавливаться во время выполнения, чтобы дать другим задачам возможность выполнения.

Future - это объект, который представляет результат асинхронной операции. Он позволяет выполнить операцию асинхронно, а затем получить результат, когда операция завершится.

Для написания статьи о асинхронной библиотеке Python asyncio была использована методология литературного обзора. В рамках этого подхода был проведен анализ доступных ресурсов, связанных с темой асинхронного программирования в Python, в том числе книг, статей, видео-лекций и онлайн-курсов. Были выявлены основные темы и проблемы, связанные с использованием asyncio, а также сравнены достоинства и недостатки этой технологии в сравнении с другими подходами к асинхронному программированию.

В результате литературного обзора были выявлены следующие ключевые выводы:

asyncio - это мощная библиотека Python для асинхронного программирования, которая обеспечивает высокую производительность при работе с сетевыми приложениями и операциями ввода-вывода.

Одним из главных преимуществ asyncio является возможность использования корутин, которые позволяют создавать легковесные процессы, работающие параллельно в рамках одного потока.

Кроме того, asyncio предоставляет удобный механизм для управления событиями, основанный на событийном цикле, что позволяет эффективно использовать системные ресурсы.[2]

Несмотря на все преимущества, использование asyncio может ограничиваться в случаях, когда требуется выполнение сложных математических вычислений или операций, связанных с большим объемом данных.

Для этих целей могут быть использованы альтернативные подходы, такие как многопоточность или параллельное программирование.

Среди наиболее полезных ресурсов для литературного обзора следует отметить книгу "Python Asyncio: A Guide to Asynchronous

Programming in Python" (Dmitry Rodionov), бесплатный онлайн-курс "Asyncio: A Primer on Asynchronous Programming in Python" (Real Python) и официальную документацию по asyncio на сайте Python.org. Кроме того, стоит упомянуть видео-лекции "Asyncio — How the hell does it work?" (Nick Coghlan) и "Python Concurrency From the Ground Up" (David Beazley), которые предоставляют подробное объяснение основ работы с asyncio и другими подходами к асинхронному программированию в Python.

Другие полезные источники включают статьи "Async IO in Python: A Complete Walkthrough" (Real Python), "Understanding Python's Asyncio by Example" (Geir Arne Hjelle) и "Exploring asyncio with examples" (Miguel Grinberg), которые предлагают различные примеры использования asyncio в реальных приложениях.

Также стоит упомянуть книгу "High Performance Python: Practical Performant Programming for Humans" (Micha Gorelick и Ian Ozsvald), которая не только охватывает тему асинхронного программирования, но также предоставляет ценные советы по оптимизации производительности Python-приложений.

В целом, литературный обзор показал, что asyncio является мощным инструментом для асинхронного программирования в Python, который позволяет создавать эффективные и масштабируемые сетевые приложения и операции ввода-вывода. Однако, как и при использовании любой другой технологии, необходимо учитывать ее особенности и ограничения, чтобы выбирать подходящие инструменты для каждой конкретной задачи.[3]

Результаты.

Асинхронное программирование может быть использовано для ускорения выполнения задач, связанных с вводом/выводом. Недостатки асинхронного программирования с использованием библиотеки asyncio включают в себя следующие:

Сложность отладки: отладка асинхронного кода может быть сложной, особенно если используются сложные конструкции асинхронного программирования, такие как корутины и футуры.

Ограниченная поддержка некоторыми библиотеками: не все библиотеки поддерживают асинхронное программирование. Это может создать проблемы при интеграции существующего кода с асинхронной библиотекой asyncio.[4]

Потребление памяти: асинхронные приложения могут потреблять больше памяти, чем синхронные, из-за того, что каждая корутина требует своей собственной стековой памяти.

Сложность работы с блокирующими операциями: хотя асинхронное программирование позволяет выполнять операции асинхронно, некоторые операции все равно могут быть блокирующими, и при их выполнении поток будет останавливаться.

В целом, библиотека `asyncio` является мощным инструментом для разработки асинхронных приложений на Python. Однако, разработчикам необходимо учитывать как преимущества, так и недостатки данного подхода, чтобы выбрать наиболее подходящий для своего проекта метод программирования.

Примеры применения `asyncio` включают в себя:

Сетевые операции: приложения, которые работают с сетью, могут использовать `asyncio` для выполнения сетевых запросов асинхронно. Это позволяет увеличить скорость работы приложения и снизить время ожидания ответа.

Базы данных: `asyncio` может использоваться для выполнения операций чтения/записи в базу данных асинхронно. Это позволяет увеличить производительность приложения при работе с большими объемами данных.

Веб-серверы: `asyncio` может использоваться для создания асинхронных веб-серверов. Это позволяет обрабатывать большое количество запросов одновременно, увеличивая производительность приложения.[5]

Микросервисы: `asyncio` может быть использован для создания микросервисов, которые могут обрабатывать запросы асинхронно. Это позволяет создавать более масштабируемые и производительные приложения.

Обсуждение. Асинхронное программирование имеет свои преимущества и недостатки. Одним из основных преимуществ является увеличение производительности приложения и снижение времени ответа. Также асинхронное программирование позволяет лучше использовать ресурсы процессора. Однако, асинхронное программирование может быть сложным в использовании и требует от разработчиков определенного уровня опыта и знаний. Кроме того, не все операции можно

выполнять асинхронно, например, если операция не является блокирующей. Библиотека `asyncio` в Python представляет собой мощный инструмент для асинхронного программирования. Она позволяет разрабатывать асинхронный код, который может работать с сетевыми приложениями, базами данных, операциями ввода-вывода и другими асинхронными задачами.

Однако, `asyncio` не является универсальным решением для всех сценариев асинхронного программирования и может не подходить для некоторых приложений. Кроме того, она имеет свои собственные особенности и требует от разработчиков понимания асинхронного программирования и ее концепций.

Один из главных недостатков `asyncio` заключается в том, что ее использование может быть сложным для новичков в программировании на Python. В особенности, многие разработчики могут испытывать трудности при работе с `Event Loop` и `Coroutines`, которые являются ключевыми концепциями `asyncio`.

Несмотря на это, библиотека `asyncio` продолжает быстро развиваться и улучшаться, что делает ее все более доступной и полезной для разработчиков. Она позволяет существенно улучшить производительность асинхронного кода и обеспечивает удобство использования в контексте современных асинхронных приложений.

В целом, использование библиотеки `asyncio` может быть очень полезным для разработчиков, которые хотят создавать эффективный и масштабируемый асинхронный код на Python. Однако, как и при любом использовании технологии, необходимо понимать ее особенности и использовать ее в соответствии с требованиями проекта.

Заключение. Асинхронное программирование с использованием библиотеки `asyncio` может значительно улучшить производительность приложения, особенно при работе с задачами, связанными с вводом/выводом. Однако, перед тем как использовать асинхронное программирование, необходимо убедиться, что оно подходит для конкретного проекта и что разработчики имеют достаточный уровень знаний и опыта в использовании этой технологии. В заключение, асинхронная библиотека Python `asyncio` предоставляет разработчикам возможность создавать высокопроизводительные приложения,

обеспечивая масштабируемость и эффективность взаимодействия с внешними сервисами и ресурсами. Благодаря простоте использования и многофункциональности библиотеки, многие компании выбирают Python и asyncio для разработки своих продуктов.

Однако, при использовании asyncio необходимо учитывать недостатки, такие как сложность отладки, ограниченная поддержка некоторыми библиотеками, потребление памяти и сложность работы с блокирующими операциями. В целом, при правильном использовании и соблюдении рекомендаций по написанию асинхронного кода, библиотека asyncio может быть очень полезной и эффективной для разработки масштабируемых и производительных приложений на Python. Кроме того, следует отметить, что asyncio не является панацеей для всех задач, связанных с асинхронным программированием. В некоторых случаях, особенно при работе с большими объемами данных или при необходимости выполнения сложных операций, может быть более эффективным использование многопоточности или многопроцессорности.

Также следует учитывать, что использование asyncio требует некоторой предварительной подготовки и обучения, особенно для разработчиков, не знакомых с асинхронным программированием. В этом случае может потребоваться дополнительное время на изучение документации и примеров кода. Несмотря на некоторые ограничения и сложности, асинхронная библиотека Python asyncio является мощным инструментом для разработки высокопроизводительных и масштабируемых приложений. Она может быть особенно полезной в приложениях, связанных с сетевыми взаимодействиями, базами данных и обработкой больших объемов данных.

"Asyncio — How the hell does it work?" (Nick Coghlan) - это видео-лекция, которая подробно объясняет основы работы asyncio и дает обзор примеров кода. Доступно на YouTube.

Литература:

[1] Bjørndalen J. M., Vinter B., Anshus O. aPyCSP–Asynchronous PyCSP Using Python Coroutines and Asyncio //Communicating Process

Architectures 2017 & 2018. – IOS Press, 2019. – С. 281-297.

[2] Hattingh C. Using Asyncio in Python: understanding Python's asynchronous programming features. – "O'Reilly Media, Inc.", 2020.

[3] Савостин П. А., Ефремова Н. Э. Практическое применение асинхронного программирования на языке Python при помощи пакета asyncio //Программные системы и вычислительные методы. – 2018. – №. 2. – С. 11-16.

[4] "Python Asyncio: A Guide to Asynchronous Programming in Python" by Dmitry Rodionov

[5] "Asyncio: A Primer on Asynchronous Programming in Python" by Real Python "Exploring asyncio with examples" by Miguel Grinberg