УНИВЕРСИТЕТ СИНЕРГИЯ

ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ

Продвинутый Python

Андрей Вильмов

СОДЕРЖАНИЕ ТЕМЫ

01 Что такое потоки

02 Как работать с классом Thread

03 Как управлять потоками используя ThreadPoolExecutor

ПОТОКИ

В программировании представляют собой независимые последовательности инструкций, которые могут выполняться параллельно внутри одного процесса

Могут совместно использовать ресурсы процесса, такие как память и файлы, но при этом должны быть организованы синхронизацией, чтобы избежать конфликтов доступа

Позволяют распараллелить выполнение задач и увеличить производительность программы

Имеют собственные стек вызовов, регистры и счетчик команд, что позволяет выполнять различные операции одновременно

THREADING

Для работы с **потоками** в Python используется модуль **threading**

Для создания потока используется класс **Thread()**

СОЗДАНИЕ И ЗАПУСК ПОТОКА


```
...
Thread(target=parser,args=("Site 1",))
```

Определение понятия

THREADING

Для запуска потока используют метод **start()**

Метод join()

Блокирует выполнение главного потока до тех пор, пока указанный поток не завершит свою работу. Это означает, что строка кода "Программа завершила выполнение" не будет выполнена до того, как поток завершит свою работу

Может принимать необязательный аргумент timeout, который указывает максимальное время ожидания завершения потока. Если время ожидания истекает, главный поток продолжит выполнение без ожидания завершения потока

Очень полезен, когда нужно выполнить определенные действия после завершения фоновых задач в потоках

JOIN


```
...
my_thread = Thread(target=my_func, args=(“str”,))
my_thread.start()
my_thread.join()
```

LOCK

В Python модуль `threading` предоставляет класс **`Lock`**, который представляет примитив синхронизации в виде блокировки

Lock (блокировка) позволяет управлять доступом к общим ресурсам из нескольких потоков

Lock работает по принципу "захватить" и "освободить". Когда поток хочет получить доступ к общему ресурсу, он "захватывает" блокировку. Если блокировка уже захвачена другим потоком, текущий поток будет ожидать, пока блокировка не будет освобождена. После того, как текущий поток закончит работу с общим ресурсом, он "освобождает" блокировку, чтобы другие потоки могли ее захватить

RLOCK (РЕКУРСИВНАЯ БЛОКИРОВКА)

Это расширение блокировки (Lock) в модуле threading в Python

Позволяет потоку захватить блокировку несколько раз без блокировки самого себя, в отличие от обычной блокировки. Это означает, что тот же самый поток может получить блокировку несколько раз, прежде чем должен будет ее освободить

Поддержка рекурсии является основным отличием от Lock. Когда поток захватывает RLock, он увеличивает счетчик блокировки. Каждый раз, когда поток снова захватывает RLock, этот счетчик увеличивается. Поток должен освободить RLock столько же раз, сколько он его захватил, чтобы другие потоки могли получить доступ к ресурсу

THREADPOOLEXECUTOR

Это класс, предоставляемый модулем concurrent.futures в Python, который представляет пул потоков для эффективного выполнения асинхронных операций

ThreadPoolExecutor предоставляет простой и удобный интерфейс для параллельного выполнения задач в фоновых потоках

ОСНОВНЫЕ ОСОБЕННОСТИ THREADPOOLEXECUTOR

01 **Создание пула потоков**

Вы можете создать экземпляр ThreadPoolExecutor указав количество потоков, которые вы хотите использовать, или использовать значение по умолчанию, которое равно количеству ядер процессора

02 **Планирование задач**

ThreadPoolExecutor предоставляет метод submit(), который используется для планирования выполнения задачи в пуле потоков. Метод submit() возвращает объект Future, который представляет результат выполнения задачи или ее текущий статус.

03 **Управление завершением потоков**

ThreadPoolExecutor предоставляет метод submit(), который используется для планирования выполнения задачи в пуле потоков. Метод submit() возвращает объект Future, который представляет результат выполнения задачи или ее текущий статус.

ПОДВЕДЁМ ИТОГИ

- Параллельное программирование запускает выполнение кода в отдельном потоке
- ThreadPoolExecuter - ограничивает кол-во запущенных одновременно потоков
- Для доступа к одному и тому же ресурсу из разных потоков стоит использовать блокировки