Контроль 04.04.2020 программирование

Корнеенков К.Д., ФЕ-81

Вопрос №1: Многопоточные вычисления в Python. Процесс-ориентированные решения (библиотека multiprocessing).

Многопоточность - это основная концепция программирования, которую поддерживают почти все языки программирования высокого уровня.

Потоки управления (threads) образуются и работают в рамках одного процесса. В однопоточном приложении (программе, которая не использует дополнительных потоков) имеется только один поток управления. Говоря упрощенно, при запуске программы этот поток последовательно исполняет встречаемые в программе операторы, направляясь по одной из альтернативных ветвей оператора выбора, проходит через тело цикла нужное число раз, выбирается к месту обработки исключения при возбуждении исключения. В любой момент времени интерпретатор Python знает, какую команду исполнить следующей. После исполнения команды становится известно, какой команде передать управление. Эта ниточка непрерывна в ходе выполнения программы и обрывается только по ее завершении.

Поддержка многопоточности в языке Python доступна через использование ряда модулей. В стандартном модуле threading определены нужные для разработки многопоточной (multithreading) программы классы: несколько видов семафоров (классы замков Lock, RLock и класс Semaphore ) и другие механизмы взаимодействия между потоками (классы Event и Condition ), класс Timer для запуска функции по прошествии некоторого времени. Модуль Queue реализует очередь, которой могут пользоваться сразу несколько потоков. Для создания и (низкоуровневого) управления потоками в стандартном модуле thread определен класс Thread.

*Процесс-ориентированные решения (библиотека multiprocessing):*

multiprocessing позволяет работать с процессам как с потоками. Это значит, что модуль берет на себя проблему синхронизации отдельных Python-процессов. Много процессов - много GIL'ов (каждый в своем процессе) - нет проблем с использованием процессора.

Вопрос №2: Обработки асинхронных событий в GUI (примеры).

Основной концепцией асинхронного ввода-вывода является цикл обработки событий. В программе может быть несколько циклов событий. Каждый поток будет иметь не более одного активного цикла обработки событий. Цикл обработки событий предоставляет следующие возможности:

Регистрация, выполнение и отмена отложенных вызовов (с задержками).

Создание клиентских и серверных транспортов для различных видов связи.

Запуск подпроцессов и связанных транспортов для связи с внешней программой.

Делегирование ресурсозатратных вызовов функций в пул потоков.

import asyncio

async def foo(delay):

    for i in range(10):

        print(i)

        await asyncio.sleep(delay)

def stopper(loop):

    loop.stop()

loop = asyncio.get\_event\_loop()

# Schedule a call to foo()

loop.create\_task(foo(0.5))

loop.create\_task(foo(1))

loop.call\_later(12, stopper, loop)

# Block until loop.stop() is called()

loop.run\_forever()

loop.close()