**Тема занятия №20: Многопоточное, асинхронное и мультипроцессорное программирование**

**1. Multiprocessing**

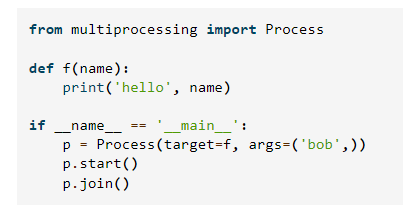
**Multiprocessing** — это пакет, поддерживающий порождение процессов с использованием API, аналогичный модулю threading. Пакет multiprocessing предлагает как локальный, так и удаленный параллелизм, эффективно обходя Глобальную блокировку интерпретатора за счёт использования подпроцессов вместо потоков. Благодаря этому, модуль multiprocessing позволяет программисту полностью использовать несколько процессоров на компьютере. Он работает как в Unix, так и в Windows.

В модуле multiprocessing также представлены API, не имеющие аналогов в модуле threading. Ярким примером этого является объект Pool, который предлагает удобные средства распараллеливания выполнения функции по нескольким входным значениям, распределяя входные данные по процессам (параллелизм данных). В следующем примере демонстрируется обычная практика определения таких функций в модуле, чтобы дочерние процессы могли успешно импортировать этот модуль. Это базовый пример параллелизма данных с использованием Pool:



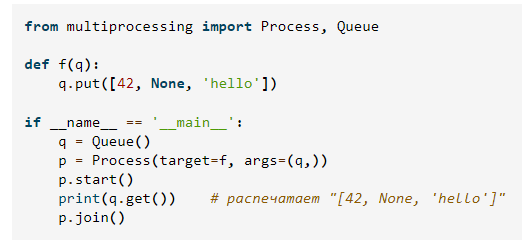
**Класс Process**

В multiprocessing процессы порождаются путём создания объекта Process и последующего вызова его метода start(). Process следует API threading.Thread. Тривиальный пример многопроцессорной программы:



**Очереди**

Класс Queue является почти клоном queue.Queue. Например:



**Конвейеры (pipes)**

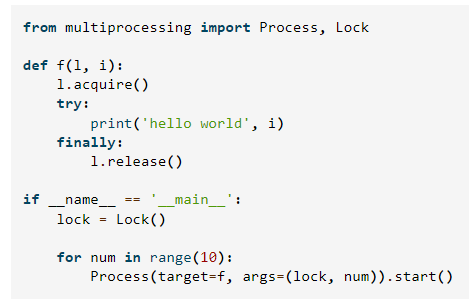
Функция Pipe() возвращает пару объектов соединения, соединенных конвейером (pipe), который по умолчанию является дуплексным (двусторонним). Например:



Два объекта подключения, возвращаемые Pipe(), представляют собой два конца конвейера. Каждый объект подключения содержит методы send() и recv() (среди прочих). Обратите внимание, что данные в конвейере могут быть повреждены, если два процесса (или потока) пытаются читать или писать в один и тот же конец конвейера одновременно. Конечно, нет риска повреждения из-за процессов, использующих разные концы конвейера одновременно.

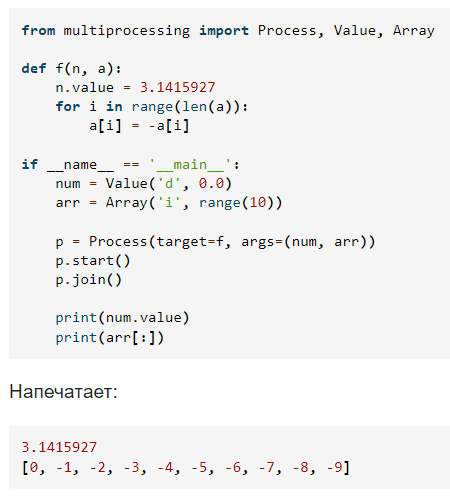
**Синхронизация процессов**

multiprocessing содержит эквиваленты всех примитивов синхронизации из threading. Например, можно использовать блокировку, чтобы гарантировать, что только один процесс печатает в стандартный вывод за раз:



**Общая память**

Данные могут храниться в отображении общей памяти с использованием Value или Array. Например, следующий код



**Использование пула рабочих¶**

Класс Pool представляет собой пул рабочих процессов. У него есть методы, которые позволяют передавать задачи рабочим процессам несколькими способами.

Например:

