Есть два типа задач: работа с процессором, работа с IU – input/output.

Ко второй категории относится работа с файлами, с сетями и проч.

Именно для второго типа и требуется ассинхронность. Ассинхронность отличается от многопроцессороности и многопотоковости.

Последовательное выполнение команд не подходит для работы с сетью. Так как при работе с одним клиентом для остальных порт будет заблокирован.

Здесь нам и требуется асинхронное выполнение.

Методы для ассинхронности:

1. Loop’ы c select

С использованием команды select. Он принимает 3 списка: список для проверки на доступность к чтению, список для проверки на доступность к записи, список для проверки на ошибки. Мы используем только для чтения.

Таким образом, блокирующая операция у нас будет только select, функции для приема связи и отправки ответа будут без циклов выплнятся моментально и ждать следующего вызова. А селект будет запускать тот файл, в котором произошли изменения в доступности для чтения.

1. Selectors

Способ весьма схожий с предыдущим примером. Здесь мы создаем переменную класса селекторс и приравниваем его значению DefaultSelector. В эту переменную при регистрации задается объект – сервер\_сокет, евент – доступность к чтению, - дата – любые данные сопутствующие объекту, в нашем случае это функция. Далее в луп принцип работы такой же, как в селект: выполняется выборка последнего изменения, затем вызывается элемент дата нашего селектора, то есть привязанная к объекту функция.

1. Генераторы – функции, которые можно поставить на паузу, не доводя до конца, а позднее продолжить их выполнение. С помощью next выполнение функции сдвигается до следующего yield.

Здесь также стоит упомянуть о принципе работы RoundRobbin:

Мы задаем несколько генераторов в один список. При каждой итерации мы выполняем next нулевого генератора и перемещаем это нулевой генератор в конец списка.

Алгоритм Дэвида Бизли «Конкурентность питона»:

Здесь мы будем использовать генераторы вместо обычных функций, т.е. yield. Возвращать мы будем кортежи из двух элементов, где второй элемент — это сокет, а первый элемент описывает read или write, то есть что предполагается делать с этим сокетом. С помощью метода select мы теперь слушаем готовность и к чтению, и к записи.

Снова используем наш loop, а в генераторах бесконечные циклы убираем.

Селект, получив измененное состояние одного из элементов для рид или райт заносит соответствующий кортеж в предварительно созданный пустой список tasks.

Далее в зависимости от первого элемента занесенного кортежа read/write, заносим в соответствующий словарь to\_read или to\_write сокет и его объект генератора.