**7. Реализация сокетов в языке Python. Модуль socket.**

Сокет — название программного интерфейса для обеспечения обмена данными между процессами. Существуют клиентские и серверные сокеты. Серверный сокет прослушивает определенный порт, а клиентский подключается к серверу. После того, как было установлено соединение начинается обмен данными. Сокет на языке Python позаимствован из языка программирования C и операционной системы Unix. Для работы с сокетами нужен клиент(ы) и сервер.

В Python для работы с сокетами используется модуль socket:

1. Подключение библиотеки для работы с сокетами. import socket
2. Создание сокета: sock = socket.socket()
3. Определяемся с хостом и портом сервера sock.bind(‘localhost’,6789)
4. Указываем количество клиентов, которые будет слушать наш сервер. Sock.listen(5)
5. Client, addr = sock.accept()

* Sock.recv – получает данные
* Sock.send – отправляет данные.

Мы можем построить наши сервера как по udp, так и по tcp протоколу.

**UDP** отправляет сообщения, но их размер ограничен и не гарантируется, что они достигнут места назначения. UDP не гарантирует в каком порядке придут сообщения. **TCP** вместо сообщений отправляет потоки байтов. Нельзя сказать, сколько байтов отправит или получит система с каждым вызовом. TCP доставляет данные в том порядке, в котором они были отправлены.

**7. Понятие программного потока. Процессы и потоки.**

**Программный поток** – поток в котором выполняются задачи программы. Все они выполняются последовательно. С появлением многоядерных процессоров стала общеупотребительной практика распространять нагрузку на все доступные ядра. Существует два основных подхода в распределении нагрузки: использование процессов и потоков. Использование нескольких процессов фактически означает использование нескольких программ, которые выполняются независимо друг от друга.

В Python за это отвечают модулю subprocessing и multiprocessing.

Import multiprocessing

Import os

def foo(n):

print(n, os.getpid())

If \_\_name\_\_ “\_\_main\_\_”:

For n in range(4):

multiprocessing.Process(target=poo, args = n).start()

For I in range(4):

Для увеличения скорости работы программы используются потоки и процессы. Если нужно, чтобы ваше приложение выполняло несколько задач в одно и то же время, то можете воспользоваться потоками (threads). **Потоки** позволяют приложениям выполнять в одно и то же время множество задач. Многопоточность (multi-threading) важна во множестве приложений, от примитивных серверов до современных сложных и ресурсоёмких игр. За многопоточность отвечает модуль Threading. Создание отдельного потока:

import Threading

def foo(n):

print(n)

threading.thread(target=foo, args=10).start