ЛАБОРАТОРНАЯ РАБОТА № 8
КЛИЕНТ-СЕРВЕРНОЕ ПРОГРАММИРОВАНИЕ В PYTHON

Время выполнения – 8 часов.

Цель работы: научиться разрабатывать клиент-серверные приложения.

Задачи работы

Ознакомиться этапами разработки клиент-серверных приложений;

Перечень обеспечивающих средств

Для выполнения работы необходимо иметь компьютер с установленной операционной системой семейства Windows, установленным python и IDE PyCharm Professional.

Общие теоретические сведения

Чтобы понять программирование сокетов Python, нам нужно знать о трех интересных темах – Socket Server, Socket Client и Socket. Итак, что такое сервер? Сервер – это программное обеспечение, которое ожидает запросов клиентов и обслуживает или обрабатывает их соответственно. С другой стороны, клиент запрашивает эту услугу. Клиентская программа запрашивает некоторые ресурсы к серверу, и сервер отвечает на этот запрос. Socket – это конечная точка двунаправленного канала связи между сервером и клиентом. Сокеты могут обмениваться данными внутри процесса, между процессами на одной машине или между процессами на разных машинах. Для любого взаимодействия с удаленной программой мы должны подключаться через порт сокета. Основная цель этого руководства по программированию сокетов

– познакомить вас с тем, как сервер сокетов и клиент взаимодействуют друг с другом. Вы также узнаете, как написать программу сервера сокетов в Python.

Задание

Разработать серверную и клиентскую модель, для обмена сообщениями. Программа сервера сокетов запускается сначала и ждет любого запроса. Клиентская программа сначала инициирует диалог. Затем серверная программа будет реагировать на запросы клиента соответственно. Клиентская программа будет завершена, если пользователь введет сообщение «до свидания». Серверная программа также завершится, когда завершится клиентская программа, это необязательно, и мы можем поддерживать выполнение серверной программы на неопределенный срок или завершить работу с помощью какой-либо конкретной команды в клиентском запросе.

Технология выполнения работы

Сервер сокетов

Мы сохраним программу сервера сокетов, как socket_server.py. Чтобы использовать соединение, нам нужно импортировать модуль сокета. Затем последовательно нам нужно выполнить некоторую задачу, чтобы установить соединение между сервером и клиентом. Мы можем получить адрес хоста с помощью функции socket.gethostname(). Рекомендуется использовать адрес порта пользователя выше 1024, поскольку номер порта меньше 1024 зарезервирован для стандартного интернет-протокола. Смотрите приведенный ниже пример кода сервера:

```
import socket


def server_program():
    # get the hostname
    host = socket.gethostname()
    port = 5000  # initiate port no above 1024

    server_socket = socket.socket()  # get instance
    # look closely. The bind() function takes tuple as argument
    server_socket.bind((host, port))  # bind host address and port together

    # configure how many client the server can listen simultaneously
    server_socket.listen(2)
    conn, address = server_socket.accept()  # accept new connection
    print("Connection from: " + str(address))
    while True:
        # receive data stream. it won't accept data packet greater than 1024 byte
s
        data = conn.recv(1024).decode()
        if not data:
            # if data is not received break
            break
        print("from connected user: " + str(data))
        data = input(' -> ')
        conn.send(data.encode())  # send data to the client

    conn.close()  # close the connection


if __name__ == '__main__':
    server_program()
```

Итак, наш сервер сокетов работает на порту 5000 и будет ждать запроса клиента. Если вы хотите, чтобы сервер не завершал работу при закрытии клиентского соединения, просто удалите условие if и оператор break. Цикл while используется для бесконечного запуска серверной программы и ожидания клиентского запроса.

Клиент сокета

Мы сохраним клиентскую программу сокета python как socket_client.py. Эта программа похожа на серверную, за исключением привязки. Основное различие между серверной и клиентской программой состоит в том, что в

серверной программе необходимо связать адрес хоста и адрес порта вместе. Смотрите ниже пример кода клиента сокета:

```
import socket


def client_program():
    host = socket.gethostname()  # as both code is running on same pc
    port = 5000  # socket server port number

    client_socket = socket.socket()  # instantiate
    client_socket.connect((host, port))  # connect to the server

    message = input(" -> ")  # take input

    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())  # send message
        data = client_socket.recv(1024).decode()  # receive response

        print('Received from server: ' + data)  # show in terminal

        message = input(" -> ")  # again take input

    client_socket.close()  # close the connection


if __name__ == '__main__':
    client_program()
```

Чтобы увидеть результат, сначала запустите программу сервера сокетов. Затем запустите клиентскую программу. После этого напишите что-нибудь из клиентской программы. Затем снова напишите ответ от серверной программы. Наконец, напишите «до свидания» из клиентской программы, чтобы завершить обе программы. Ниже короткое видео покажет, как это работало на моем тестовом прогоне примеров программ сервера сокетов и клиента.

Обратите внимание, что сервер сокетов работает на порту 5000, но клиенту также требуется порт сокета для подключения к серверу. Этот порт назначается случайным образом при вызове клиентского соединения. В данном случае это 57822.

Вопросы и задания для защиты работы

1. Назовите этапы разработки клиент-серверного приложения?
2. Основные особенности клиент-серверной архитектуры?
3. Что такое сокет?
4. Что такое сервер?
5. Что такое порт?
6. Что такое клиент?