Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики.

Факультет инфокоммуникационных технологий.

Лабораторная работа №1 по теме: «Работа с сокетами» по дисциплине: Web-программирование.

Выполнила: Тихонова Е.К.
Группа: К33401

Преподаватель: Говоров Антон Игоревич

Санкт-Петербург, 2020

1. Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

Содержимое файла client.py:

```
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    conn.connect(("127.0.0.1", 14900))
    conn.send(b"Hello, server\n")
    data = conn.recv(16384)
    udata = data.decode("utf-8")
    print(udata)
except ConnectionResetError as e:
    print('Сервер разорвал соединение, ваше сообщение не было получено')
finally:
    conn.close()
```

По второй строке можно видеть, что клиент и сервер взаимодействуют по tcp - socket.SOCK_STREAM. Обработка ConnectionResetError здесь только затем, чтобы сделать вывод файла более удобочитаемым. Ошибка появляется, потому что в данном случае разрывает соединение сервер: conn.close()

Содержимое файла server.py:

```
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("127.0.0.1", 14900))
conn.listen(10)

while True:
    try:
        clientsocket, address = conn.accept()
        data = clientsocket.recv(16384)
        udata = data.decode("utf-8")
        print("Data:", udata)
        clientsocket.send(b"Hello, client\n")
    except KeyboardInterrupt:
        conn.close()
        break
```

conn.bind(("127.0.0.1", 14900)) привязывает сокет к хосту и порту. К ним же затем клиент и подключается.

2. Реализовать клиентскую и серверную часть приложения. Клиент запрашивает у сервера выполнение математической операции, параметры, которые вводятся с клавиатуры. Сервер обрабатывает полученные данные и возвращает результат клиенту.
   Вариант (16):
   d. Поиск площади параллелограмма.

Содержимое файла client.py:

```
import socket
```

```
import json

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

conn.connect(("127.0.0.1", 14900))
print(conn.recv(16384).decode("utf-8"))
a = float(input('Сторона параллелограмма: '))
a_h = float(input('Высота опущенная на эту сторону: '))
b = float(input('Другая сторона параллелограмма: '))
b_h = float(input('Высота опущенная на эту сторону: '))
alpha = float(input('Угол между сторонами: '))

paral_info = json.dumps(
    {'a': a, 'b': b, 'a_h': a_h, 'b_h': b_h, 'alpha': alpha})
conn.send(paral_info.encode("utf-8"))
print(conn.recv(16384).decode("utf-8"))
conn.close()
```

Кроме сокетов в данном задании используется json – для более удобной обработки переданных на сервер данных. После подключения к серверу клиент получает все необходимые данные для нахождения поиска площади параллелограмма, после чего отправляет их на сервер, упакованные в json формат. Затем выводит результат, который получает от сервера.

Содердимое файла server.py:

```
import socket
import json
from math import sin

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("127.0.0.1", 14900))
conn.listen(10)

clientsocket, address = conn.accept()
clientsocket.send(b"""Hello, client
    \rPlease tell me everything you know about your parallelogram
    \r0 for field you don't know""")
data = clientsocket.recv(16384).decode("utf-8")
paral_info = json.loads(data)
result = [0,0,0]
result[0] = paral_info['a']*paral_info['a_h']
result[1] = paral_info['b']*paral_info['b_h']
result[2] = paral_info['a']*paral_info['b']*sin(paral_info['alpha'])
for res in result:
    if res != 0:
        clientsocket.send(("Площадь параллелограмма =
"+str(res)).encode("utf-8"))
        conn.close()
try:
    clientsocket.send("Данных недостаточно, чтобы посчитать площадь
параллелограмма".encode("utf-8"))
except Error as e:
    pass
finally:
    conn.close()
```

Кроме пакетов, о которых было сказано ранее, сервер импортирует функцию sin() из пакета math. Так как она нужна ему для вычисления. Логика вычислений очень проста – есть две формулы

(высота на сторону и две стороны на синус) и три варианта поиска этого решения (два для первой формулы, так как спрашивается две стороны и две высоты). Если какие-то данные не известны, клиент отправляет вместо них 0. В том случае, если какой-то из формул данных достаточно, результат будет != 0. Он и будет отправлен клиенту. Иначе будет отправлено сообщение, говорящее о том, что данных недостотчно.

3. Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

Содержимое файла index.html:

```
<html>
    <head>
        <title>HTML PAGE!</title>
    </head>
    <body>
        <h1>Hello</h1>
        <p>Nice to see you on this page!</p>
    </body>
</html>
```

Здесь заголовок и абзац.

Содержимое файла client.py:

```
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:

    s.connect(('localhost', 8080))
    s.send(b"GET / HTTP/1.1\r\nHost: localhost:8080\r\nAccept:
text/html\r\n\r\n")

    while True:
        data = s.recv(1024)
        if not data:
            break
        print(data.decode())
```

Разница с предыдущими заданиями в основном в том, что здесь используется with. Так же, вместо указания 127.0.0.1, было записано ‘localhost’, что есть одно и то же. Сперва от клиента серверу отправляется запрос. Далее клиент получает всю информацию от сервера и завершает свою работу.

Содержимое файла server.py:

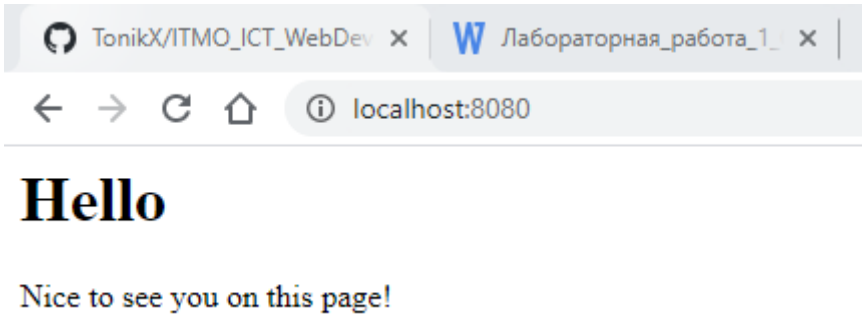
```
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as conn:
    conn.bind(('localhost', 8080))
    conn.listen(10)

    clientsocket, address = conn.accept()

    data = clientsocket.recv(1024)
    print(data.decode())

    header = 'HTTP/1.1 200 OK\n'
```

```
    header += 'Content-Type: '+'text/html' + '\n\n'
    header = header.encode("utf-8")

    with open('index.html', 'rb') as index:
        response = index.read()
    clientsocket.sendall(header+response)
```

Сервер принимает входящее соединение и отправляет страницу клиенту. Если запустить клиент в консоли вывод будет следующий:

```
$ python client.py
HTTP/1.1 200 OK
Content-Type: text/html

<html>
    <head>
        <title>HTML PAGE!</title>
    </head>
    <body>
        <h1>Hello</h1>
        <p>Nice to see you on this page!</p>
    </body>
</html>
```

Если же после запуска сервера, открыть http://localhost:8080/ в браузере, то можно будет увидеть страницу



4. Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского часа позволяет получить максимальное количество баллов. В данном случае реализован **многопользовательский** чат.

Запускается сервер. Подключается клиент. После чего клиент получает от сервера запрос на отправку имени (логина в чате). Затем клиент может участвовать в переписке вместе с другими клиентами. Их (участников чата) количество указывается в коде сервера.

Содержимое файла client.py:

```
import socket
import threading

def receive_messages(socket):
    try:
        while True:
            data = socket.recv(1024)
            print(data.decode("utf-8"), end='')
    except ConnectionAbortedError as e:
        print('Вы покинули чат')


if __name__ == "__main__":
```

```
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:

        s.connect(('localhost', 8080))

        data = s.recv(1024)
        print(data.decode("utf-8"))

        name = input().encode("utf-8")
        s.send(name)

        print('Print you messages...')
        print('q for quit. "." for update')

        x = threading.Thread(target=receive_messages, args=(s,))
        x.start()

        while True:
            message = input().encode("utf-8")
            if message == b'.':
                print("\r")
            s.send(message)
            if message == b'q':
                break
```

Код клиента все еще очень прост. Здесь потоки не то чтобы очень нужны. Так меньше нагромождений кода и вложенных while True. Клиент может обновить чат – попросить сервер отправить ему все сообщения, которые пришли с момента его последнего обновления или отправки сообщения – для этого ему нужно написать “.”. Так же клиент может покинуть чат, отправив q.

В строке print(data.decode("utf-8"), end='') в функции receive_messages можно видеть, что функции print передается аргумент end. Сервер после каждого сообщения от других пользователей добавляет символ \n. И отправляет несколько сообщений сразу, поэтому получаемые от сервера данные уже содержат все необходимые переносы строк.

Содержимое файла server.py:

```
import socket
import threading

def get_connect(number):
    print("I am thread number", number)
    clientsocket, address = socket.accept()
    clientsocket.send(b'Please enter your name: ')
    name = clientsocket.recv(1024)
    name = name.decode("utf-8")

    messages[name] = []

    print(name, 'is connected')
    while True:
        data = clientsocket.recv(1024)
        if data == b'q':
            print(name, 'left the chat')
            break
        else:
```

```
            if data != b'.':
                message = name + ':' + data.decode("utf-8")
                print(message)
                for key in messages:
                    if key != name:
                        messages[key].append(message.encode("utf-8"))
            while True:
                if bool(messages[name]):
                    clientsocket.send(messages[name].pop(0)+b'\n')
                else:
                    break

if __name__ == "__main__":

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind(('localhost', 8080))
        s.listen(10)
        messages = {}
        messages['all'] = []
        threads = []
        for i in range(3):
            x = threading.Thread(target=get_connect, args=(i+1,))
            threads.append(x)
            x.start()
        for i in range(3):
            threads[i].join()
        #print(messages['all'])
```

Словарь messages содержит списки для сообщений для каждого участника сайта (те, сообщения, которые нужно отправлять клиенту, все, которые написал не он).

Сперва создается указанное количество потоков – столько пользователей сможет пользоваться чатом. Целевая функция потоков get_connect принимает подключение, «знакомится» с клиентом. После чего в бесконечном цикле принимает от него сообщения и добавляет их в messages ко всем остальным клиентам, чтобы другие потоки занялись их отправкой. Так же, после приема сообщений от клиента эта функция в порядке очереди отправляет все сообщения клиенту, которые накопились в его списке messages[name]. Если полученное сообщение “.”, то оно не добавляется в списки сообщений других пользователей, но действует как кнопка «обновить» - сервер так же отправляет пользователю все накопившиеся у него сообщения от других клиентов.

Так выглядит вывод:

Server:

```
$ python server.py
I am thread number 1
I am thread number 2
I am thread number 3
Lena is connected
Fifa is connected
Fifa:Hello!!!
Lena:Hello, Fifa!
Lena:What's up?
Fifa:I am fine! The wheather is amazing!
Fifa:Let's go play tennis?
Lena:Yes, that's great idea!
Lena:I'll be near your home in 2 minutes
Fifa:fine! where we'll go?
Fifa:Ok!
```

```
Fifa left the chat
Lena left the chat
Kolya is connected
Kolya:Hello!
Kolya:anybody?..
Kolya left the chat
```

Client 1: Lena

```
$ python client.py
Please enter your name:
Lena
Print you messages...
q for quit. "." for update
.

Fifa:Hello!!!
Hello, Fifa!
What's up?
.

Fifa:I am fine! The wheather is amazing!
Fifa:Let's go play tennis?
Yes, that's great idea!
I'll be near your home in 2 minutes
.

Fifa:fine! where we'll go?
Fifa:Ok!
q
Вы покинули чат
```

Client 2: Fifa

```
$ python client.py
Please enter your name:
Fifa
Print you messages...
q for quit. "." for update
Hello!!!
.

Lena:Hello, Fifa!
Lena:What's up?
I am fine! The wheather is amazing!
Let's go play tennis?
.

Lena:Yes, that's great idea!
fine! where we'll go?
Lena:I'll be near your home in 2 minutes
Ok!
q
Вы покинули чат
```

Client 3: Kolya

```
$ python client.py
Please enter your name:
Kolya
Print you messages...
q for quit. "." for update
```

```
Hello!
.

.

anybody?..
q
Вы покинули чат
```

Коля не видит чужих сообщений, так как он подключился, когда все ушли. Однако легко можно изменить эту логику. При подключении клиента, можно в его список-очередь сообщений добавлять messages[‘all’].

После отключения всех подключенных пользователей сервер завершает свою работу.