Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет инфокоммуникационных технологий

**ОТЧЕТ**

**О ЛАБОРАТОРНОЙ РАБОТЕ № 1**

по дисциплине: Web-программирование

**Специальность**:

45.03.04 Интеллектуальные системы в гуманитарной сфере

|  |  |
| --- | --- |
| **Проверил**:  Говоров А.И. \_\_\_\_\_\_\_\_\_\_  **Дата**: «\_\_» \_\_\_\_ 2022 г.  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Выполнил**:  студент группы K33421  Боброва М.И. |

Санкт-Петербург 2022 г.

**Практическое задание:**

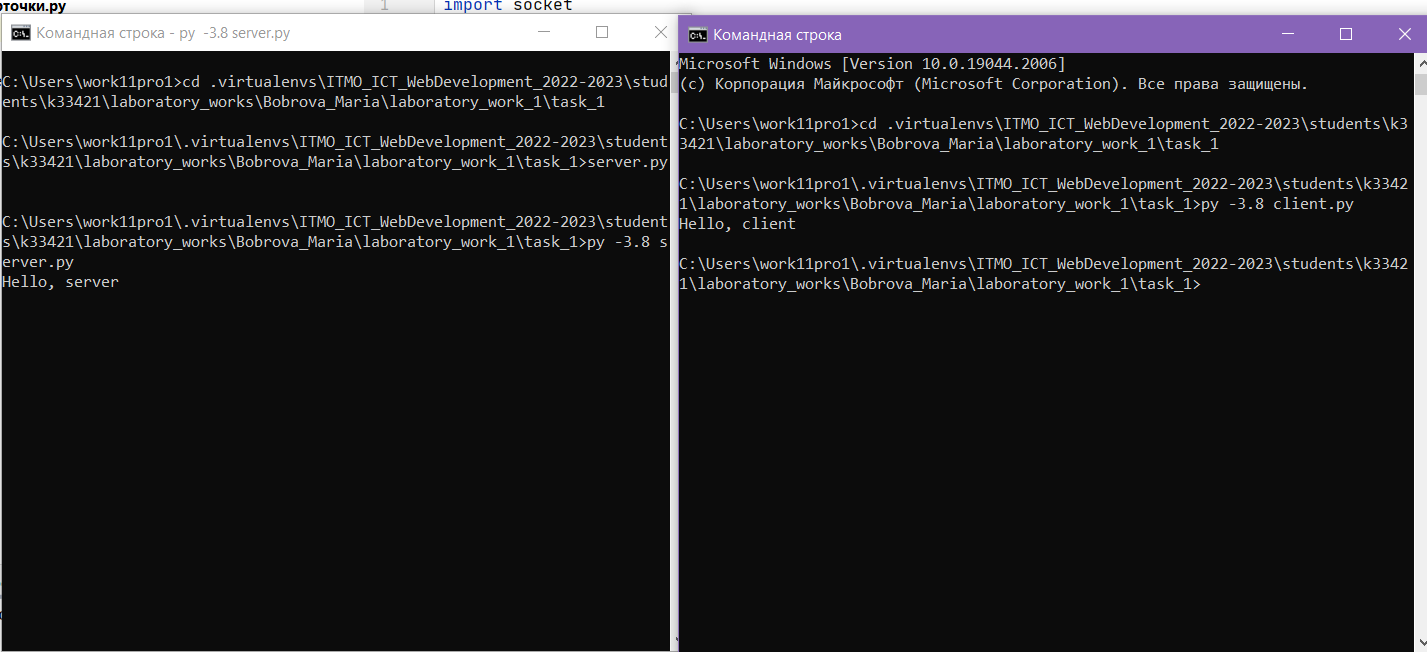
1. Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента. Обязательно использовать библиотеку socket. Реализовать с помощью протокола UDP.

**Код для сервера:**

import socket *#импорт библиотеки*sock = socket.socket(socket.SOCK\_DGRAM) *#создание сокета для сервера UDP  
# Для привязки используется функция bind сокета, которая принимает массив, содержащий два элемента: хост и порт.*sock.bind((socket.gethostname(), 1234))  
sock.listen(1) *#максимальное число соединений*while True:  
 try: *#пока верно, выполняется блок инструкицй  
 #Ф-ция ждёт появление входящего соединения и возвращает связанный с ним сокет и адрес подключившегося.  
 #Адрес — массив, состоящий из IP-адреса и порта.* conn, addr = sock.accept()  
 data = conn.recv(16384) *#чтение данных с определенным кол-м байт* udata = data.decode(**"utf-8"**) *#декодирование данных из bytes в строку* print(udata)  
 msg = **"Hello, client"** *#строка для клиента* conn.send(msg.encode(**"utf-8"**)) *#отправка данных в закодированном виде, т.к. ф-ция send принимает тип bytes* conn.close() *#закрытие сокета* except KeyboardInterrupt: *#в случае ошибки, выполняется блок инструкицй* sock.close() *#закрытие сокета* break

**Код для клиента:**

import socket *#импорт библиотеки*conn = socket.socket(socket.SOCK\_DGRAM) *#создание сокета для сервера UDP*conn.connect((socket.gethostname(), 1234)) *#подключение к хосту (IP-адрес и порт)*msg = **'Hello, server'** *#сообщение для сервера*conn.send(msg.encode(**"utf-8"**)) *#*data = b"" *#создание пустой байтовой строки*tmp = conn.recv(16384) *#отсюда все как обычно*while tmp:  
 data += tmp  
 tmp = conn.recv(16384)  
print(data.decode(**"utf-8"**))  
conn.close()

**Результат:**

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: Теорема Пифагора.

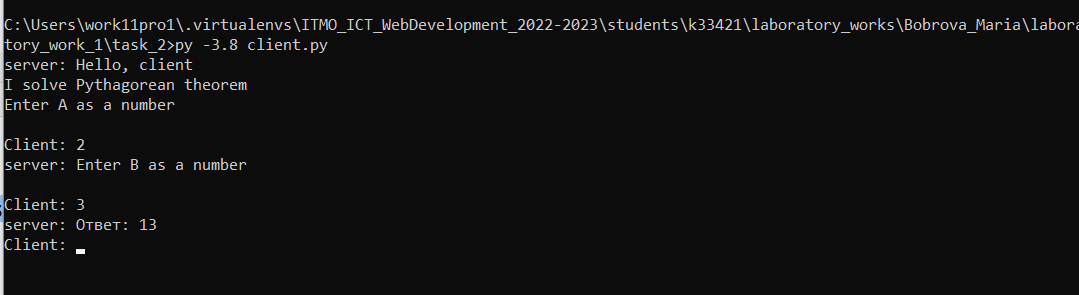
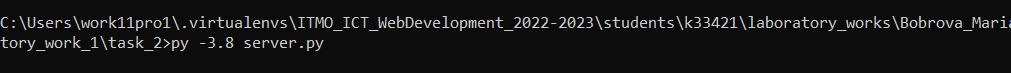
Реализовать с помощью протокола TCP

**Код для сервера:**

import socket  
  
  
def pifagor(a, b): *#функия, которая возвращает гипотенузу* return a\*\*2 + b\*\*2  
  
  
sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) *#создание сокета для сервера TCP*sock.bind((socket.gethostname(), 1234))  
*# Для привязки используется функция bind сокета, которая принимает массив, содержащий два элемента: хост и порт.*sock.listen(1) *#максимальное число соединений  
#Ф-ция ждёт появление входящего соединения и возвращает связанный с ним сокет и адрес подключившегося.  
#Адрес — массив, состоящий из IP-адреса и порта.*conn, addr = sock.accept()  
*#отправка данных в закодированном виде, т.к. ф-ция send принимает тип bytes*conn.send(str.encode(**f"Hello, client**\n**"  
 f"I solve Pythagorean theorem**\n**"  
 f"Enter A as a number**\n**"**))  
a = **""** *#катет*b = **""** *#катет*c = **""** *#гипотенуза*while not c:  
 while not a: *#первый катет* data = conn.recv(1024).decode() *#чтение данных с определенным кол-м байт* if data.isdigit(): *#если данные числовые* a = int(data) *#перевод в тип данных integer* conn.send(b"Enter B as a number\n") *#сообщение появляется в терминале* else:  
 conn.send(b"Not a number\n"   
 b"Enter A as a number\n") *#сообщение появляется в терминале, если данные не числовые* while not b: *#второй катет* data = conn.recv(1024).decode()  
 if data.isdigit():  
 b = int(data)  
 else:  
 conn.send(b"Not a number\n"  
 b"Enter B as a number\n")  
  
 c = pifagor(a, b) *#функция принимает два введенных числа и считает гипотенузу* conn.send(str.encode(**f"Ответ:** {c}**"**)) *#отправка ответа*conn.close() *#закрытие сокета*

**Код для клиента:**

import socket  
  
sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
sock.connect((socket.gethostname(), 1234))  
while True:  
 data = sock.recv(1024)  
 print(**f"server:** {data.decode()}**"**)  
 msg = input(**"Client: "**)  
 sock.send(str.encode(msg))

**Результат: **

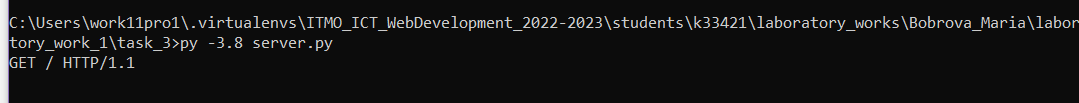
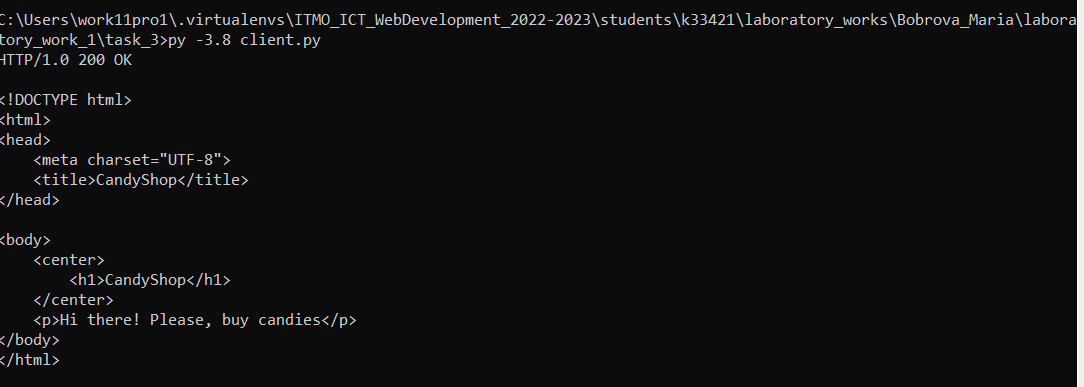
3. Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

**Код для сервера:**

import socket  
with socket.socket() as sock:  
 sock.bind((socket.gethostname(), 1234))  
 sock.listen(1)  
 conn, addr = sock.accept()  
 with conn:  
 with open(**'C:/Users/work11pro1/.virtualenvs/ITMO\_ICT\_WebDevelopment\_2022-2023/students/k33421/laboratory\_works/Bobrova\_Maria/laboratory\_work\_1/task\_3/index.html'**) as f:  
 msg = f.read()  
 print(conn.recv(16348).decode(**'utf-8'**))  
 response = **'HTTP/1.0 200 OK**\n\n**'** + msg  
 conn.sendall(response.encode(**'utf-8'**))

**Код для клиента:**

import socket  
  
with socket.socket() as sock:  
 sock.connect((socket.gethostname(), 1234))  
 sock.settimeout(5)  
 sock.send(b"GET / HTTP/1.1\n")  
 data = sock.recv(16384)  
 udata = data.decode(**'utf-8'**)  
 print(udata)

**Результат:**

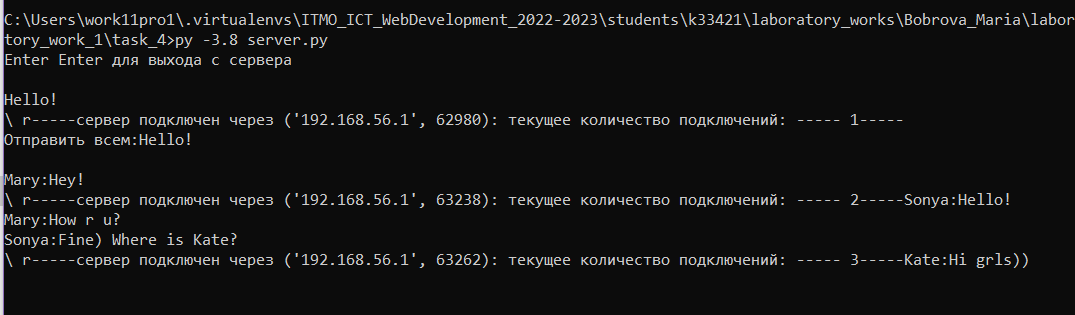
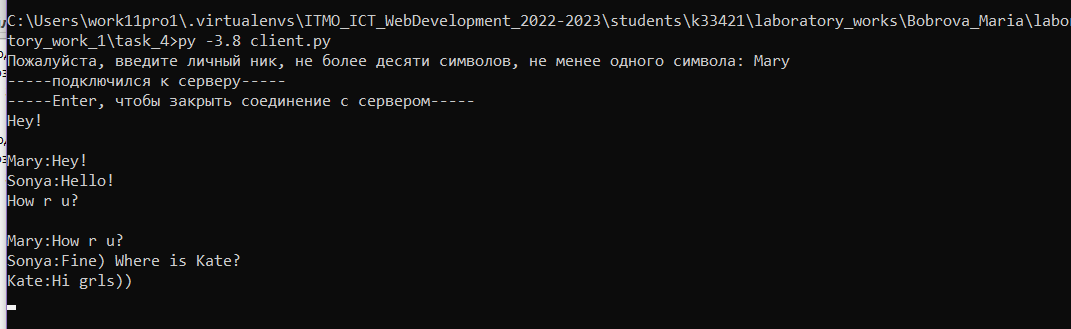
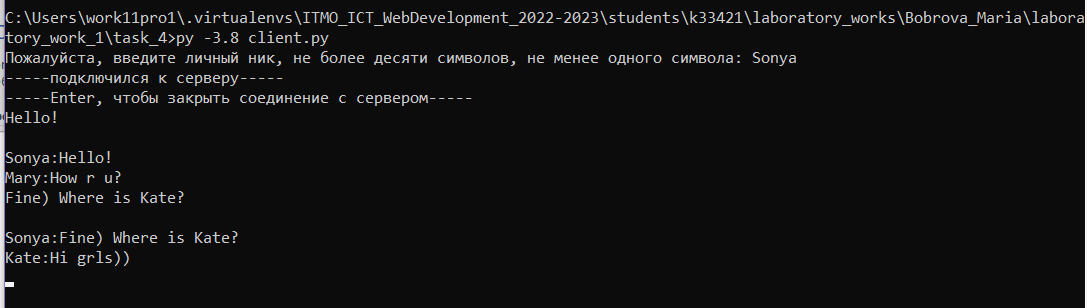
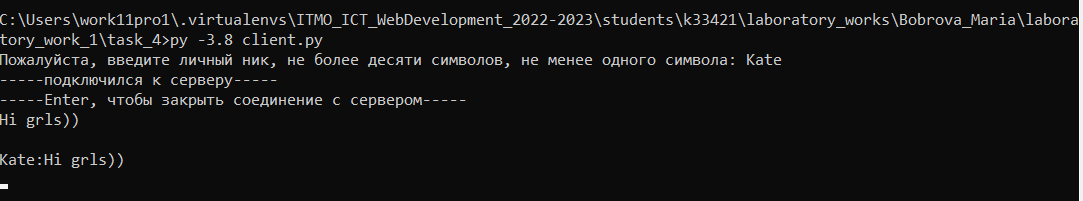
4. Реализовать двухпользовательский или многопользовательский чат. Реализован многопользовательский чат

**Код для сервера:**

import socket, threading  
  
*# Создаем объект сокета*server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
*# Укажите IP и порт сервера*server.bind((socket.gethostname(), 1234))  
*# Максимальное количество подключений*server.listen(5)  
print(**'Enter Enter для выхода с сервера'**)  
*# Создайте список клиентов*clients = list()  
*# Хранить клиентов, которые создали потоки*end = list()  
*# Блокировка ожидания подключения клиента, возврата объекта подключения и адреса косвенного объекта*def accept():  
 while True:  
 client, addr = server.accept()  
 clients.append(client)  
 print(**"\ r"** + **'-'** \* 5 + **f'сервер подключен через** {addr}**: текущее количество подключений: -----** {len (clients)}**'** + **'-'** \* 5, end = **''**) *#Взаимодействие с другими людьми*def recv\_data(client):  
 while True:  
 *# Принимаем информацию от клиента* try:  
 indata = client.recv(1024)  
 except Exception as e:  
 clients.remove(client)  
 end.remove(client)  
 print(**"\ r"** + **'-'** \* 5 + **f'Сервер отключен: текущее количество подключений: -----** {len (clients)}**'** + **'-'** \* 5, end = **''**)  
 break  
 print(indata.decode(**'utf-8'**))  
 for clien in clients:  
 *# Перенаправить информацию от клиента и отправить ее другим клиентам* if clien != client:  
 clien.send(indata)  
  
  
def outdatas():  
 while True:  
 *# Введите информацию, которая будет предоставлена клиенту* print(**''**)  
 outdata = input(**''**)  
 print()  
 if outdata == **'enter'**:  
 break  
 print(**'Отправить всем:% s'** % outdata)  
 *# Отправлять информацию каждому клиенту* for client in clients:  
 client.send(**f"Сервер:** {outdata}**"**.encode(**'utf-8)'**))  
  
def indatas():  
 while True:  
 *# Выполните цикл подключенных клиентов и создайте соответствующий поток* for clien in clients:  
 *# Если поток уже существует, пропустить* if clien in end:  
 continue  
 index = threading.Thread(target=recv\_data, args=(clien,))  
 index.start()  
 end.append(clien)  
  
*# Создать многопоточность  
# Создать получающую информацию, объект потока*t1 = threading.Thread(target=indatas, name=**'input'**)  
t1.start()  
  
*# Создать отправляемое сообщение, объект потока*t2 = threading.Thread(target=outdatas, name=**'out'**)  
t2.start()  
  
*# Ожидание подключения клиента, объект потока*t3 = threading.Thread(target=accept(), name=**'accept'**)  
t3.start()  
  
*# Блокировать округ, пока подпоток не будет завершен, и основной поток не может закончиться  
# t1.join()*t2.join()  
  
*# Выключите все серверы*for client in clients:  
 client.close()  
print(**'-'** \* 5 + **'сервер отключен'** + **'-'** \* 5)

**Код для клиента:**

import socket, threading  
  
*# Создать клиентский объект*client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
while True:  
 name = input(**'Пожалуйста, введите личный ник, не более десяти символов, не менее одного символа: '**)  
 if 1 < len(name) < 10:  
 break  
  
*# Подключить клиента*client.connect((socket.gethostname(), 1234))  
print(**'-'** \* 5 + **'подключился к серверу'** + **'-'** \* 5)  
print(**'-'** \* 5 + **'Enter, чтобы закрыть соединение с сервером'** + **'-'** \* 5)  
  
  
def outdatas():  
 while True:  
  
 *# Введите информацию, которая будет отправлена на сервер* outdata = input(**''**)  
 print()  
 if outdata == **'enter'**:  
 break  
 *# Отправить на сервер* client.send(**f'**{name}**:**{outdata}**'**.encode(**'utf-8'**))  
 print(**'%s:%s'** % (name, outdata))  
  
  
def indatas():  
 while True:  
 *# Принимаем информацию с сервера* indata = client.recv(1024)  
  
 *# Закодировать полученную информацию* print(indata.decode(**'utf-8'**))  
  
  
*# Создать многопоточность  
# Установить получение информации, объект потока*t1 = threading.Thread(target=indatas, name=**'input'**)  
  
*# Создание выходной информации, объект потока*t2 = threading.Thread(target=outdatas, name=**'out'**)  
  
*# Начать многопоточность*t1.start()  
t2.start()  
  
*# Заблокировать поток, основной поток не может завершиться, пока не завершится выполнение дочернего потока.  
# t1.join()*t2.join()  
  
*# Закрыть соединение*print(**'-'** \* 5 + **'сервер отключен'** + **'-'** \* 5)  
client.close()

**Результат:**

5. Необходимо написать простой web-сервер для обработки GET и POST http запросов средствами Python и библиотеки socket.