

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
ИТМО»**

Факультет «Инфокоммуникационных технологий» Направление подготовки «09.03.03  
Прикладная информатика» Бакалаврская программа «Мобильные и сетевые технологии»

**Лабораторная работа No1**  
**по дисциплине «Веб программирование»**

**«Работа с сокетами»**

**Выполнил**

\_\_\_\_\_ / Бурак П.В ., КЗ3402

(подпись) (Фамилия И.О., группа)

**Проверил**

\_\_\_\_\_ / Говоров А. И. **Дата** \_\_\_\_\_

**Санкт-Петербург 2021**

## Цель работы

Овладеть практическими навыками и умениями реализации web-серверов и использования сокетов.

## Task 1

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

### Server.py

```
1 import socket
2
3 server = socket.socket()
4 host = '127.0.0.1'
5 port = 8000
6 server.bind((host, port))
7 server.listen()
8
9 const, addr = server.accept()
10 data = const.recv(16384).decode('utf-8')
11 print(data)
12 const.send(b'Hello, client\n')
13 const.close()
```

### Client.py

```
1 import socket
2
3 const = socket.socket()
4 const.connect(('127.0.0.1', 8000))
5 const.send(b'Hello, server')
6
7 print(const.recv(16384).decode('utf-8'))
8
9 const.close()
10
```

## Task 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 пифогора.

## Server.py

```
1 import socket
2
3 server = socket.socket()
4 host = '127.0.0.1'
5 port = 8000
6 server.bind((host, port))
7 server.listen()
8
9 conn, addr = server.accept()
10 first = int(conn.recv(16384).decode('utf-8'))
11 second = int(conn.recv(16384).decode('utf-8'))
12 print(first, second, sep='\n')
13
14 hypotenuse = str((first ** 2 + second ** 2) ** (1 / 2))
15 conn.send(hypotenuse.encode())
16 conn.close()
17
```

## Client.py

```
1 import socket
2
3 conn = socket.socket()
4 conn.connect(('127.0.0.1', 8000))
5
6 conn.send(input("Первая сторона").encode())
7 conn.send(input("Вторая сторона").encode())
8 hypotenuse_conn = conn.recv(16384)
9 hypotenuse = hypotenuse_conn.decode('utf-8')
10 print(hypotenuse)
11 conn.close()
12
```

## Task 3

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

Задание: сделать сервер, который может:

- Принять и записать информацию о дисциплине и оценке по дисциплине.
- Отдать информацию обо всех оценках по дисциплине в виде html-страницы.

## Client.py

```
109     }
110 }
111     </style>
112 </head>
113 <body>
114     <div class="login-box">
115         <form action="/" method="post">
116             <div class="user-box">
117                 <label for="name"></label>
118                 <input type="text" id="name" placeholder="subject" name="subject"/>
119             </div>
120             <div class="user-box">
121                 <label for="mail"></label>
122                 <input type="number" id="mail" placeholder="mark" name="mark"/>
123             </div>
124             <input type="submit">
125         </form>
126     </div>
127 </body>
128 </html>
```

## Server.py

```
import socket
import sys

class MyHTTPServer:

    def __init__(self, host, port, name):

        self.host = host
        self.port = port
        self.name = name

    # 1. Запуск сервера на сокете, обработка входящих соединений
    def serve_forever(self):
        # создание сокета
        conn = socket.socket(# задаем семейство протоколов 'Интернет' (INET)
                             socket.AF_INET, # задаем тип передачи данных 'поточковый' (TCP)
                             socket.SOCK_STREAM)# выбираем протокол 'по умолчанию' для TCP, т.е. IP
        conn.bind((self.host, self.port))
        conn.listen(10)

        while True:
            # Ждем пользователя
            client_sock, client_addr = conn.accept()
            self.serve_client(client_sock)

    def serve_client(self, client_sock):
        # 2. Обработка клиентского подключения и вызываем parse_request
        data = client_sock.recv(16384)
        data = data.decode('utf-8')
        url, method, headers, body = self.parse_request(data)
        resp = self.handle_request(url, method, body)
```

```

    if resp:
        self.send_response(client_sock, resp)

def parser_request(self, data):
    # 3. Разбивает данные на куски
    data = data.replace('\r', '')
    lines = data.split('\n')

    method, url, protocol = lines[0].split()
    i = lines.index('')
    headers = lines[1:i]
    body = lines[-1]

    return url, method, headers, body

# 4. совершает проверку. И смотрит какой запрос пришел
def handle_request(self, url, method, body):

    if url == "/":

        if method == "GET":
            response = "HTTP/1.1 200 OK\n\n"

            with open('index.html', 'r') as f:
                response += f.read()
            return response

        if method == "POST":
            newText = body.split('&')

            for a in newText:
                if a.split('=')[0] == 'subject':
                    subjects.append(a.split('=')[1])

                if a.split('=')[0] == 'mark':
                    marks.append(a.split('=')[1])

            response = "HTTP/1.1 200 OK\n\n"
            response += '<html>' \
                '<head>' \
                '<title>appraisals</title>' \
                '</head>' \
                '<body><table border=2>'
            for s, m in zip(subjects, marks):
                response += f"<tr><td>{s}</td><td>{m}</td></tr>"
            response += "</table></body></html>"
            return response

def send_response(self, client_sock, resp):
    # 5. Функция для отправки ответа
    client_sock.send(resp.encode('utf-8'))

```

```

if __name__ == '__main__':
    host = '127.0.0.1'
    port = 5010
    name = 'localhost'
    serv = MyHTTPServer(host, port, name)
    subjects = []
    marks = []
    try:
        serv.serve_forever()
    except KeyboardInterrupt:
        pass

```

subject

mark

Отправить

piter	12
ikonenko	1
gleb	23

## Task 4

Реализовать двухпользовательский или многопользовательский чат.

Client.py

```

import threading
import os

UDP_MAX_SIZE = 65535

def listen(s: socket.socket):
    while True:
        msg = s.recv(UDP_MAX_SIZE)
        print('\r\r' + msg.decode('ascii') + '\n' + f'you:', end='')

def connect(host: str = '127.0.0.1', port: int = 8084):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    s.connect((host, port))

    threading.Thread(target=listen, args=(s,), daemon=True).start()

    s.send('__join'.encode('ascii'))

    while True:
        msg = input(f'you: ')
        s.send(msg.encode('ascii'))

if __name__ == '__main__':
    os.system('clear')
    print('Welcome to chat')
    connect()

```

server.py

```

import socket

UDP_MAX_SIZE = 65535

def listen(host: str = '127.0.0.1', port: int = 8084):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.bind((host, port))
    print(f'Listening at {host}:{port}')

    members = []
    while True:
        msg, addr = s.recvfrom(UDP_MAX_SIZE)
        if addr not in members:
            members.append(addr)

        if not msg:
            continue

        client_id = addr[1]
        if msg.decode('ascii') == '__join':
            print(f'client{client_id} join chat')
            continue

        msg = f'client{client_id}: {msg.decode("ascii")}'
        for member in members:
            if member == addr:
                continue

            s.sendto(msg.encode('ascii'), member)

if __name__ == '__main__':
    listen()

```

Вывод: Я научился реализовывать веб-сервер с помощью библиотеки socket в языке программирования python.