

Отчёт по дисциплине  
«Web-программирование»

Лабораторная работа №1

Выполнила:

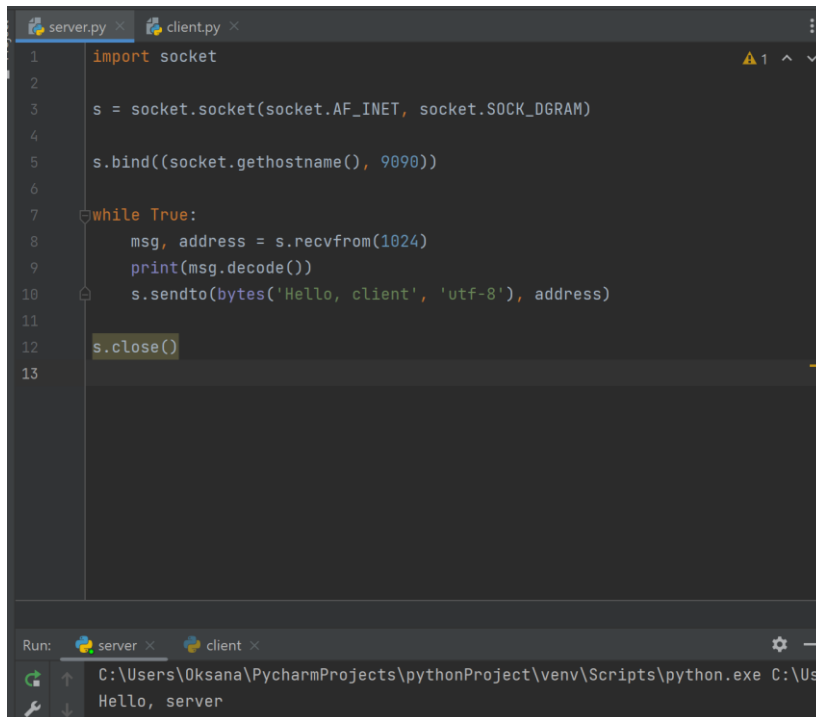
Панкратова Оксана Александровна К33401

Санкт-Петербург

2023г

## Пункт 1

Со стороны сервера запускается сокет на порту 9090, который получает и выводит сообщение от клиента и отправляет ответ.

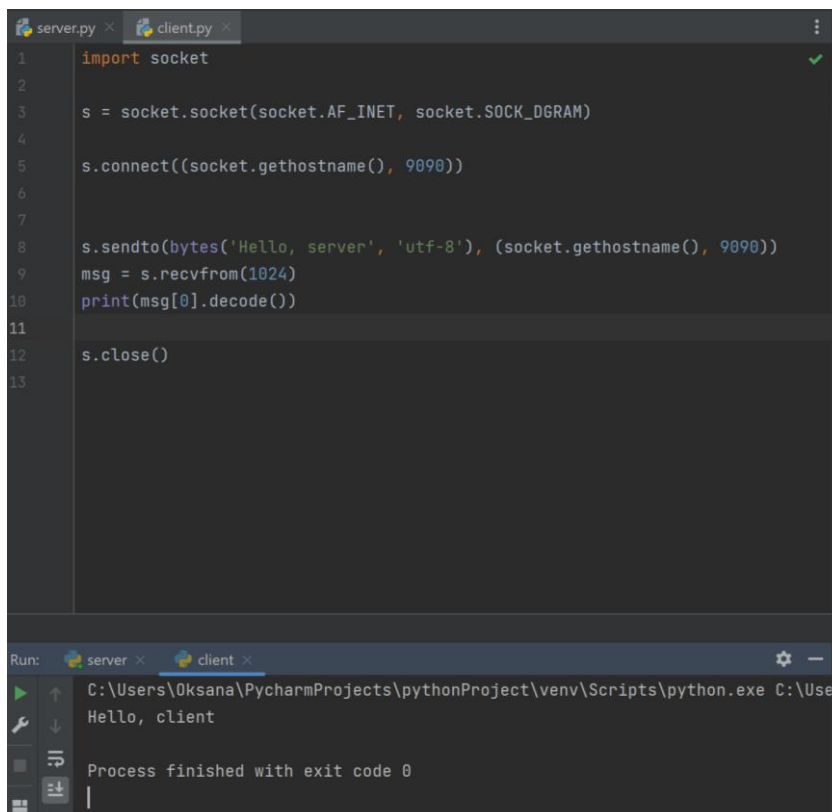


```
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
4
5 s.bind((socket.gethostname(), 9090))
6
7 while True:
8     msg, address = s.recvfrom(1024)
9     print(msg.decode())
10    s.sendto(bytes('Hello, client', 'utf-8'), address)
11
12 s.close()
13
```

Run: server x client x

C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Use  
Hello, server

На стороне клиента также запускается сокет, который подключается к серверу через порт 9090, отправляет сообщение, получает ответ, выводит его в командную строку и закрывается.



```
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
4
5 s.connect((socket.gethostname(), 9090))
6
7
8 s.sendto(bytes('Hello, server', 'utf-8'), (socket.gethostname(), 9090))
9 msg = s.recvfrom(1024)
10 print(msg[0].decode())
11
12 s.close()
13
```

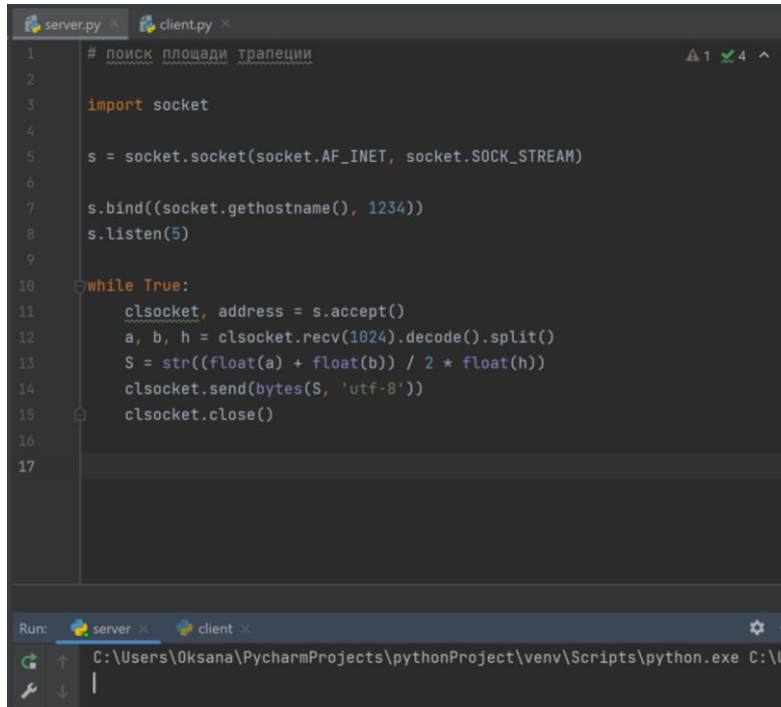
Run: server x client x

C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Use  
Hello, client

Process finished with exit code 0

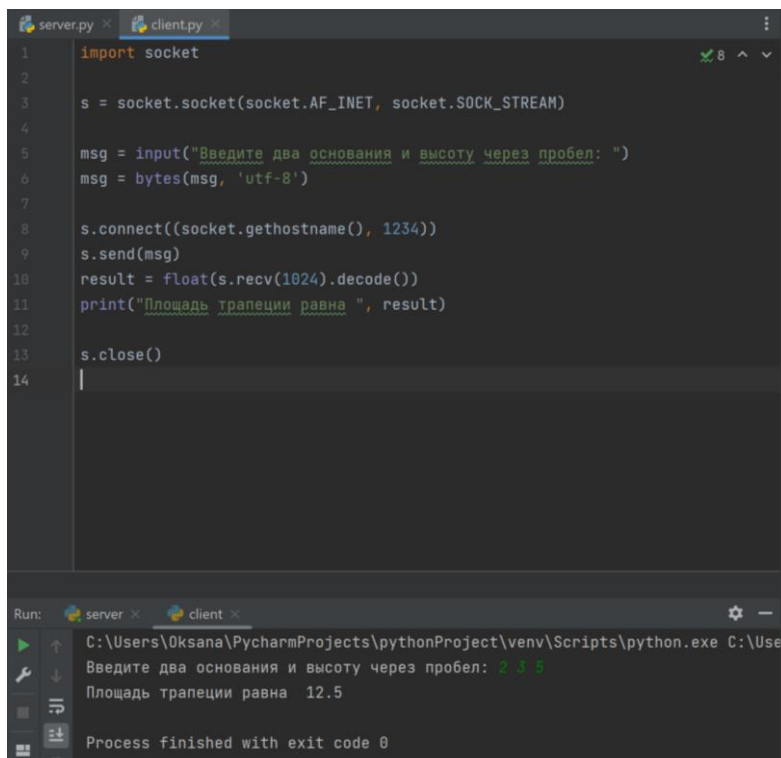
## Пункт 2

В данном случае серверный сокет не выводит никакой информации, только получает запрос от клиента, обрабатывает его и отправляет ответ.



```
1 # поиск площади трапеции
2
3 import socket
4
5 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6
7 s.bind((socket.gethostname(), 1234))
8 s.listen(5)
9
10 while True:
11     csocket, address = s.accept()
12     a, b, h = csocket.recv(1024).decode().split()
13     S = str((float(a) + float(b)) / 2 * float(h))
14     csocket.send(bytes(S, 'utf-8'))
15     csocket.close()
16
17
```

Клиент запрашивает данные у пользователя, формирует сообщение, подключается к серверу и отправляет запрос. После получения ответа выводит их пользователю.



```
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 msg = input("Введите два основания и высоту через пробел: ")
6 msg = bytes(msg, 'utf-8')
7
8 s.connect((socket.gethostname(), 1234))
9 s.send(msg)
10 result = float(s.recv(1024).decode())
11 print("Площадь трапеции равна ", result)
12
13 s.close()
14
```

Run: server x client x

C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Use

Введите два основания и высоту через пробел: 2 3 5

Площадь трапеции равна 12.5

Process finished with exit code 0

### Пункт 3

Здесь сокет после установки соединения отправляет html-сообщение клиенту построчно, а в конце так же передает содержимое файла index.html, содержащего шаблон html-страницы.

```
server.py x index.html x client.py x
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 s.bind((socket.gethostname(), 1212))
6 s.listen(5)
7
8 while True:
9     csocket, addr = s.accept()
10    csocket.send(bytes('HTTP/1.1 200 OK\n', 'utf-8'))
11    csocket.send(bytes('Content-Type: text/html; charset=utf-8\n', 'utf-8'))
12    csocket.send(bytes('Content-Length: 144\n', 'utf-8'))
13    csocket.send(bytes('\n', 'utf-8'))
14    csocket.send(bytes('\n', 'utf-8'))
15
16    file = open('index.html', 'rb')
17
18    for line in file:
19        csocket.send(line)
20    csocket.close()
21
22 s.close()
23
```

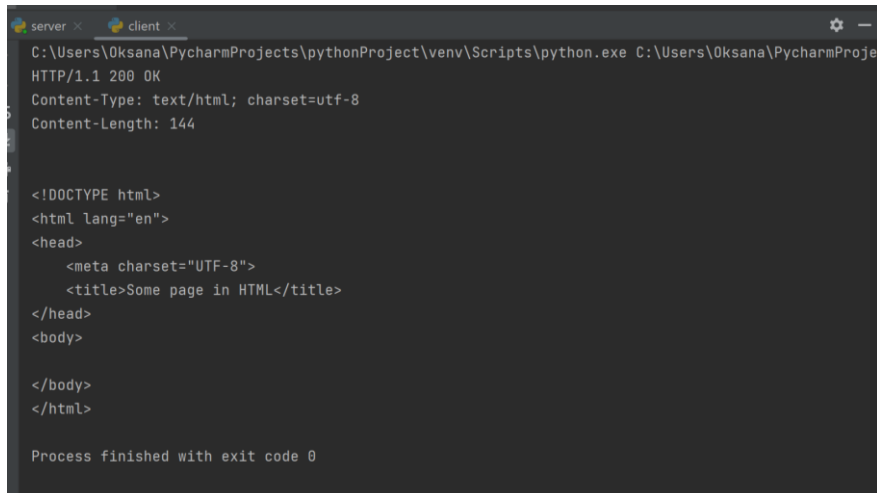
Шаблон html-страницы, передаваемый сокетом.

```
server.py x index.html x client.py x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Some page in HTML</title>
6 </head>
7 <body>
8
9 </body>
10 </html>
11
```

В данном случае клиент только получает данные, отправляемые сервером, и ВЫВОДИТ ИХ В КОНСОЛЬ.

```
server.py x index.html x client.py x
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 s.connect((socket.gethostname(), 1212))
6 msg = ''
7
8 while True:
9     line = s.recv(1024)
10    if not line:
11        break
12    msg += line.decode()
13
14 print(msg)
15
16 s.close()
17
```

## Консольный вывод клиентского сокета.



```
server x client x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 144

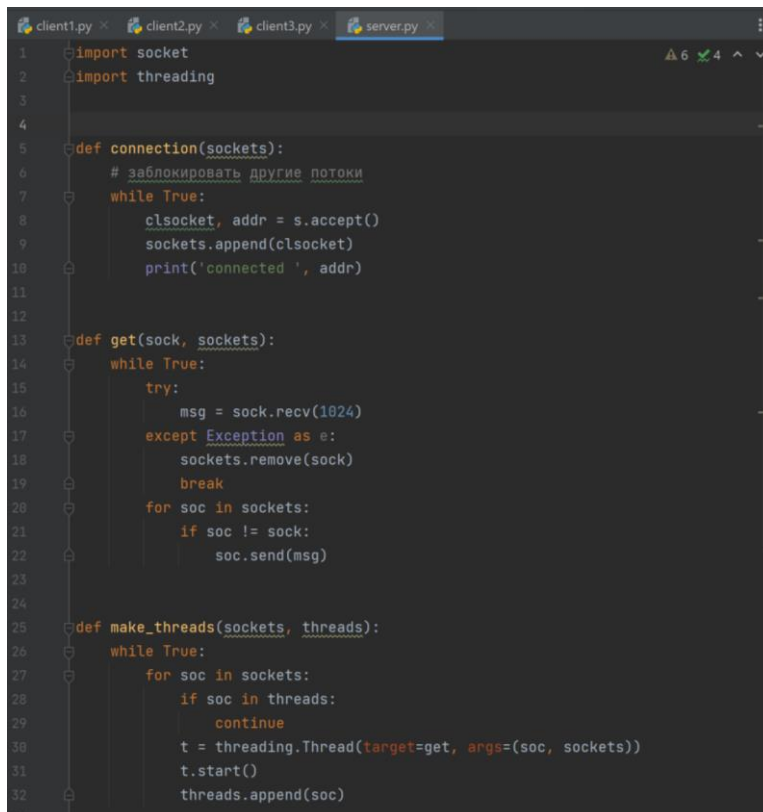
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Some page in HTML</title>
</head>
<body>

</body>
</html>

Process finished with exit code 0
```

## Пункт 4

В файле серверного сокета создается несколько функций. Первая отвечает за подключение пользователя, о чем выводит информацию в консоль. Вторая получает сообщение от клиента, если возвращается ошибка, то данное подключение удаляется из списка, и отправляет всем подключенным клиентам. Третья отвечает за создание потока для каждого клиента, чтобы они могли подключаться синхронно.



```
client1.py x client2.py x client3.py x server.py
1 import socket
2 import threading
3
4
5 def connection(sockets):
6     # заблокировать другие потоки
7     while True:
8         csocket, addr = s.accept()
9         sockets.append(csocket)
10        print('connected ', addr)
11
12
13 def get(sock, sockets):
14     while True:
15         try:
16             msg = sock.recv(1024)
17         except Exception as e:
18             sockets.remove(sock)
19             break
20         for soc in sockets:
21             if soc != sock:
22                 soc.send(msg)
23
24
25 def make_threads(sockets, threads):
26     while True:
27         for soc in sockets:
28             if soc in threads:
29                 continue
30             t = threading.Thread(target=get, args=(soc, sockets))
31             t.start()
32             threads.append(soc)
```

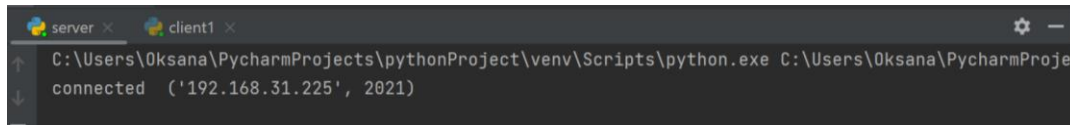
После всех функций, в основной части, запускается серверный сокет, объявляются массивы для хранения подключений и потоков. После чего создаются два потока для возможности одновременной обработки новых подключений и обработки сообщений из чата.

```
client1.py x client2.py x client3.py x server.py x
16 msg = sock.recv(1024)
17 except Exception as e:
18     sockets.remove(sock)
19     break
20 for soc in sockets:
21     if soc != sock:
22         soc.send(msg)
23
24
25 def make_threads(sockets, threads):
26     while True:
27         for soc in sockets:
28             if soc in threads:
29                 continue
30             t = threading.Thread(target=get, args=(soc, sockets))
31             t.start()
32             threads.append(soc)
33
34
35 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
36
37 s.bind((socket.gethostname(), 2020))
38 s.listen(10)
39
40 sockets = []
41 threads = []
42
43 t1 = threading.Thread(target=connection, args=(sockets,))
44 t1.start()
45
46 t2 = threading.Thread(target=make_threads, args=(sockets, threads,))
47 t2.start()
48
49 t2.join()
```

У всех клиентов скрипт один. Он объявляет функцию принятия сообщения, подключается к серверу и запускает поток для отслеживания сообщений, а в основном цикле отправляет сообщения введенные пользователем.

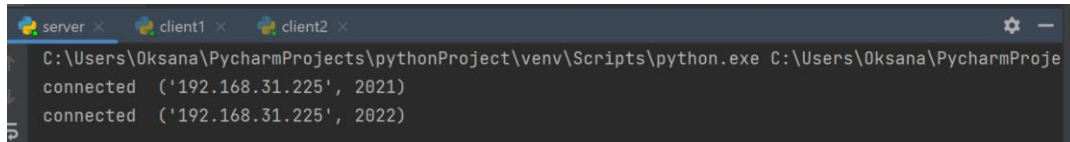
```
client1.py x client2.py x client3.py x server.py x
1 import socket
2 import threading
3
4
5 def get_msg(sock):
6     while True:
7         msg = sock.recv(1024)
8         print(msg.decode())
9
10
11 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12
13 s.bind((socket.gethostname(), 2021))
14 s.connect((socket.gethostname(), 2020))
15
16 get = threading.Thread(target=get_msg, args=(s,))
17 get.start()
18
19 while True:
20     msg_send = input()
21     s.send(bytes(msg_send, 'utf-8'))
22
```

Пример работы программы. Запуск сервера, подключение первого пользователя.



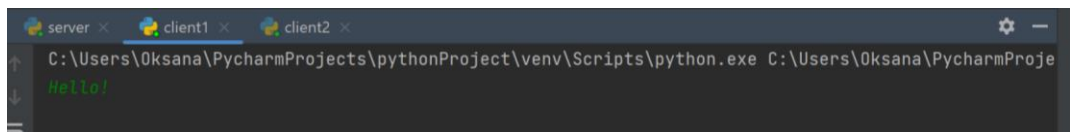
```
server x client1 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
connected ('192.168.31.225', 2021)
```

Подключение второго пользователя.



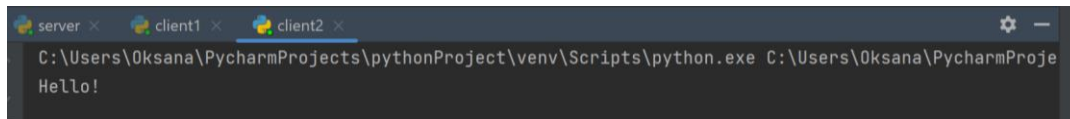
```
server x client1 x client2 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
connected ('192.168.31.225', 2021)
connected ('192.168.31.225', 2022)
```

Первый пользователь отправил сообщение.



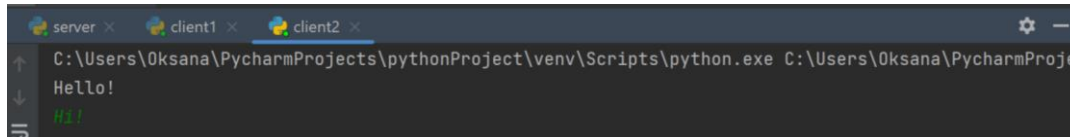
```
server x client1 x client2 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
```

Оно отобразилось у второго.



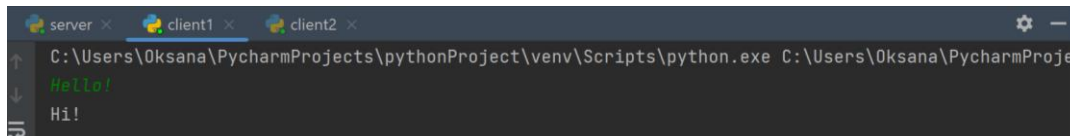
```
server x client1 x client2 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
```

Второй пользователь ответил.



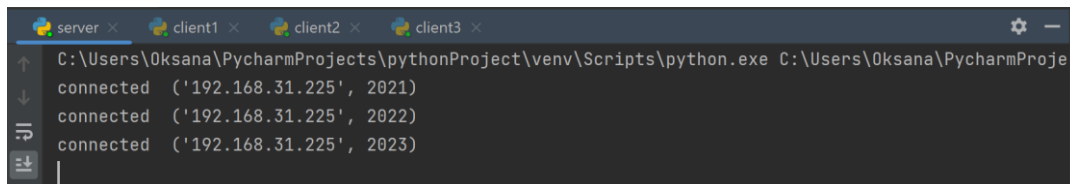
```
server x client1 x client2 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
Hi!
```

Ответ отобразился у первого



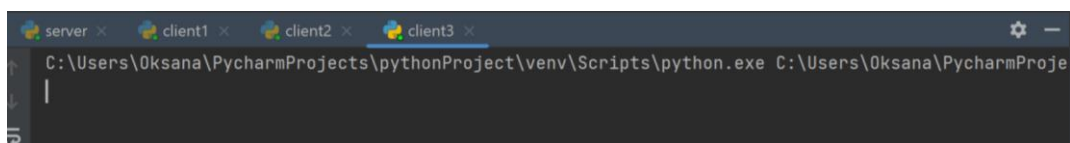
```
server x client1 x client2 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
Hi!
```

Подключение третьего пользователя.



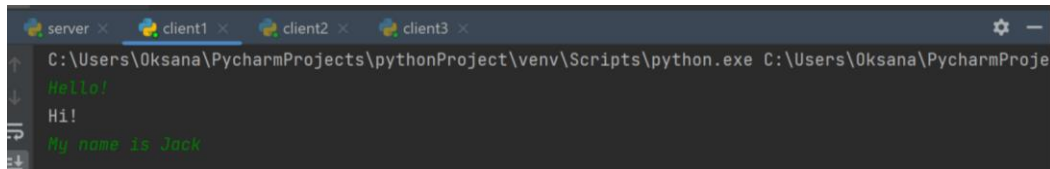
```
server x client1 x client2 x client3 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
connected ('192.168.31.225', 2021)
connected ('192.168.31.225', 2022)
connected ('192.168.31.225', 2023)
```

У него предыдущих сообщений нет, поскольку он подключился позже.



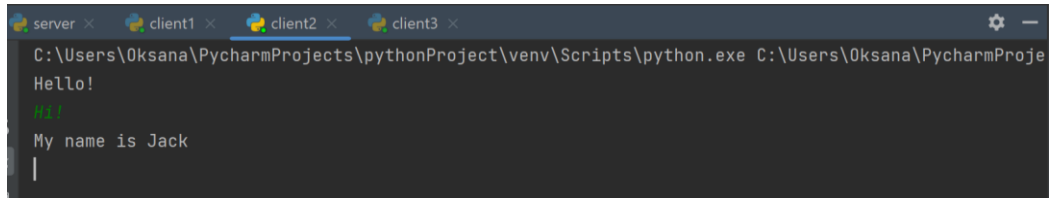
```
server x client1 x client2 x client3 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
connected ('192.168.31.225', 2021)
connected ('192.168.31.225', 2022)
connected ('192.168.31.225', 2023)
```

Первый пользователь снова отправляет сообщение.



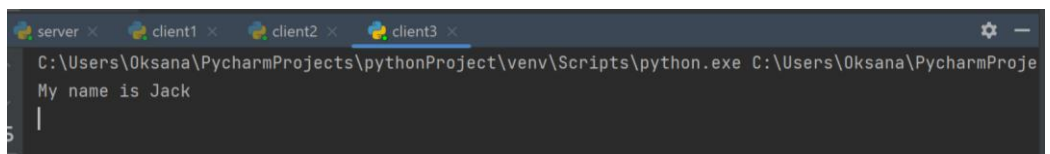
```
server x client1 x client2 x client3 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
Hi!
My name is Jack
```

Его получают второй.



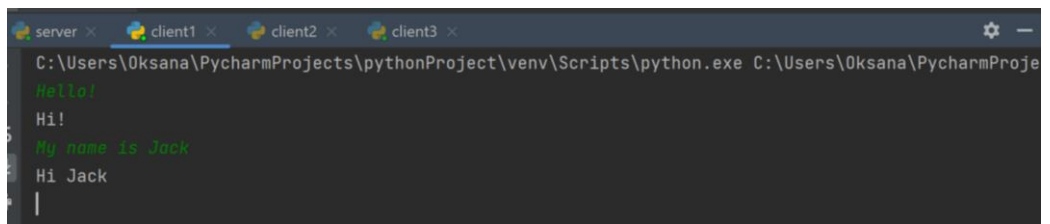
```
server x client1 x client2 x client3 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
Hi!
My name is Jack
|
```

И третий пользователи.



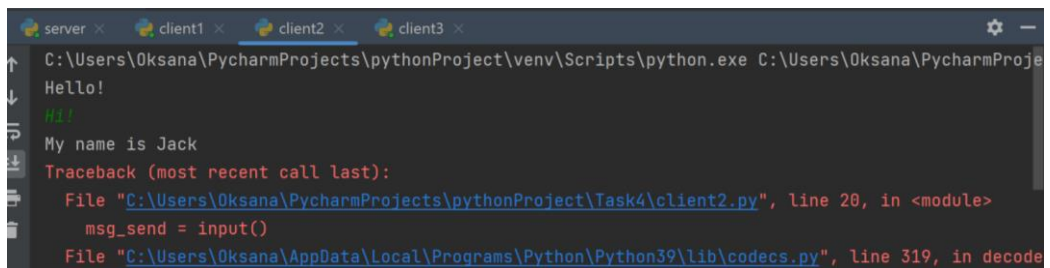
```
server x client1 x client2 x client3 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
My name is Jack
|
```

Второй клиент отключается и третий пишет сообщение. Оно приходит первому.



```
server x client1 x client2 x client3 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
Hi!
My name is Jack
Hi Jack
|
```

Но уже не приходит второму.



```
server x client1 x client2 x client3 x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProje
Hello!
Hi!
My name is Jack
Traceback (most recent call last):
  File "C:\Users\Oksana\PycharmProjects\pythonProject\Task4\client2.py", line 20, in <module>
    msg_send = input()
  File "C:\Users\Oksana\AppData\Local\Programs\Python\Python39\lib\codecs.py", line 319, in decode
```

## Пункт 5

В данном пункте сервер выполняет роль http-сервера. В первой функции проходит его инициализация и инициализация словаря для дисциплин, куда будут записываться оценки студентов. Далее описана основная функция сервера, где он запускается и обрабатывает клиентские подключения.

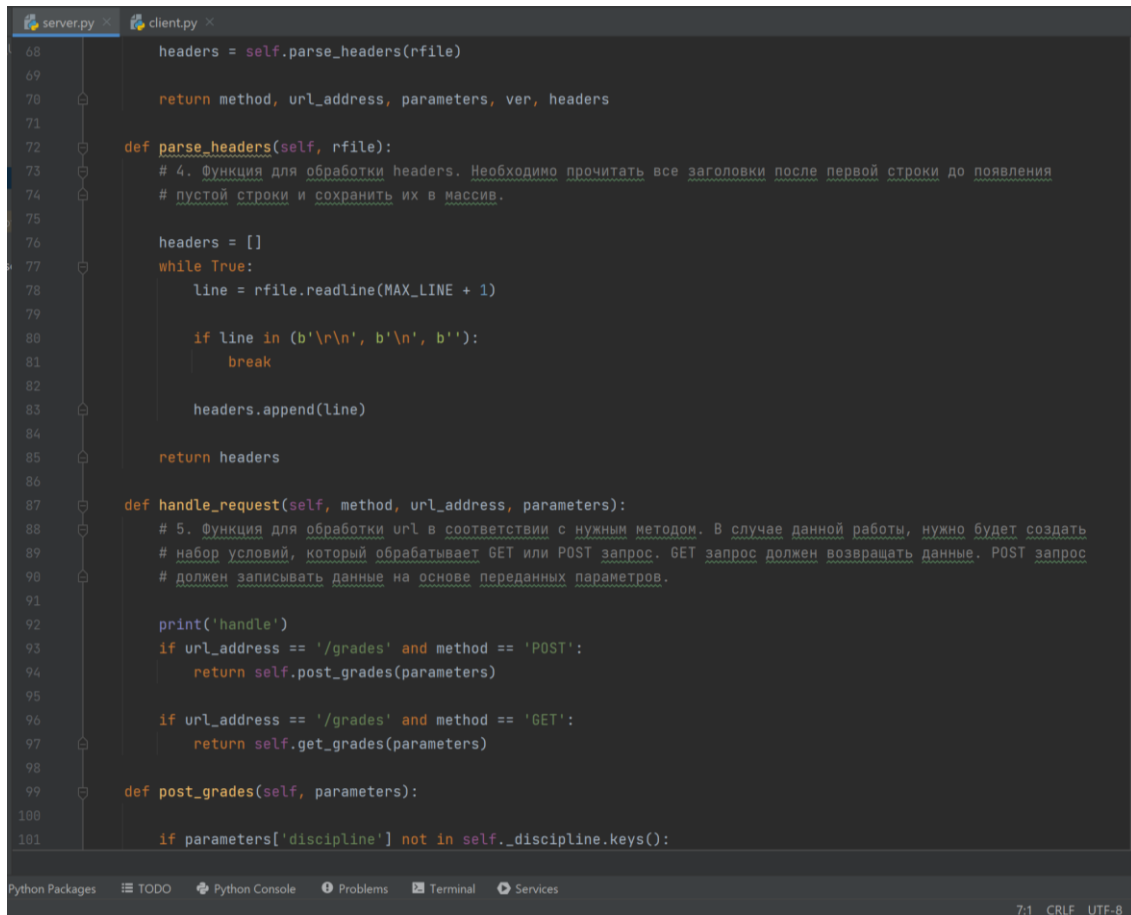


```
server.py client.py
1 import socket
2 import sys
3
4 MAX_LINE = 64 * 1024
5 MAX_HEADERS = 100
6
7
8 class MyHTTPServer:
9     # Параметры сервера
10     def __init__(self, host, port, server_name):
11         self._host = host
12         self._port = port
13         self._server_name = server_name
14
15         self._discipline = {}
16
17     def serve_forever(self):
18         # 1. Запуск сервера на сокете, обработка входящих соединений
19         serv_socket = socket.socket(
20             socket.AF_INET,
21             socket.SOCK_STREAM
22         )
23
24         try:
25             serv_socket.bind((self._host, self._port))
26             serv_socket.listen()
27
28             while True:
29                 conn, _ = serv_socket.accept()
30                 try:
31                     self.serve_client(conn)
32                 except Exception as e:
33                     print('Client serving failed', e)
34             finally:
35                 serv_socket.close()
36
37     def serve_client(self, conn):
38         # 2. Обработка клиентского подключения
39
40         method, url_address, parameters, ver, headers = self.parse_request(conn)
41         resp = self.handle_request(method, url_address, parameters)
42         self.send_response(conn, resp)
43
44     def parse_request(self, conn):
45         # 3. функция для обработки заголовка http-запроса. Python, сокет предоставляет возможность создать вокруг
46         # него некоторую обертку, которая предоставляет file object интерфейс. Это дает возможность построчно
47         # обрабатывать запрос. Заголовок всегда - первая строка. Первую строку нужно разбить на 3 элемента (метод +
48         # url + версия протокола). URL необходимо разбить на адрес и параметры (isu.ifmo.ru/pls/apex/f?p=2143 ,
49         # где isu.ifmo.ru/pls/apex/f, а p=2143 - параметр p со значением 2143)
50
51         rfile = conn.makefile('rb')
52         raw = rfile.readline(MAX_LINE + 1)
53
54         req_line = str(raw, 'iso-8859-1')
55         req_line = req_line.rstrip('\r\n')
56         words = req_line.split()
57         method, url, ver = words
58
59         url_address, parameters = url.split('?')
60
61         pairs = parameters.split('&')
62
63         parameters = {}
64         for pair in pairs:
65             key, value = pair.split('=')
66             parameters[key] = value
67
68         return method, url_address, parameters, ver, headers
```

В обработку самого клиентского подключения входит парсинг заголовка, составление ответа и его отправка. Парсинг заголовка осуществляется следующим образом: получаются данные запроса, разбиваются на метод, адрес и версию протокола, после чего адрес разбивается на конкретно адрес и переданные параметры, которые в свою очередь оформляются в словарь.

```
server.py client.py
34 finally:
35     serv_socket.close()
36
37 def serve_client(self, conn):
38     # 2. Обработка клиентского подключения
39
40     method, url_address, parameters, ver, headers = self.parse_request(conn)
41     resp = self.handle_request(method, url_address, parameters)
42     self.send_response(conn, resp)
43
44 def parse_request(self, conn):
45     # 3. функция для обработки заголовка http-запроса. Python, сокет предоставляет возможность создать вокруг
46     # него некоторую обертку, которая предоставляет file object интерфейс. Это дает возможность построчно
47     # обрабатывать запрос. Заголовок всегда - первая строка. Первую строку нужно разбить на 3 элемента (метод +
48     # url + версия протокола). URL необходимо разбить на адрес и параметры (isu.ifmo.ru/pls/apex/f?p=2143 ,
49     # где isu.ifmo.ru/pls/apex/f, а p=2143 - параметр p со значением 2143)
50
51     rfile = conn.makefile('rb')
52     raw = rfile.readline(MAX_LINE + 1)
53
54     req_line = str(raw, 'iso-8859-1')
55     req_line = req_line.rstrip('\r\n')
56     words = req_line.split()
57     method, url, ver = words
58
59     url_address, parameters = url.split('?')
60
61     pairs = parameters.split('&')
62
63     parameters = {}
64     for pair in pairs:
65         key, value = pair.split('=')
66         parameters[key] = value
67
68     return method, url_address, parameters, ver, headers
```

В конце парсинга запроса происходит парсинг заголовков, который описан в следующей функции. В ее цикле построчно читается файл, переданный из предыдущей функции и до появления пустой строки заголовки записываются в соответствующий массив. После идет функция для создания ответа. В зависимости от переданного метода она вызывает либо функцию записи новых оценок, либо функцию получения информации по дисциплине.



```
server.py client.py
68 headers = self.parse_headers(rfile)
69
70 return method, url_address, parameters, ver, headers
71
72 def parse_headers(self, rfile):
73     # 4. Функция для обработки headers. Необходимо прочитать все заголовки после первой строки до появления
74     # пустой строки и сохранить их в массив.
75
76     headers = []
77     while True:
78         line = rfile.readline(MAX_LINE + 1)
79
80         if line in (b'\r\n', b'\n', b''):
81             break
82
83         headers.append(line)
84
85     return headers
86
87 def handle_request(self, method, url_address, parameters):
88     # 5. Функция для обработки url в соответствии с нужным методом. В случае данной работы, нужно будет создать
89     # набор условий, который обрабатывает GET или POST запрос. GET запрос должен возвращать данные. POST запрос
90     # должен записывать данные на основе переданных параметров.
91
92     print('handle')
93     if url_address == '/grades' and method == 'POST':
94         return self.post_grades(parameters)
95
96     if url_address == '/grades' and method == 'GET':
97         return self.get_grades(parameters)
98
99 def post_grades(self, parameters):
100
101     if parameters['discipline'] not in self._discipline.keys():
```

В функции записи информации проверяется наличие необходимого ключа (переданной дисциплины), при необходимости он создается, происходит добавление данных в словарь, объявленный при инициализации и возвращаются атрибуты ответа. Если же нужно получить данные, то построчно создается тело ответа с добавлением необходимых данных в соответствии с переданными параметрами и также возвращаются атрибуты ответа.

```
server.py client.py
101 if parameters['discipline'] not in self._discipline.keys():
102     self._discipline[parameters['discipline']] = {}
103
104 self._discipline[parameters['discipline']][parameters['name']] = parameters['grade']
105
106 print(self._discipline)
107
108 return [204, 'Created']
109
110
111 def get_grades(self, parameters):
112     content_type = 'text/html; charset=utf-8'
113
114     body = '<html><head></head><body>'
115     body += f'<div>Дисциплина ({parameters["discipline"]})</div>'
116     body += '<ul>'
117     for u in self._discipline[parameters["discipline"]].keys():
118         body += f'<li>#{{u}} {self._discipline[parameters["discipline"]][u]}</li>'
119     body += '</ul>'
120     body += '</body></html>'
121
122     body = body.encode('utf-8')
123     headers = [('Content-Type', content_type),
124               ('Content-Length', len(body))]
125     return [200, 'OK', headers, body]
126
127 def send_response(self, conn, resp):
128     # 6. Функция для отправки ответа. Необходимо записать в соединение status line вида HTTP/1.1 <status_code>
129     # <reason>. Затем, построчно записать заголовки и пустую строку, обозначающую конец секции заголовков.
130
131     wfile = conn.makefile('wb')
132     status_line = f'HTTP/1.1 {resp[0]} {resp[1]}\r\n'
133     wfile.write(status_line.encode('iso-8859-1'))
134
135     if len(resp) > 2:
136         for (key, value) in resp[2]:
137             header_line = f'{key}: {value}\r\n'
138             wfile.write(header_line.encode('iso-8859-1'))
139
140     wfile.write(b'\r\n')
141
142     if len(resp) > 3:
143         wfile.write(resp[3])
144
145     wfile.flush()
146     wfile.close()
147
148 host = socket.gethostname()
149 port = 3030
150 name = 'myserver'
151 serv = MyHTTPServer(host, port, name)
152 try:
153     serv.serve_forever()
154 except KeyboardInterrupt:
155     pass
156
```

В функции отправки ответа создается уже сам файл, который будет отправлен клиенту, в него в необходимом порядке вставляются атрибуты из предыдущих функций.

```
server.py client.py
125
126 def send_response(self, conn, resp):
127     # 6. Функция для отправки ответа. Необходимо записать в соединение status line вида HTTP/1.1 <status_code>
128     # <reason>. Затем, построчно записать заголовки и пустую строку, обозначающую конец секции заголовков.
129
130     wfile = conn.makefile('wb')
131     status_line = f'HTTP/1.1 {resp[0]} {resp[1]}\r\n'
132     wfile.write(status_line.encode('iso-8859-1'))
133
134     if len(resp) > 2:
135         for (key, value) in resp[2]:
136             header_line = f'{key}: {value}\r\n'
137             wfile.write(header_line.encode('iso-8859-1'))
138
139     wfile.write(b'\r\n')
140
141     if len(resp) > 3:
142         wfile.write(resp[3])
143
144     wfile.flush()
145     wfile.close()
146
147
148 host = socket.gethostname()
149 port = 3030
150 name = 'myserver'
151 serv = MyHTTPServer(host, port, name)
152 try:
153     serv.serve_forever()
154 except KeyboardInterrupt:
155     pass
156
```

Для клиента код простой состоит из отправки данных на сервер и их же получения.

```
server.py x client.py x
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 s.connect((socket.gethostname(), 3030))
6
7 msg1 = 'POST /grades?discipline=CV&name=Vasia&grade=4 HTTP/1.1\r\nHost: example.local\r\nAccept: text/html\r\nUser-Agent: Mozilla/5.0\r\n\r\n'
8 s.send(msg1.encode('iso-8859-1'))
9 msg4 = s.recv(1024)
10 print(msg4.decode())
11 s.close()
12
13 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14
15 s.connect((socket.gethostname(), 3030))
16
17 msg2 = 'GET /grades?discipline=CV HTTP/1.1\r\nHost: example.local\r\nAccept: text/html\r\nUser-Agent: Mozilla/5.0\r\n\r\n'
18 s.send(msg2.encode('iso-8859-1'))
19 msg = s.recv(1024)
20 print(msg.decode())
21 s.close()
22
```

При запуске клиента в консоль выводятся следующие ответы.

```
server x client x
C:\Users\Oksana\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Oksana\PycharmProjects\pythonProject\Task5\client.py
HTTP/1.1 204 Created

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 100

<html><head></head><body><div>Дисциплина (CV)</div><ul><li>#Vasia 4</li></ul></body></html>

Process finished with exit code 0
```