

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1
по теме: **Работа с сокетами**
по дисциплине: Web-программирование

Специальность:
09.03.03 Мобильные и сетевые технологии

Выполнил:
студент группы К33401
Дорофеева А. Д.

Санкт-Петербург 2021/2022

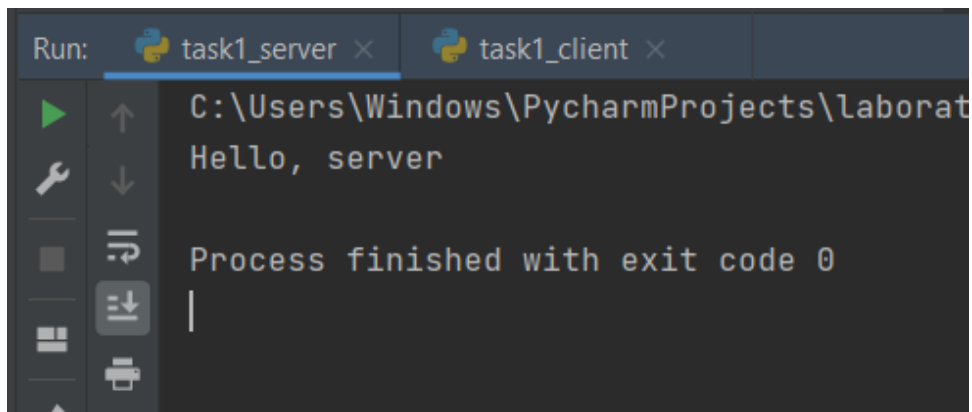
Цель работы: овладеть практическими навыками и умениями реализации web-серверов и использования сокетов.

Практическое задание

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.

```
task1_server.py × task1_client.py ×
1  import socket
2
3
4  class Server:
5      def __init__(self, port: int):
6          self.sock = socket.socket()          # create socket
7          self.sock.bind(('127.0.0.1', port))  # open socket
8          self.sock.listen(1)                  # open connection queue
9
10     def speak(self, msg):
11         conn, port = self.sock.accept()      # accept connection
12         data = conn.recv(1024)               # receive data
13         print(data.decode('utf-8'))          # print message
14         conn.send(msg.encode('utf-8'))       # send message
15         conn.close()                         # close socket
16
17
18  if __name__ == '__main__':
19      server = Server(14900)
20      server.speak('Hello, client')
21
```

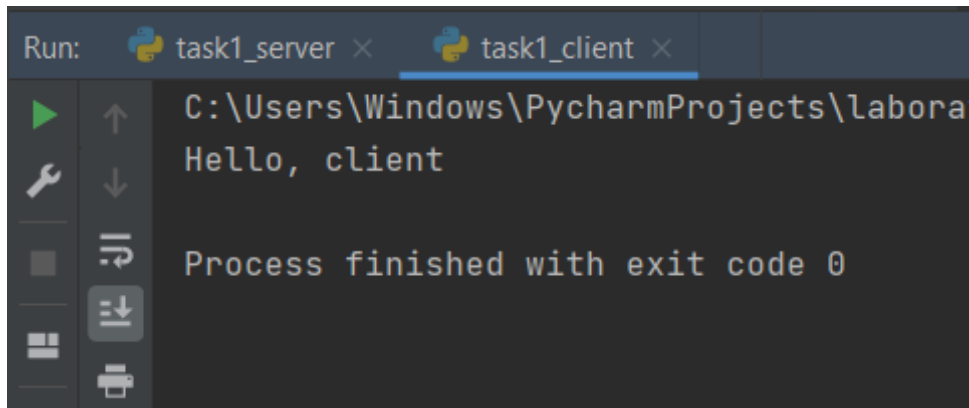
```
task1_server.py × task1_client.py ×
1  import socket
2
3
4  class Client:
5      def __init__(self, port: int):
6          self.conn = socket.socket()          # create socket
7          self.conn.connect(('127.0.0.1', port)) # connect server
8
9      def speak(self, msg):
10         self.conn.send(msg.encode('utf-8'))   # send message
11         data = self.conn.recv(1024)           # receive data
12         print(data.decode('utf-8'))           # print message
13         self.conn.close()                     # close connection
14
15
16  if __name__ == '__main__':
17      client = Client(14900)
18      client.speak('Hello, server')
19
```



Run: task1_server × task1_client ×

```
C:\Users\Windows\PycharmProjects\laborat
Hello, server

Process finished with exit code 0
```



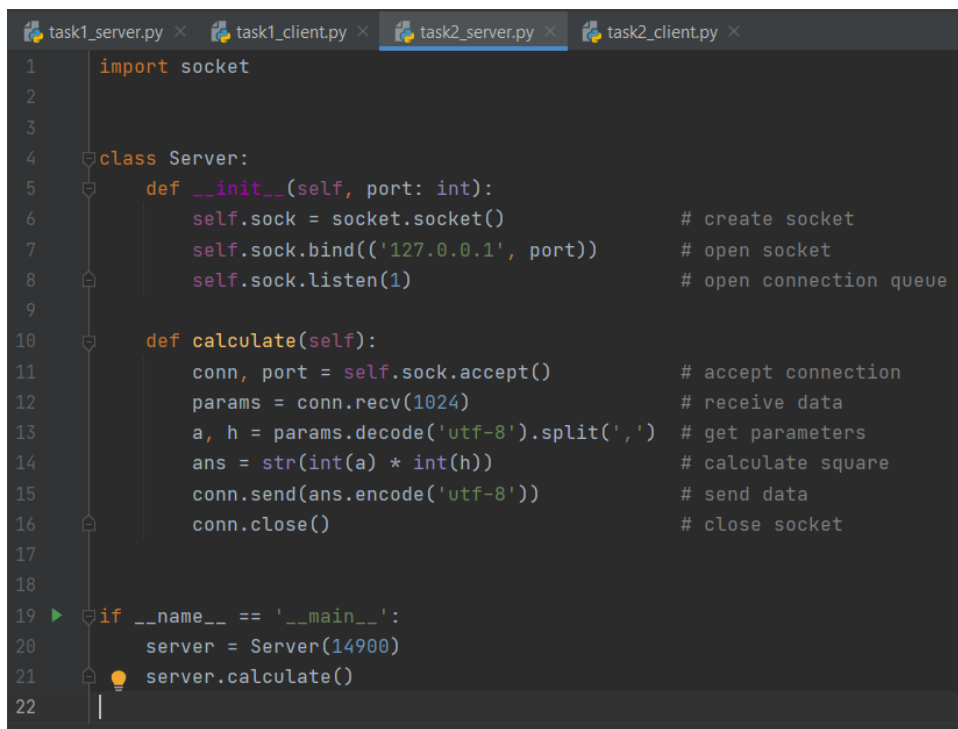
Run: task1_server × task1_client ×

```
C:\Users\Windows\PycharmProjects\labora
Hello, client

Process finished with exit code 0
```

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: d. Поиск площади параллелограмма.



```
task1_server.py × task1_client.py × task2_server.py × task2_client.py ×
1 import socket
2
3
4 class Server:
5     def __init__(self, port: int):
6         self.sock = socket.socket()           # create socket
7         self.sock.bind(('127.0.0.1', port))    # open socket
8         self.sock.listen(1)                   # open connection queue
9
10    def calculate(self):
11        conn, port = self.sock.accept()        # accept connection
12        params = conn.recv(1024)               # receive data
13        a, h = params.decode('utf-8').split(',') # get parameters
14        ans = str(int(a) * int(h))              # calculate square
15        conn.send(ans.encode('utf-8'))         # send data
16        conn.close()                           # close socket
17
18
19 if __name__ == '__main__':
20     server = Server(14900)
21     server.calculate()
22
```

```
task1_server.py × task1_client.py × task2_server.py × task2_client.py ×
import socket

class Client:
    def __init__(self, port: int):
        self.conn = socket.socket()          # create socket
        self.conn.connect(('127.0.0.1', port)) # connect server

    def calculate(self, a: int, h: int):
        self.conn.sendall(bytes(f'{a},{h}', 'utf-8')) # send data
        data = self.conn.recv(1024)                 # receive data
        print('the square is:', data.decode('utf-8')) # print answer
        self.conn.close()                            # close connection

if __name__ == '__main__':
    client = Client(14900)
    client.calculate(int(input('input a: ')), int(input('input h: ')))
```

```
task2_server × task2_client ×
C:\Users\Windows\PycharmProjects\laboratory_work_1
input a: 15
input h: 11
the square is: 165
Process finished with exit code 0
```

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.

```
index.html × task3_server.py × task3_client.py ×
1 import socket
2
3
4 class Server:
5     def __init__(self, port: int):
6         self.sock = socket.socket()          # create socket
7         self.sock.bind(('127.0.0.1', port))  # open socket
8         self.sock.listen(1)                  # open connection queue
9
10    def load(self):
11        conn, port = self.sock.accept()       # accept connection
12        with open('index.html', 'r') as file: # open file
13            response_type = 'HTTP/1.0 200 OK\r\n' # response type
14            headers = 'Content-Type: text/html\r\n' # headers
15            body = file.read()                 # read data
16            response = response_type + headers + body # form response
17            conn.send(response.encode('utf-8')) # send data
18            conn.close()                       # close connection
19
20
21 if __name__ == '__main__':
22     server = Server(14900)
23     server.load()
24
```

```
index.html x task3_server.py x task3_client.py x
1 import socket
2
3
4 class Client:
5     def __init__(self, port: int):
6         self.conn = socket.socket() # create socket
7         self.conn.connect(('127.0.0.1', port)) # connect server
8
9     def get(self):
10        self.conn.send(bytes(f'.', 'utf-8')) # send data
11        data = self.conn.recv(1024) # receive data
12        print(data.decode('utf-8')) # print message
13        self.conn.close() # close connection
14
15
16 if __name__ == '__main__':
17     client = Client(14900)
18     client.get()
19
```

```
index.html x task3_server.py x task3_client.py x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>hello</title>
6 </head>
7 <body>
8
9 </body>
10 </html>
```

```
Run: task3_server x task3_client x
C:\Users\Windows\PycharmProjects\laboratory_work_1\venv\Scripts\python.exe
HTTP/1.0 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>hello</title>
</head>
<body>

</body>
</html>

Process finished with exit code 0
```

4. Реализовать двухпользовательский или многопользовательский чат.

```
task4_server.py task4_client0.py task4_client1.py task4_client2.py
1 import socket
2
3
4 class Server:
5     def __init__(self, port: int):
6         self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # create socket
7         self.sock.bind(('127.0.0.1', port)) # open socket
8
9     def run(self):
10        clients = [] # create users array
11        while True:
12            data, address = self.sock.recvfrom(1024) # receive data
13            print(address[0], address[1]) # users' addresses
14            if address not in clients:
15                clients.append(address) # add new user
16            for client in clients:
17                if client == address:
18                    continue
19                self.sock.sendto(data, client) # send data
20
21
22 if __name__ == '__main__':
23     server = Server(14900)
24     server.run()
25
```

```
task4_server.py task4_client0.py task4_client1.py task4_client2.py
4
5 class Client:
6     def __init__(self, port: int):
7         self.server = '127.0.0.1', port # server data
8         self.conn = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # create socket
9         self.conn.bind(('', 0)) # open socket
10
11     def read(self):
12        while True:
13            data = self.conn.recv(1024)
14            print(data.decode('utf-8'))
15
16     def chat(self):
17        print("enter your nickname to start chatting: ")
18        nickname = input() # get nickname
19        self.conn.sendto((nickname + ' is online').encode('utf-8'), self.server) # notify everyone
20
21        while True:
22            self.conn.sendto((nickname + ': ' + input()).encode('utf-8'), self.server) # send message
23
24
25
26 if __name__ == '__main__':
27     client = Client(14900)
28     thread1, thread2 = threading.Thread(target=client.read), threading.Thread(target=client.chat)
29     thread1.start(), thread2.start()
30
```

```
Run: task4_server task4_client0 task4_client1 task4_client2
C:\Users\Windows\PycharmProjects\laboratory_work_1\venv\Scripts\p
127.0.0.1 52177
127.0.0.1 59251
127.0.0.1 59252
127.0.0.1 52177
127.0.0.1 59252
127.0.0.1 59251
127.0.0.1 52177
127.0.0.1 59252
127.0.0.1 52177
```

```
Run: task4_server x task4_client0 x task4_client1 x task4_client2 x
C:\Users\Windows\PycharmProjects\laboratory_work_1\venv\Scripts\python.
enter your nickname to start chatting:
kuriyummy
vasya is online
axolotlya is online
hello everyone!!!
axolotlya: aaaaaaaaaooooooooaaa
vasya: loooo1111
xD
axolotlya: dfyuyuiuopliyktfjtyedysdtyuio
vasya: this is fine
```

```
Scratches and Consoles
Run: task4_server x task4_client0 x task4_client1 x task4_client2 x
C:\Users\Windows\PycharmProjects\laboratory_work_1\venv\Scripts\python.exe
enter your nickname to start chatting:
vasya
axolotlya is online
kuriyummy: hello everyone!!!
axolotlya: aaaaaaaaaooooooooaaa
loooo1111
kuriyummy: xD
axolotlya: dfyuyuiuopliyktfjtyedysdtyuio
this is fine
```

```
Run: task4_server x task4_client0 x task4_client1 x task4_client2 x
C:\Users\Windows\PycharmProjects\laboratory_work_1\venv\Script
enter your nickname to start chatting:
axolotlya
kuriyummy: hello everyone!!!
aaaaaaaaooooooooaaa
vasya: loooo1111
kuriyummy: xD
dfyuyuiuopliyktfjtyedysdtyuio
vasya: this is fine
```