***Сокеты***

**Экран поддержки**

Реализуйте передачу данных с нескольких клиент-сокетов на один сервер-сокет. Внутри одного скрипта следует создать несколько клиент-сокетов, которые будут отсылать сообщения из списка сообщений на сервер-сокет.

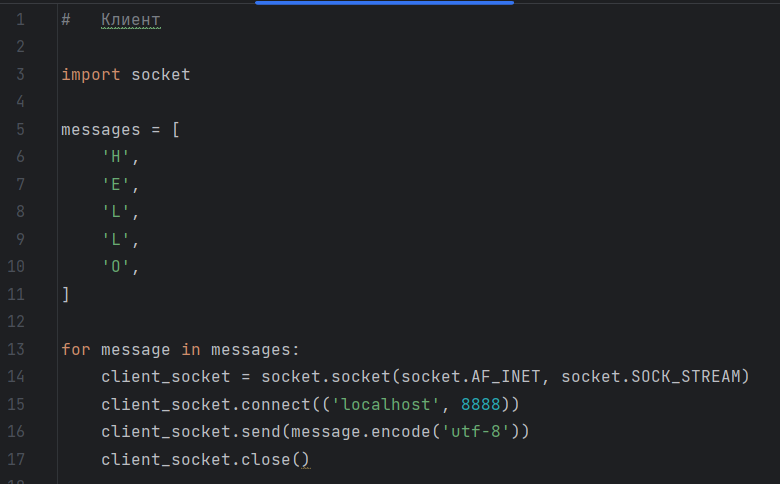


Рисунок – Клиентский скрипт с массивом данных



Рисунок – Серверный скрипт

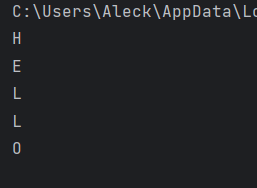


Рисунок – Полученный массив данных

client\_dispay\_helper.py

# Клиент

import socket

messages = [

'H',

'E',

'L',

'L',

'O',

]

for message in messages:

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('localhost', 8888))

client\_socket.send(message.encode('utf-8'))

client\_socket.close()

server\_display\_helper.py

# Сервер

import socket

import threading

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('localhost', 8888))

server\_socket.listen(5)

def handle\_client(client\_socket):

while True:

data = client\_socket.recv(1024)

if not data:

break

print(data.decode('utf-8'))

client\_socket.close()

while True:

client\_socket, addr = server\_socket.accept()

client\_thread = threading.Thread(target=handle\_client, args=(client\_socket,))

client\_thread.start()

**Передача объектов**

Реализуйте передачу объекта между клиентом и сервером. Сначала необходимо будет реализовать класс с несколькими свойствами и функциями и поместить его в модуль. Со стороны клиент-сокета нужно обработать объект и передать его сервер-сокету. Со стороны сервер-сокета проведите несколько методов и отобразите его свойства.

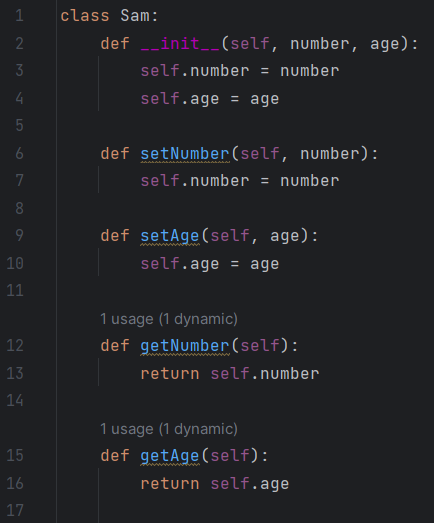


Рисунок 4 – Класс

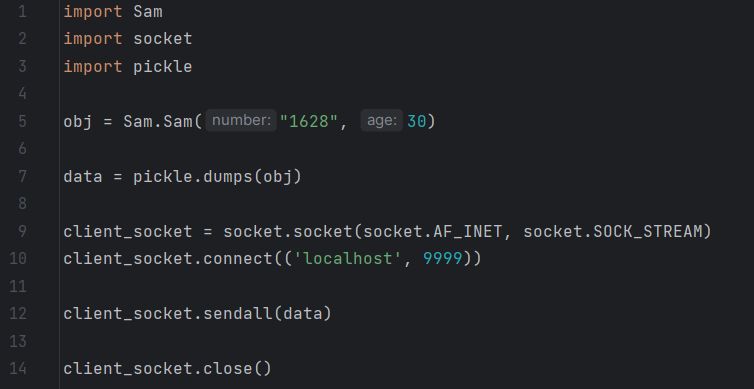


Рисунок 5 – Клиент



Рисунок 6 – Сервер

client.py

import Sam

import socket

import pickle

obj = Sam.Sam("1628", 30)

data = pickle.dumps(obj)

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('localhost', 9999))

client\_socket.sendall(data)

client\_socket.close()

Sam.py

class Sam:

def \_\_init\_\_(self, number, age):

self.number = number

self.age = age

def setNumber(self, number):

self.number = number

def setAge(self, age):

self.age = age

def getNumber(self):

return self.number

def getAge(self):

return self.age

server.py

import socket

import pickle

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('localhost', 9999))

server\_socket.listen(1)

client\_socket, address = server\_socket.accept()

data = client\_socket.recv(4096)

obj = pickle.loads(data)

print("Number:", obj.getNumber())

print("Age:", obj.getAge())

client\_socket.close()

server\_socket.close()

**Полудуплекс**

Реализуйте скрипт с сокетом, который может быть сервер-сокетом или клиент-сокетом, по желанию пользователя. С помощью такого сокета передайте данные другому такому же сокету и наоборот.



Рисунок 7 – Первый сервер

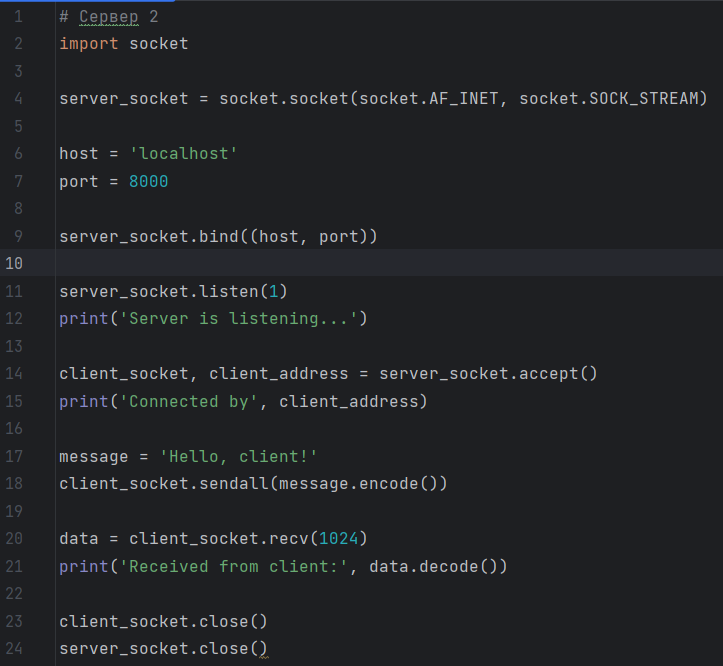


Рисунок 8 – Второй сервер

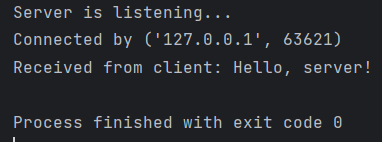


Рисунок 9 – Первый сервер

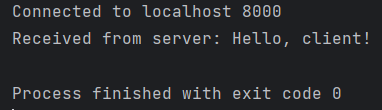


Рисунок 10 – Второй сервер

first\_server.py

# Сервер 2

import socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

host = 'localhost'

port = 8000

server\_socket.bind((host, port))

server\_socket.listen(1)

print('Server is listening...')

client\_socket, client\_address = server\_socket.accept()

print('Connected by', client\_address)

message = 'Hello, client!'

client\_socket.sendall(message.encode())

data = client\_socket.recv(1024)

print('Received from client:', data.decode())

client\_socket.close()

server\_socket.close()

second\_server.py

# Сервер 1

import socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

host = 'localhost'

port = 8000

client\_socket.connect((host, port))

print('Connected to', host, port)

data = client\_socket.recv(1024)

print('Received from server:', data.decode())

message = 'Hello, server!'

client\_socket.sendall(message.encode())

client\_socket.close()

**Обработка HTTP**

Отправьте на сайте (например на example.org) HTTP-запрос и по полученному HTTP-ответу из тела достаньте значение из тега <title>.



Рисунок – Код обработки html/content

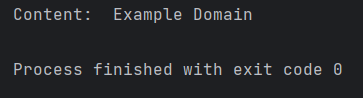


Рисунок 12 – Результат

Only\_tag.py

from urllib import request

responce = request.urlopen("http://example.com")

html = responce.read().decode()

html\_str = str(html)

start = html\_str.find('<title>')

end = html\_str.find('</title>')

char\_count = len('<title>')

print('Content: ', html\_str[start+char\_count:end])

Итог по всей проделанной работе следует оформить в виде файла Word (.docx) с подробными скриншотами (win+shift+s) результата работы программы по всем заданиями. Также в конце отчета следует добавить код вашей программы.