Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

“Российский экономический университет имени Г.В. Плеханова”

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

Специальность: 09.02.07 Информационные системы и программирование  
Квалификация: программист

ПРАКТИЧЕСКИЕ РАБОТЫ ПО КУРСУ

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON

Листов: 26

Выполнил студент

Группы: 18.01-15-05

Игошев Ростислав Вадимович

Москва 2023

ПРАКТИЧЕСКАЯ РАБОТА №1

«Вывод данных термометра»

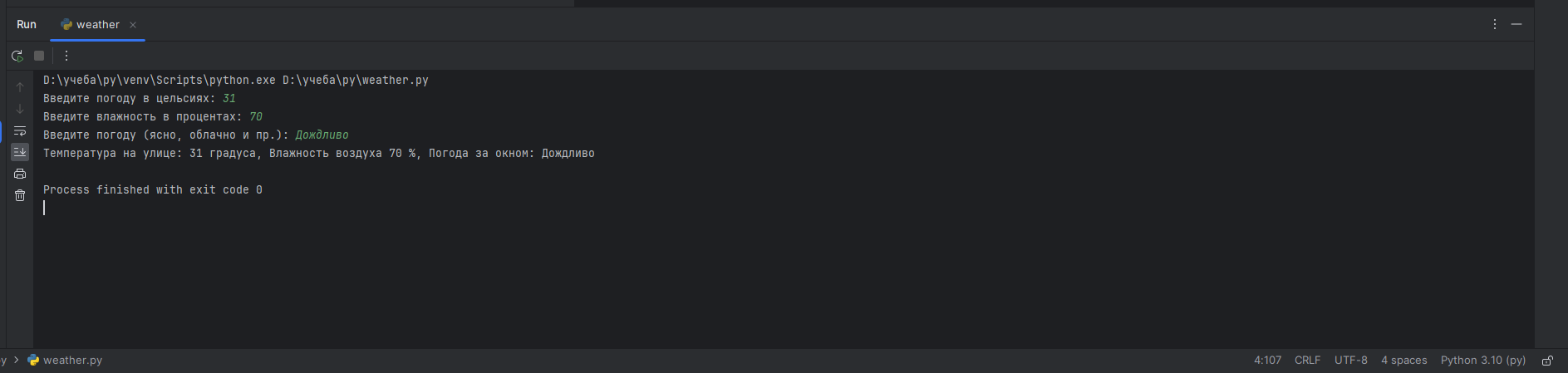


Рисунок 1 – Результат работы вывода погоды

Код программы:

celsius = input("Введите погоду в цельсиях: ")

water = input("Введите влажность в процентах: ")

weather = input("Введите погоду (ясно, облачно и пр.): ")

print(f"Температура на улице: {celsius} градуса, Влажность воздуха {water} %, Погода за окном: {weather}")

ПРАКТИЧЕСКАЯ РАБОТА №2

«Калькулятор»

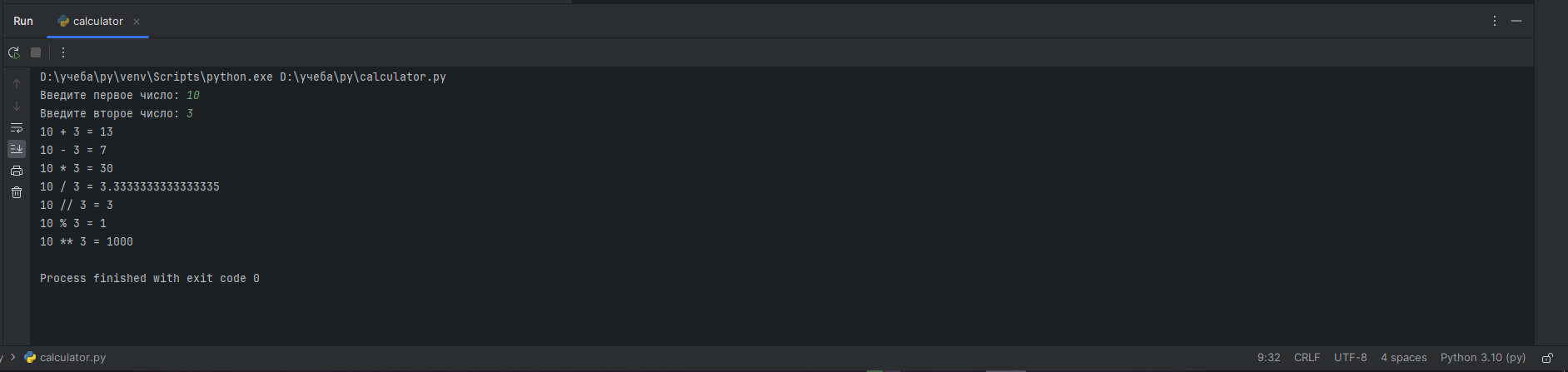


Рисунок 2 – Результат работы калькулятора

Код работы:

a = int(input("Введите первое число: "))

b = int(input("Введите второе число: "))

print(f"{a} + {b} = {a + b}")

print(f"{a} - {b} = {a - b}")

print(f"{a} \* {b} = {a \* b}")

print(f"{a} / {b} = {a / b}")

print(f"{a} // {b} = {a // b}")

print(f"{a} % {b} = {a % b}")

print(f"{a} \*\* {b} = {a \*\* b}")

ПРАКТИЧЕСКАЯ РАБОТА №3

«Расчет геометрических фигур»

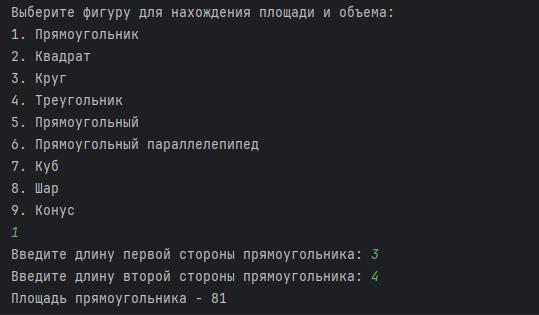


Рисунок 3 – Нахождение площади прямоугольника

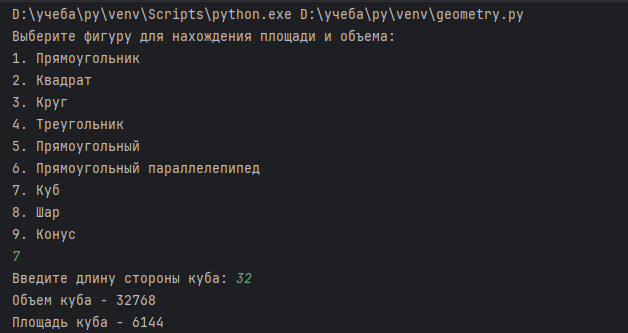


Рисунок 4 – Нахождение площади и объема куба

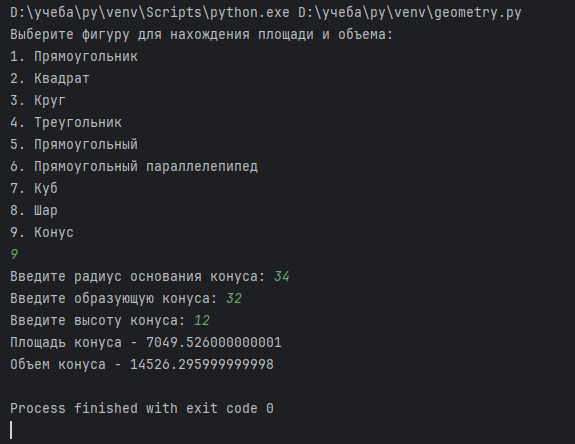


Рисунок 5 – Нахождение площади и объема конуса



Рисунок 6 – Нахождение площади и объема шара

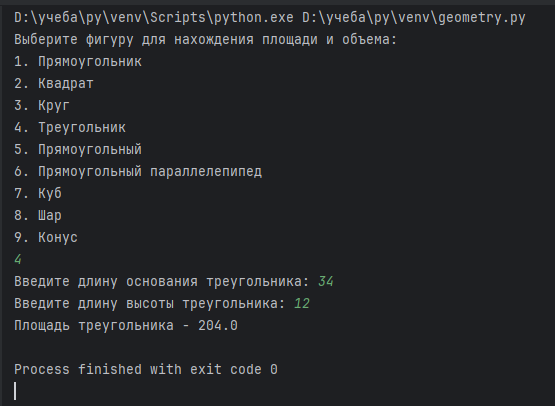


Рисунок 7 – Нахождение площади треугольника

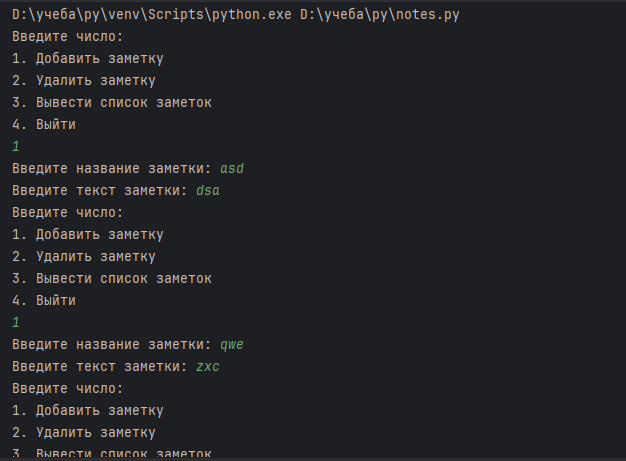


Рисунок 8 – Добавление заметок

Код программы:

pi = 3.1415

print(f"Выберите фигуру для нахождения площади и объема: \n1. Прямоугольник\n2. Квадрат\n3. Круг\n4. Треугольник\n5. Прямоугольный \n6. Прямоугольный параллелепипед \n7. Куб\n8. Шар\n9. Конус")

choice = int(input())

match(choice):

case 1:

a = int(input("Введите длину первой стороны прямоугольника: "))

b = int(input("Введите длину второй стороны прямоугольника: "))

print(f"Площадь прямоугольника - {a \*\* b}")

case 2:

a = int(input("Введите длину первой стороны квадрата: "))

b = int(input("Введите длину второй стороны квадрата: "))

print(f"Площадь квадрата - {a \* b}")

case 3:

r = int(input("Введите радиус круга: "))

print(f"Площадь круга - {pi \* r\*\*2}")

case 4:

a = int(input("Введите длину основания треугольника: "))

h = int(input("Введите длину высоты треугольника: "))

print(f"Площадь треугольника - {a / 2 \* h}")

case 5:

a = int(input("Введите длину первого катета прямоугольного треугольника: "))

b = int(input("Введите длину второго катета прямоугольного треугольника: "))

print(f"Площадь прямоугольного треугольника - {a / 2 \* b}")

case 6:

a = int(input("Введите длину первой стороны параллелепипеда: "))

b = int(input("Введите длину второй стороны параллелепипеда: "))

c = int(input("Введите длину третьей стороны параллелепипеда: "))

print(f"Площадь параллелепипеда - {2 \* (a \* b + a \* с + b \* c)}")

print(f"Объем параллелепипеда - { a \* b \* c}")

case 7:

a = int(input("Введите длину стороны куба: "))

print(f"Объем куба - {a \*\* 3}")

print(f"Площадь куба - {6 \* a \*\* 2}")

case 8:

r = int(input("Введите радиус шара: "))

print(f"Площадь шара - {4 \* pi \* r \*\* 2}")

print(f"Объем шара - {4/3 \* pi \* r \*\* 3}")

case 9:

r = int(input("Введите радиус основания конуса: "))

l = int(input("Введите образующую конуса: "))

h = int(input("Введите высоту конуса: "))

print(f"Площадь конуса - {pi \* r \* (r + l)}")

print(f"Объем конуса - {(1/3) \* pi \* r\*\*2 \* h}")

case \_:

print("Выбранного пункта в списке нет.")

ПРАКТИЧЕСКАЯ РАБОТА №4

«Заметки»

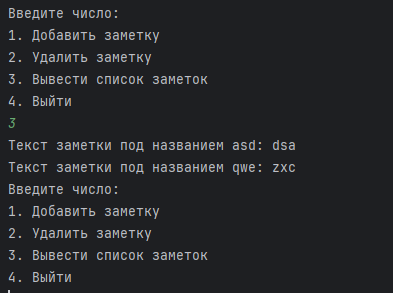


Рисунок 9 – Вывод заметок

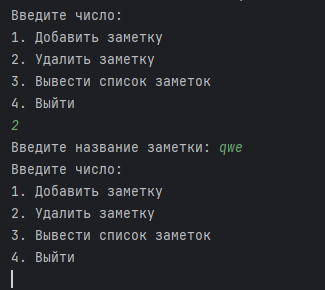


Рисунок 10 – Удаление заметок

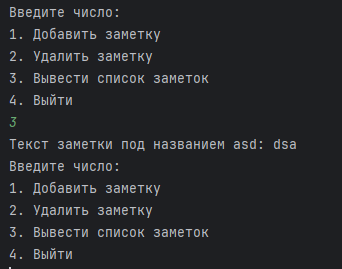


Рисунок 11 – Демонстрация удаления заметок

Код программы:

choice = 0

zametki = {}

def main():

while(True):

print("Введите число:")

print("1. Добавить заметку")

print("2. Удалить заметку")

print("3. Вывести список заметок")

print("4. Выйти")

choice = input()

match(choice):

case '1':

add()

case '2':

delete()

case '3':

list()

case '4':

exit(0)

def add():

name = input("Введите название заметки: ")

txt = input("Введите текст заметки: ")

zametki[name] = txt

def delete():

name = input("Введите название заметки: ")

if zametki.pop(name, 666) == 666:

print("Такая заметка не найдена.")

def list():

for key, value in zametki.items():

print(f'Текст заметки под названием {key}: {value}')

if \_\_name\_\_ == '\_\_main\_\_':

main()

ПРАКТИЧЕСКАЯ РАБОТА №5  
Тема: «Функции, модули и классы»

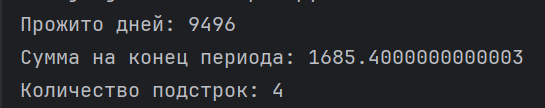


Рисунок 12 – Результат работы функций

Код:

import string\_operations  
import arithmetic  
  
  
def calculate\_age\_in\_days(full\_age):  
 current\_year = 2023  
 birth\_year = current\_year - full\_age  
  
 def is\_leap\_year(year):  
 return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)  
  
 days = 0  
 for year in range(birth\_year, current\_year + 1):  
 if is\_leap\_year(year):  
 days += 366  
 else:  
 days += 365  
 return days  
  
  
def hard\_procent(P=1000, r=0.05, n=1, t=1):  
 result = P \* ((1 + r / n) \*\* (n \* t))  
 return result  
  
  
def count\_substrings(main\_string, sub\_string):  
 return main\_string.count(sub\_string)  
  
  
def main():  
 age\_in\_days = calculate\_age\_in\_days(25)  
 print("Прожито дней:", age\_in\_days)  
 interest\_amount = hard\_procent(1500, 0.06, 1, 2)  
 print("Сумма на конец периода:", interest\_amount)  
 substring\_count = count\_substrings('abababab', 'ab')  
 print("Количество подстрок:", substring\_count)  
  
 discriminant = arithmetic.calculate\_discriminant(1, -3, 2)  
 print("Дискриминант:", discriminant)  
 roots = arithmetic.calculate\_roots(1, -3, 2)  
 print("Корни уравнения:", roots)  
 joint\_probability = arithmetic.calculate\_probability\_of\_joint\_events(0.5, 0.5)  
 print("Вероятность совместных событий:", joint\_probability)  
 independent\_probability = arithmetic.calculate\_probability\_of\_independent\_events(0.5, 0.5)  
 print("Вероятность несовместных событий:", independent\_probability)  
  
 char\_index = string\_operations.find\_character\_in\_string('l', 'Hello')  
 print("Индекс символа:", char\_index)  
 uppercase\_letters = string\_operations.get\_uppercase\_letters("Hello, World!")  
 print("Кол-во заглавных символов:", uppercase\_letters)  
 lowercase\_letters = string\_operations.get\_lowercase\_letters("Hello, World!")  
 print("Кол-во строчных символов:", lowercase\_letters)  
 numeric\_characters = string\_operations.get\_numeric\_characters("Hello 123")  
 print("Кол-во числовых символов:", numeric\_characters)  
 result1 = string\_operations.is\_uppercase('H')  
 print("Это заглавная буква? - ", result1)  
 result2 = string\_operations.is\_lowercase('h')  
 print("Это строчная буква? - ", result2)  
 result3 = string\_operations.is\_numeric('5')  
 print("Это цифра? - ", result3)  
 substring\_count = string\_operations.count\_substrings('abababab', 'ab')  
 print("Количество подстрок:", substring\_count)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

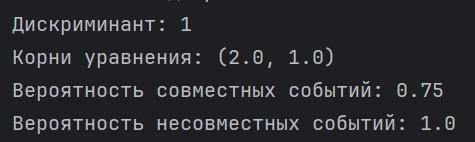


Рисунок 13 – Результат работы модуля математики

import math  
  
def calculate\_discriminant(a, b, c):  
 discriminant = b \*\* 2 - 4 \* a \* c  
 return discriminant  
  
def calculate\_roots(a, b, c):  
 discriminant = calculate\_discriminant(a, b, c)  
 if discriminant > 0:  
 x1 = (-b + math.sqrt(discriminant)) / (2 \* a)  
 x2 = (-b - math.sqrt(discriminant)) / (2 \* a)  
 return x1, x2  
 elif discriminant == 0:  
 x = -b / (2 \* a)  
 return x  
 else:  
 return "Нет корней"  
  
def calculate\_probability\_of\_joint\_events(event1, event2):  
 joint\_probability = event1 + event2 - event1\*event2  
 return joint\_probability  
  
def calculate\_probability\_of\_independent\_events(event1, event2):  
 independent\_probability = event1 + event2  
 return independent\_probability

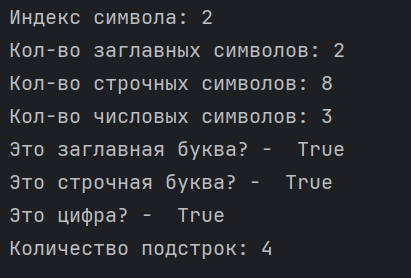


Рисунок 14 – Результат работы модуля строк

def find\_character\_in\_string(search\_char, input\_string):  
 return input\_string.find(search\_char)  
  
  
def get\_uppercase\_letters(input\_string):  
 a = 0  
 for ch in input\_string:  
 if ch.isupper():  
 a += 1  
 return a  
  
  
def get\_lowercase\_letters(input\_string):  
 a = 0  
 for ch in input\_string:  
 if ch.islower():  
 a += 1  
 return a  
  
  
def get\_numeric\_characters(input\_string):  
 a = 0  
 for ch in input\_string:  
 if ch.isdigit():  
 a += 1  
 return a  
  
  
def is\_uppercase(character):  
 return character.isupper()  
  
  
def is\_lowercase(character):  
 return character.islower()  
  
  
def is\_numeric(character):  
 return character.isdigit()  
  
  
def count\_substrings(main\_string, sub\_string):  
 return main\_string.count(sub\_string)

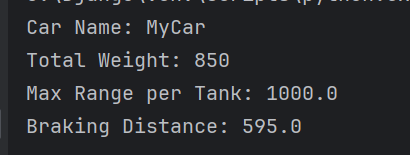


Рисунок 15 – Результат работы вывода класса

class Engine:  
 def \_\_init\_\_(self, weight, max\_speed, fuel\_consumption):  
 self.weight = weight  
 self.max\_speed = max\_speed  
 self.fuel\_consumption = fuel\_consumption  
  
class Tank:  
 def \_\_init\_\_(self, weight, tank\_volume):  
 self.weight = weight  
 self.tank\_volume = tank\_volume  
  
class Brakes:  
 def \_\_init\_\_(self, weight, brake\_efficiency):  
 self.weight = weight  
 self.brake\_efficiency = brake\_efficiency  
  
class Body:  
 def \_\_init\_\_(self, weight):  
 self.weight = weight  
  
class Car:  
 def \_\_init\_\_(self, name, engine, tank, brakes, body):  
 self.name = name  
 self.engine = engine  
 self.tank = tank  
 self.brakes = brakes  
 self.body = body  
  
 @property  
 def weight(self):  
 return self.engine.weight + self.tank.weight + self.brakes.weight + self.body.weight  
  
 @property  
 def max\_range\_per\_tank(self):  
 return (self.engine.max\_speed \* self.tank.tank\_volume) / self.engine.fuel\_consumption  
  
 @property  
 def braking\_distance(self):  
 return self.weight \* self.brakes.brake\_efficiency

ПРАКТИЧЕСКАЯ РАБОТА №6  
Тема: «Клиент-сервер»

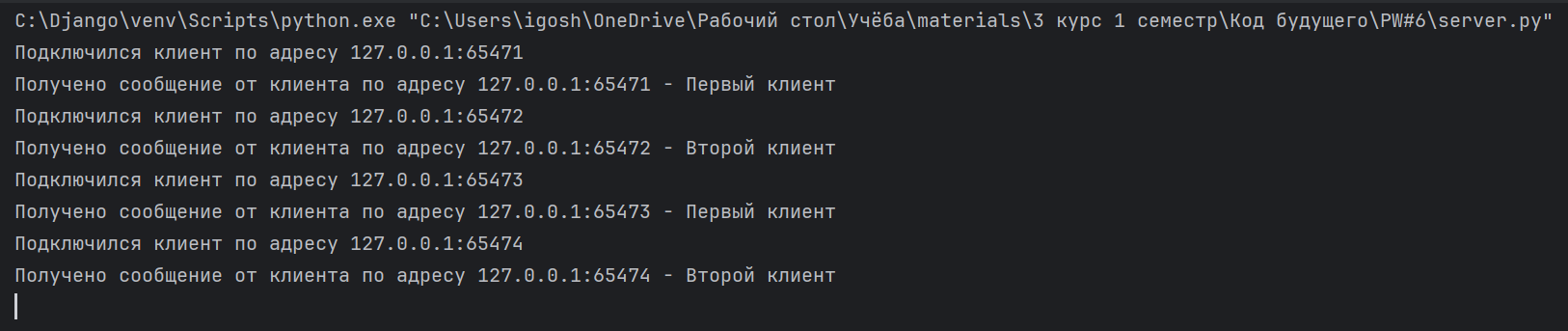


Рисунок 16 – Результат работы сервера

import socket  
import select  
  
server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
server\_socket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)  
server\_socket.bind(('localhost', 8080))  
server\_socket.listen(5)  
  
sockets\_list = [server\_socket]  
clients = {}  
while True:  
 read\_sockets, \_, exception\_sockets = select.select(sockets\_list, [], sockets\_list)  
 for notified\_socket in read\_sockets:  
 if notified\_socket == server\_socket:  
 client\_socket, client\_address = server\_socket.accept()  
 sockets\_list.append(client\_socket)  
 clients[client\_socket] = client\_address  
 print(f'Подключился клиент по адресу {client\_address[0]}:{client\_address[1]}')  
 else:  
 message = notified\_socket.recv(1024)  
 if len(message) == 0:  
 sockets\_list.remove(notified\_socket)  
 del clients[notified\_socket]  
 continue  
 client\_address = clients[notified\_socket]  
 print(f'Получено сообщение от клиента по адресу {client\_address[0]}:{client\_address[1]} - {message.decode()}')  
 for notified\_socket in exception\_sockets:  
 sockets\_list.remove(notified\_socket)  
 del clients[notified\_socket]

import socket  
client\_socket1 = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
client\_socket1.connect(('localhost', 8080))  
client\_socket1.send("Первый клиент".encode())  
client\_socket2 = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
client\_socket2.connect(('localhost', 8080))  
client\_socket2.send("Второй клиент".encode())

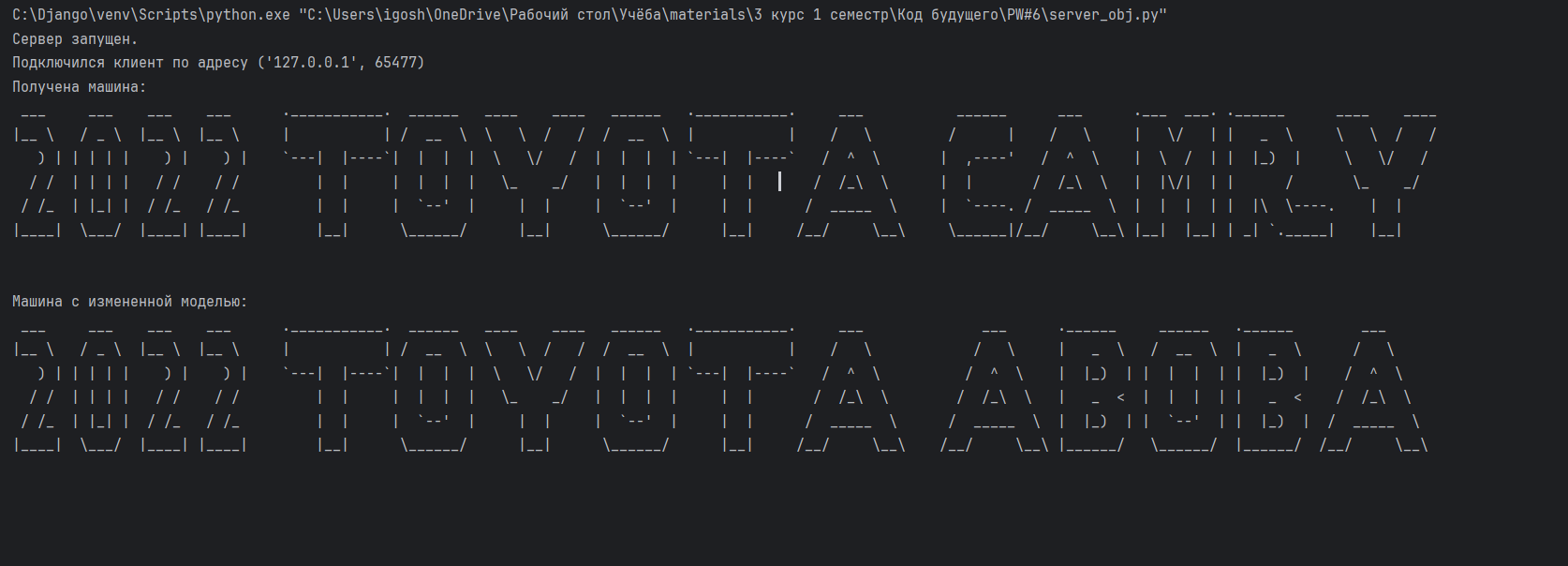


Рисунок 17 – Результат передачи объекта и изменение его полей

import socket  
import pickle  
  
server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
server\_socket.bind(('localhost', 8080))  
server\_socket.listen(1)  
print("Сервер запущен.")  
  
while True:  
 conn, addr = server\_socket.accept()  
 print(f"Подключился клиент по адресу {addr}")  
 data = conn.recv(4096)  
 car = pickle.loads(data)  
 print("Получена машина:")  
 print(car.display\_info())  
 car.change\_name("ABOBA")  
 print("Машина с измененной моделью:")  
 print(f"{car.display\_info()}")  
 conn.close()

import socket  
import pickle  
from car import Car  
car = Car("Toyota", "Camry", 2022)  
client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
client\_socket.connect(('localhost', 8080))  
data = pickle.dumps(car)  
client\_socket.sendall(data)  
client\_socket.close()

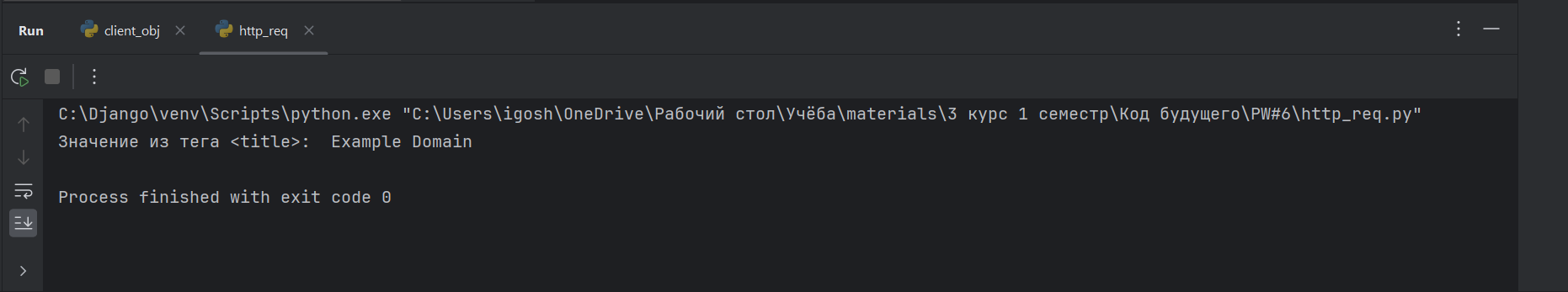


Рисунок 18 – Получение значения из титульника по HTTP-запросу

from urllib import request  
from bs4 import BeautifulSoup  
response = request.urlopen("http://example.org")  
html = response.read()  
soup = BeautifulSoup(html, "html.parser")  
title\_tag = soup.title  
if title\_tag is not None:  
 title\_text = title\_tag.string  
 print("Значение из тега <title>: ", title\_text)  
else:  
 print("Тег <title> не найден на странице")

ПРАКТИЧЕСКАЯ РАБОТА №7

Тема: «Запросы»

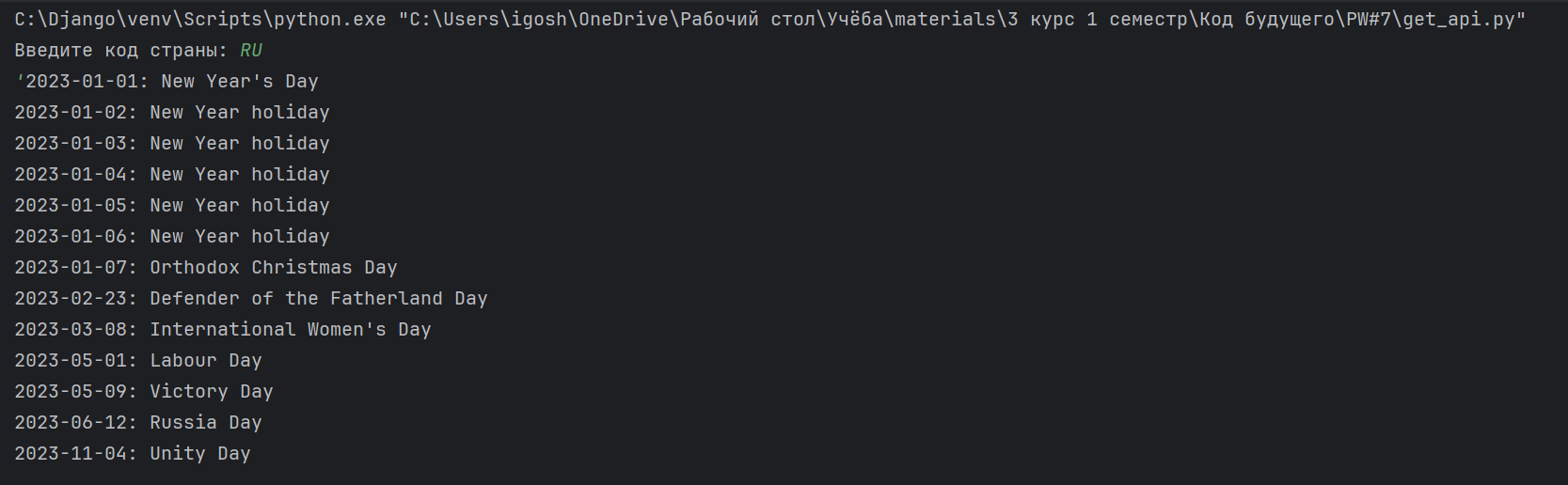


Рисунок 19 – Получение списка праздничных дней по коду

import json  
from urllib import request  
  
def get\_holidays(country\_code):  
 url = f"https://date.nager.at/api/v3/PublicHolidays/2023/{country\_code}"  
 with request.urlopen(url) as response:  
 data = response.read()  
 holidays\_data = json.loads(data)  
 return holidays\_data  
  
  
country\_code = input("Введите код страны: ")  
holidays = get\_holidays(country\_code)  
if isinstance(holidays, list):  
 for holiday in holidays:  
 print(f"{holiday['date']}: {holiday['name']}")  
else:  
 print(holidays)

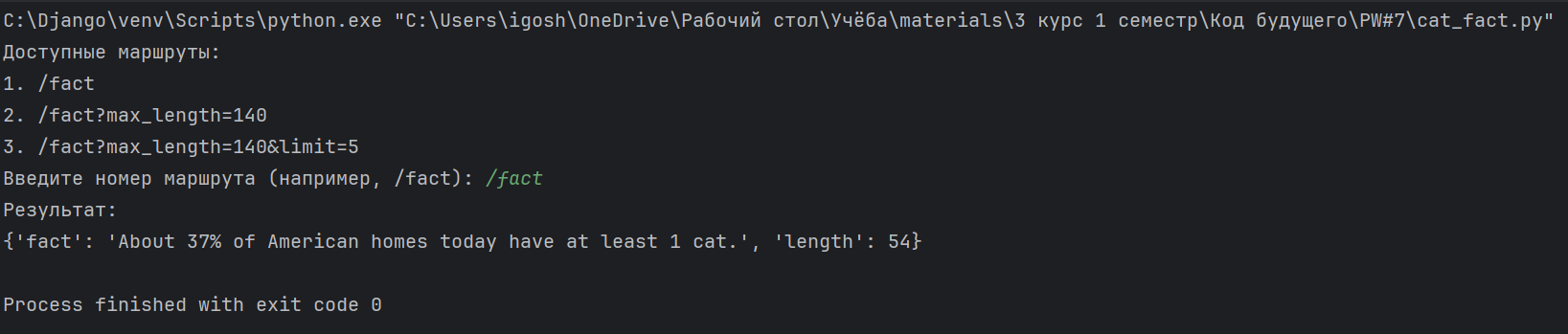


Рисунок 20 – Получение факта о котах из сайта фактов о котах

from urllib import request  
import json  
  
#У меня не работали requests, поэтому все обращения я сделал через библиотеку urllib  
def get\_cat\_fact(route):  
 url = f"https://catfact.ninja{route}"  
 with request.urlopen(url) as response:  
 data = response.read()  
 cat\_data = json.loads(data)  
 return cat\_data  
  
def main():  
 print("Доступные маршруты:\n1. /fact\n2. /fact?max\_length=140\n3. /fact?max\_length=140&limit=5")  
 choice = input("Введите номер маршрута (например, /fact): ")  
 cat\_data = get\_cat\_fact(choice)  
 print("Результат:")  
 print(cat\_data)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

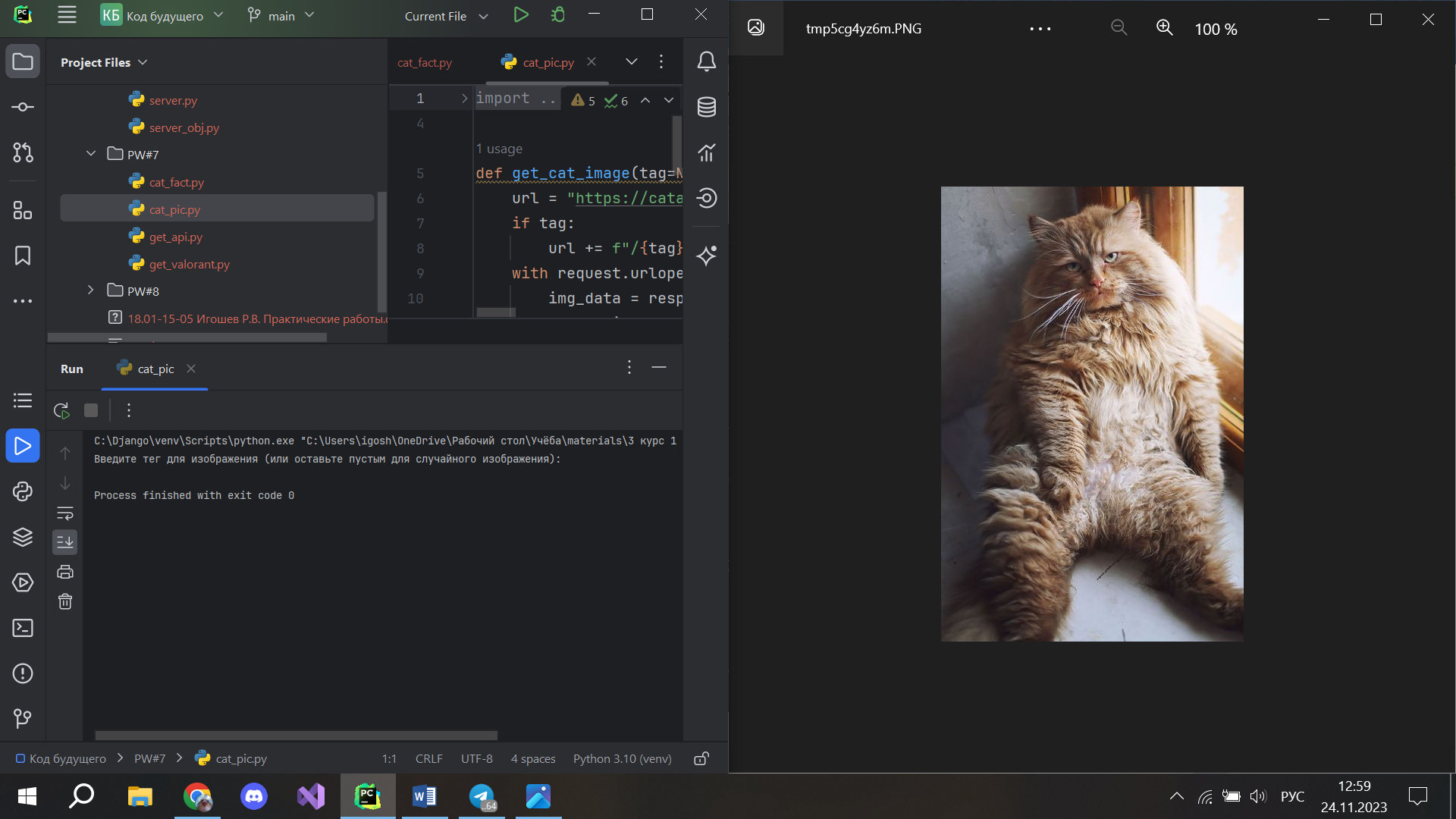
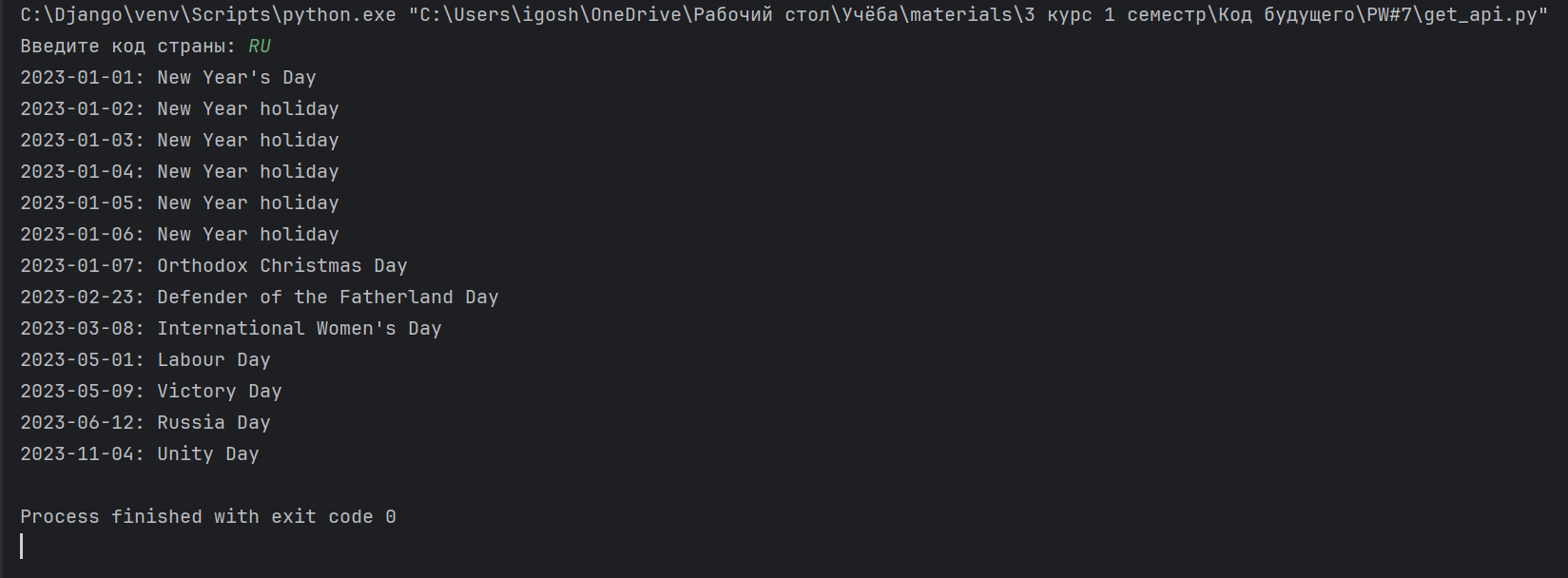


Рисунок 21 – Результат работы получения фотки котика

from urllib import request  
from PIL import Image  
from io import BytesIO  
  
def get\_cat\_image(tag=None):  
 url = "https://cataas.com/cat"  
 if tag:  
 url += f"/{tag}"  
 with request.urlopen(url) as response:  
 img\_data = response.read()  
 return img\_data  
  
def display\_image(img\_data):  
 img = Image.open(BytesIO(img\_data))  
 img.show()  
  
def main():  
 tag = input("Введите тег для изображения (или оставьте пустым для случайного изображения): ")  
 cat\_image\_data = get\_cat\_image(tag)  
 display\_image(cat\_image\_data)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()



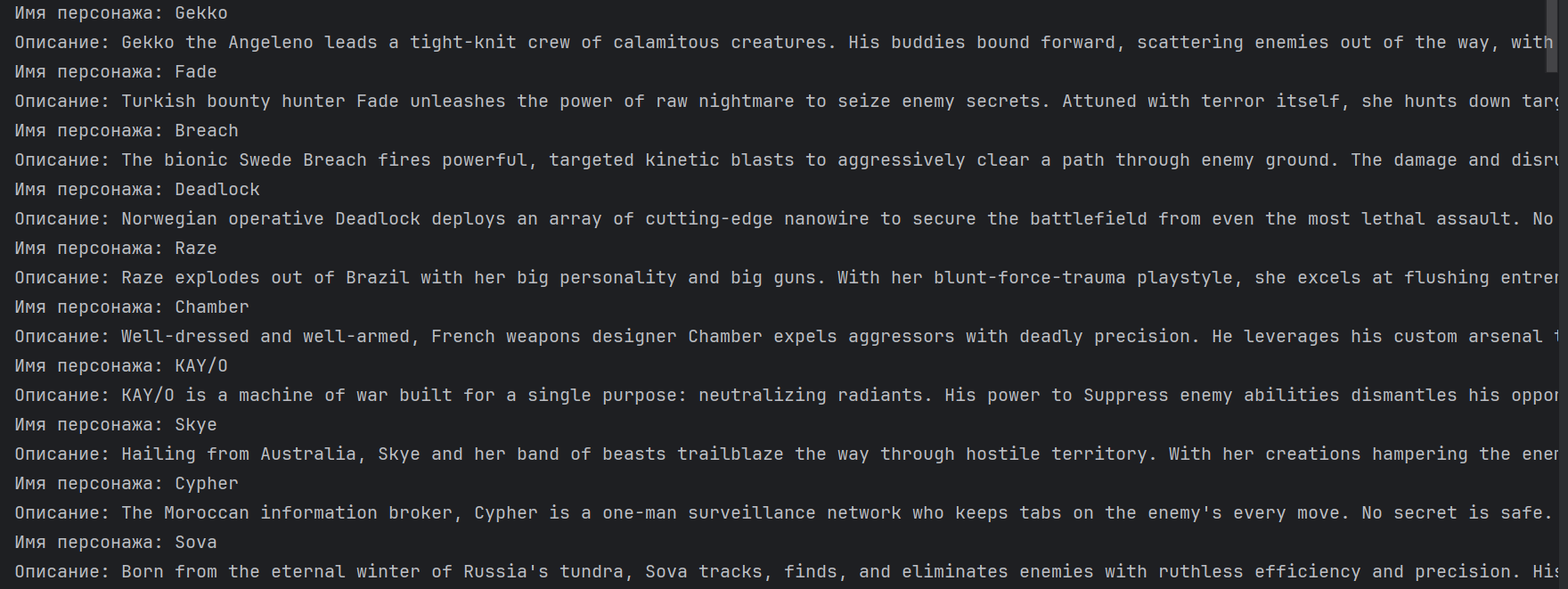


Рисунок 22 – Результат работы обращения к апи валорант персонажей

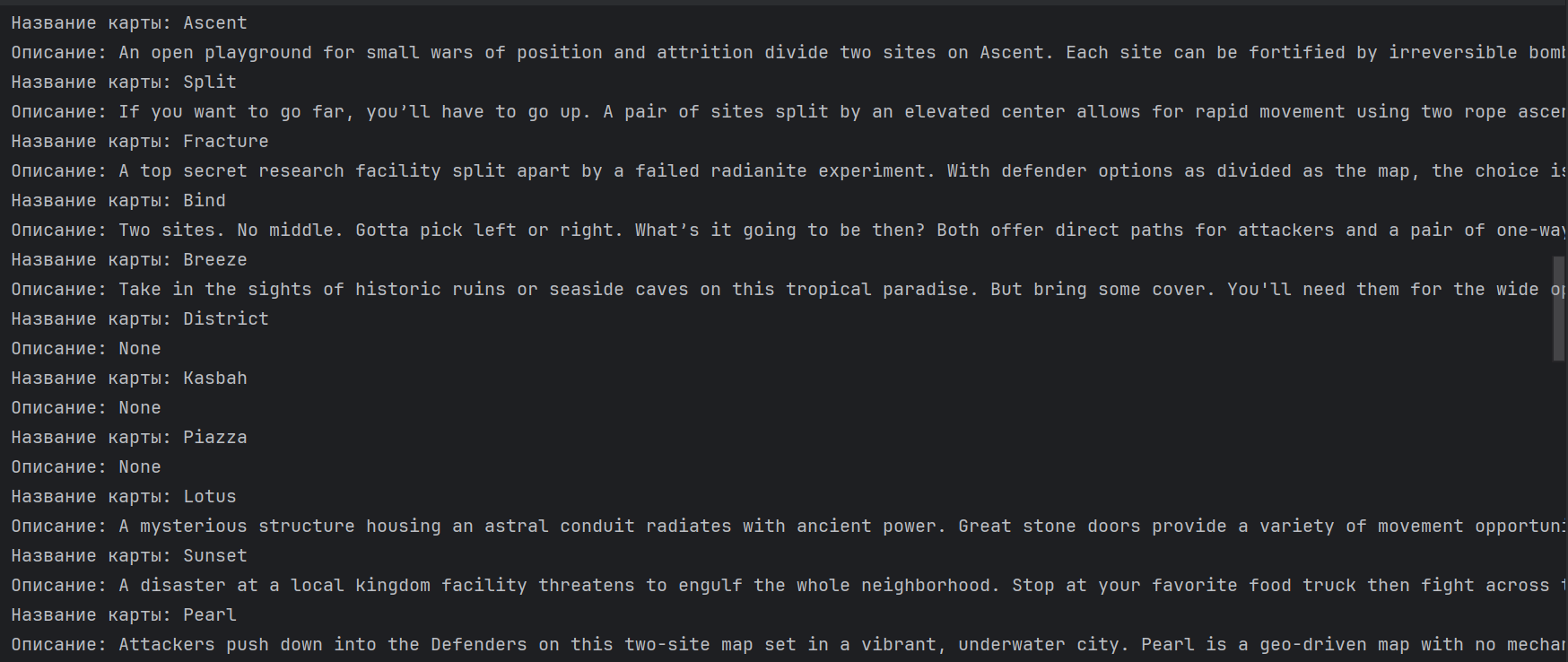


Рисунок 23 - Результат работы обращения к апи валорант карт

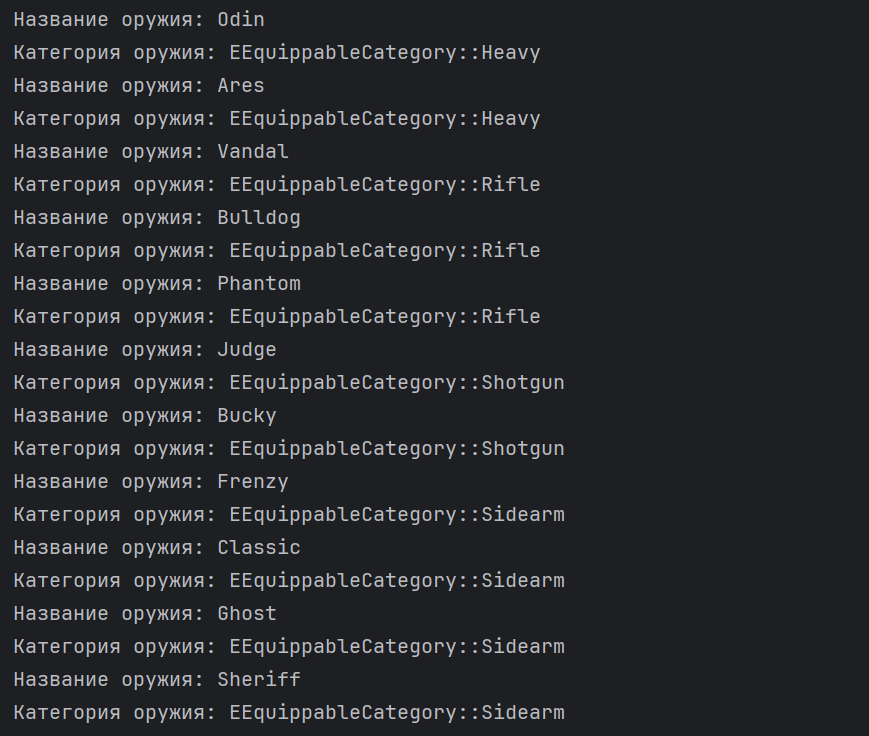


Рисунок 24 - Результат работы обращения к апи валорант Оружий

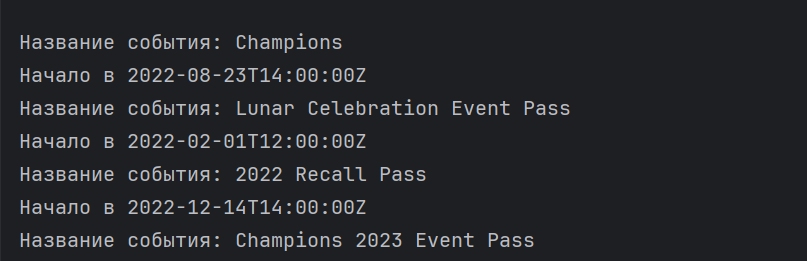


Рисунок 25 - Результат работы обращения к апи валорант событий

import urllib.request  
import json  
  
  
def print\_agent\_info():  
 url = "https://valorant-api.com/v1/agents"  
 response = urllib.request.urlopen(url)  
 data = json.load(response)  
 agents = data["data"]  
 for agent in agents:  
 print(f"Имя персонажа: {agent['displayName']}")  
 print(f"Описание: {agent['description']}")  
  
  
def print\_map\_info():  
 url = "https://valorant-api.com/v1/maps"  
 response = urllib.request.urlopen(url)  
 data = json.load(response)  
 maps = data["data"]  
 for game\_map in maps:  
 print(f"Название карты: {game\_map['displayName']}")  
 print(f"Описание: {game\_map['narrativeDescription']}")  
  
  
def print\_weapon\_info():  
 url = "https://valorant-api.com/v1/weapons"  
 response = urllib.request.urlopen(url)  
 data = json.load(response)  
 weapons = data["data"]  
 for weapon in weapons:  
 print(f"Название оружия: {weapon['displayName']}")  
 print(f"Категория оружия: {weapon['category']}")  
  
  
def print\_event\_info():  
 url = "https://valorant-api.com/v1/events"  
 response = urllib.request.urlopen(url)  
 data = json.load(response)  
 events = data["data"]  
 for event in events:  
 print(f"Название события: {event['displayName']}")  
 print(f"Начало в {event['startTime']}")  
  
  
print\_agent\_info()  
print("\n\n\n\n")  
print\_map\_info()  
print("\n\n\n\n")  
print\_weapon\_info()  
print("\n\n\n\n")  
print\_event\_info()

ПРАКТИЧЕСКАЯ РАБОТА №8

Тема: «Собственный АПИ»

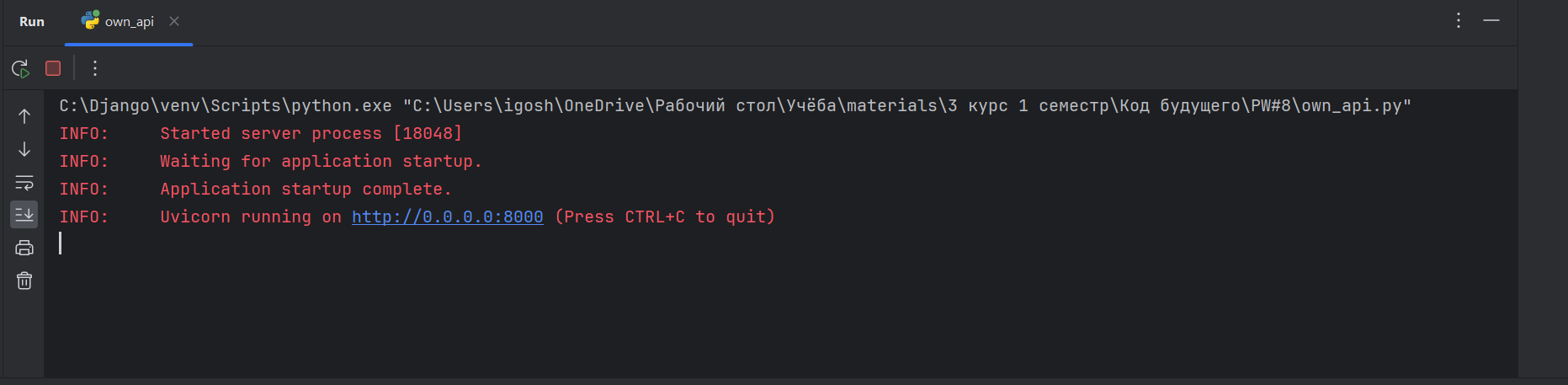


Рисунок 26 – Результат запуска АПИ-сервера

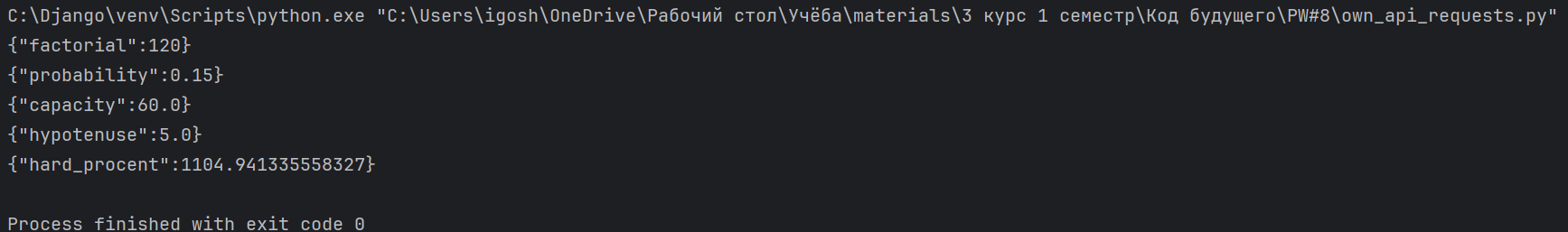


Рисунок 27 – Результат обращения к АПИ-серверу

from fastapi import FastAPI, Response  
from pydantic import BaseModel  
  
app = FastAPI()  
  
class FactorialRequest(BaseModel):  
 number: int  
  
class ProbabilityRequest(BaseModel):  
 event1\_probability: float  
 event2\_probability: float  
  
class ParallelepipedVolumeRequest(BaseModel):  
 length: float  
 width: float  
 height: float  
  
class TriangleCalculationRequest(BaseModel):  
 side\_a: float = None  
 side\_b: float = None  
 hypotenuse: float = None  
  
class CompoundInterestRequest(BaseModel):  
 start\_summ: float  
 year\_procent: float  
 number\_of\_payments: int  
 years: int  
  
@app.post("/factorial/")  
async def factorial\_calculation(request: FactorialRequest):  
 result = 1  
 for i in range(1, request.number + 1):  
 result \*= i  
 return {"factorial": result}  
  
@app.post("/probability/")  
async def probability\_calculation(request: ProbabilityRequest):  
 result = request.event1\_probability \* request.event2\_probability  
 return {"probability": result}  
  
@app.post("/parallelepiped\_capacity/")  
async def parallelepiped\_capacity\_calculation(request: ParallelepipedVolumeRequest):  
 return {"capacity": request.length \* request.width \* request.height}  
  
@app.post("/triangle\_calculation/")  
async def triangle\_calculation(request: TriangleCalculationRequest):  
 if request.side\_a is not None and request.side\_b is not None:  
 result = (request.side\_a \*\* 2 + request.side\_b \*\* 2) \*\* 0.5  
 return {"hypotenuse": result}  
 elif request.side\_a is not None and request.hypotenuse is not None:  
 result = (request.hypotenuse \*\* 2 - request.side\_a \*\* 2) \*\* 0.5  
 return {"side\_b": result}  
 elif request.side\_b is not None and request.hypotenuse is not None:  
 result = (request.hypotenuse \*\* 2 - request.side\_b \*\* 2) \*\* 0.5  
 return {"side\_a": result}  
 else:  
 return {"error": "Отправлены неверные данные"}  
  
  
@app.post("/hard\_procent/")  
async def hard\_procent\_calc(request: CompoundInterestRequest):  
 result = request.start\_summ \* ((1 + request.year\_procent / request.number\_of\_payments) \*\* (request.number\_of\_payments \* request.years))  
 return {"hard\_procent": result}  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import uvicorn  
 uvicorn.run(app, host="0.0.0.0", port=8000)

import json  
import urllib.request  
import urllib.parse  
  
factorial\_request = {"number": 5}  
params = json.dumps(factorial\_request).encode('utf-8')  
req = urllib.request.Request("http://127.0.0.1:8000/factorial/", data=params, headers={'content-type': 'application/json'})  
response = urllib.request.urlopen(req)  
print(response.read().decode('utf-8'))  
  
probability\_request = {"event1\_probability": 0.3, "event2\_probability": 0.5}  
params = json.dumps(probability\_request).encode('utf-8')  
req = urllib.request.Request("http://127.0.0.1:8000/probability/", data=params, headers={'content-type': 'application/json'})  
response = urllib.request.urlopen(req)  
print(response.read().decode('utf-8'))  
  
volume\_request = {"length": 4.0, "width": 3.0, "height": 5.0}  
params = json.dumps(volume\_request).encode('utf-8')  
req = urllib.request.Request("http://127.0.0.1:8000/parallelepiped\_capacity/", data=params, headers={'content-type': 'application/json'})  
response = urllib.request.urlopen(req)  
print(response.read().decode('utf-8'))  
  
triangle\_request = {"side\_a": 3, "side\_b": 4}  
params = json.dumps(triangle\_request).encode('utf-8')  
req = urllib.request.Request("http://127.0.0.1:8000/triangle\_calculation/", data=params, headers={'content-type': 'application/json'})  
response = urllib.request.urlopen(req)  
print(response.read().decode('utf-8'))  
  
interest\_request = {"start\_summ": 1000, "year\_procent": 0.05, "number\_of\_payments": 12, "years" : 2}  
params = json.dumps(interest\_request).encode('utf-8')  
req = urllib.request.Request("http://127.0.0.1:8000/hard\_procent/", data=params, headers={'content-type': 'application/json'})  
response = urllib.request.urlopen(req)  
print(response.read().decode('utf-8'))