Лабораторная работа №4

Листинг программы представлен на ниже.

```
Import pyautogui
import time
import webbrowser
import requests
from bs4 import BeautifulSoup
import cv2
import numpy as np
import os
import subprocess
import sys

# Установка необходимых библиотек
try:
    import PIL
except ImportError:
    print("Устанавливаем Pillow…")
    subprocess.check_call([sys.executable, "-m", "pip", "install", "pillow"])
    import PIL

try:
    import pyscreeze
except ImportError:
    print("Устанавливаем pyscreeze…")
    subprocess.check_call([sys.executable, "-m", "pip", "install",
"pyscreeze"])
    import pyscreeze

# Открываем сайт в браузере
webbrowser.open(
    'https://rasp.unn.ru/index.php?view=0&date=2025-09-
17&fac=281474976710668&gr=281474976728062&type=2&typemenu=1')

# Ждем загрузки страницы
time.sleep(5)

# Координаты для кликов
coordinates = [
    (246, 157),
    (212, 456),
    (796, 153),
    (705, 346),
    (1180, 153),
    (1043, 380),
    (194, 665)
]

# Выполняем клики по координатам
for x, y in coordinates:
    pyautogui.click(x, y)
    time.sleep(1)  # Пауза между кликами

# Ждем немного после кликов для обновления страницы
time.sleep(3)

try:
    # # Копируем HTML со страницы через буфер обмена
    # pyautogui.hotkey('ctrl', 'u')  # Открываем исходный код страницы
    # time.sleep(2)
    # pyautogui.hotkey('ctrl', 'a')  # Выделяем всё
    # time.sleep(1)
```

```
    # pyautogui.hotkey('ctrl', 'c')  # Копируем
    # time.sleep(1)


    # Альтернативный подход: делаем скриншот и ищем таблицу визуально
    print("Ищем таблицу с классом 'table table-hover'…")

    # Попробуем получить HTML через другой URL или метод
    api_url = 'https://rasp.unn.ru/ajax/schedule.php'  # Возможный API
endpoint
    try:
        response = requests.get(api_url)
        html_content = response.text
    except:
        # Если не получается через API, используем альтернативный метод
        print("Не удалось получить данные через API, используем
альтернативный подход…")
        html_content = ""

    # Парсим HTML с помощью BeautifulSoup
    soup = BeautifulSoup(html_content, 'html.parser') if html_content else
BeautifulSoup()

    # ИЩЕМ КОНКРЕТНО ТАБЛИЦУ С КЛАССОМ table table-hover
    table = soup.find('table', class_='table table-hover')

    if table:
        print("✓ Таблица найдена!")
        print("Содержимое таблицы:")
        print(table.prettify())

        # Извлекаем все строки таблицы
        rows = table.find_all('tr')

        teacher_name = None
        location = None
        subject_name = None

        for row in rows:
            cells = row.find_all('td')
            if len(cells) >= 2:
                header = cells[0].get_text(strip=True)
                value = cells[1].get_text(strip=True)

                print(f"Заголовок: '{header}', Значение: '{value}'")

                if 'Преподаватель:' in header:
                    teacher_name = value
                    print(f"Найден преподаватель: {teacher_name}")

                if 'Место проведения:' in header:
                    location = value
                    print(f"Найдено место проведения: {location}")

                if 'Название предмета:' in header:
                    subject_name = value
                    print(f"Найден предмет: {subject_name}")

        # Проверяем наличие нужного преподавателя и места проведения
        if teacher_name == "Гергель Александр Викторович":
            print("✓ Преподаватель 'Гергель Александр Викторович' найден!")
        else:
            print(f»✗ Преподаватель 'Гергель Александр Викторович' не найден.
```

```
Найден: {teacher_name}»)

        if location == «Виртуальное, Дистанционная»:
            print(«✓ Место проведения 'Виртуальное, Дистанционная' найдено!»)
        else:
            print(f»✗ Место проведения 'Виртуальное, Дистанционная' не
найдено. Найдено: {location}»)

        # Проверяем оба условия
        if teacher_name == «Гергель Александр Викторович» and location ==
«Виртуальное, Дистанционная»:
            print(«✓ УСПЕХ: Оба условия выполнены - нужный преподаватель
ведет в нужном месте!»)
        else:
            print(«✗ Условия не выполнены полностью»)

    else:
        print(«✗ Таблица с классом 'table table-hover' не найдена в HTML»)
        print(«Попробуем найти другие таблицы…»)

        # Ищем любые таблицы на странице
        all_tables = soup.find_all('table')
        print(f»Найдено таблиц: {len(all_tables)}»)

        for i, tbl in enumerate(all_tables):
            print(f»Таблица {i + 1}:»)
            print(tbl.prettify()[:500] + «…» if len(tbl.prettify()) > 500
else tbl.prettify())
            print(«-« * 50)

    print(«Все клики выполнены успешно!»)

except Exception as e:
    print(f»Произошла ошибка: {e}»)
    print(«Продолжаем выполнение без проверки…»)


# Функция для создания скриншота с использованием mss (альтернатива)
def take_screenshot_mss(region=None):
    try:
        import mss
        with mss.mss() as sct:
            if region:
                monitor = {«top»: region[1], «left»: region[0], «width»:
region[2], «height»: region[3]}
            else:
                monitor = sct.monitors[1]
            screenshot = sct.grab(monitor)
            return np.array(screenshot)
    except ImportError:
        print(«Устанавливаем mss…»)
        subprocess.check_call([sys.executable, «-m», «pip», «install»,
«mss»])
        import mss
        with mss.mss() as sct:
            if region:
                monitor = {«top»: region[1], «left»: region[0], «width»:
region[2], «height»: region[3]}
            else:
                monitor = sct.monitors[1]
            screenshot = sct.grab(monitor)
            return np.array(screenshot)
```

```
# Скриншот указанной области и сравнение с prepod.PNG
try:
    print(«\n» + «=» * 50)
    print(«СНИМАЕМ ЭКРАН И СРАВНИВАЕМ С prepod.PNG»)
    print(«=» * 50)
    # Координаты области для скриншота
    x, y, width, height = 870, 555, 1078 - 870, 587 - 555
    # Делаем скриншот указанной области с помощью mss
    screenshot_array = take_screenshot_mss(region=(x, y, width, height))
    # Конвертируем в PIL Image для сохранения
    from PIL import Image
    screenshot_pil = Image.fromarray(screenshot_array)
    screenshot_pil.save('comparison_screenshot.png')
    print(f»Скриншот области ({x}, {y}, {width}, {height}) сохранен как
'comparison_screenshot.png'»)
    # Проверяем наличие файла prepod.PNG
    if not os.path.exists('prepod.PNG'):
        print(«✗ Файл prepod.PNG не найден в текущей директории»)
        print(«Убедитесь, что файл prepod.PNG находится в той же папке, что и
скрипт»)
    else:
        # Загружаем изображения для сравнения
        screenshot_img = cv2.cvtColor(screenshot_array, cv2.COLOR_BGRA2BGR)
        prepod_img = cv2.imread('prepod.PNG')
        # Проверяем, что изображения загружены корректно
        if screenshot_img is None:
            print(«✗ Не удалось загрузить скриншот для сравнения»)
        elif prepod_img is None:
            print(«✗ Не удалось загрузить prepod.PNG»)
        else:
            # Приводим изображения к одинаковому размеру (на случай если
размеры отличаются)
            prepod_img_resized = cv2.resize(prepod_img,
(screenshot_img.shape[1], screenshot_img.shape[0]))

            # Сравниваем изображения с помощью метода template matching
            result = cv2.matchTemplate(screenshot_img, prepod_img_resized,
cv2.TM_CCOEFF_NORMED)
            min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)

            print(f»Уверенность в совпадении: {max_val:.4f}»)

            # Порог уверенности (можно настроить)
            confidence_threshold = 0.8
            if max_val >= confidence_threshold:
                print(«✓ Изображения СОВПАДАЮТ с высокой уверенностью!»)
                print(f»Уровень совпадения: {max_val * 100:.2f}%»)
            else:
                print(«✗ Изображения НЕ СОВПАДАЮТ или совпадение недостаточно
уверенное»)
                print(f»Требуется минимум {confidence_threshold * 100}%,
получено: {max_val * 100:.2f}%»)
            # Сохраняем визуализацию сравнения
            h, w = prepod_img_resized.shape[:2]
            top_left = max_loc
            bottom_right = (top_left[0] + w, top_left[1] + h)
            # Рисуем прямоугольник вокруг найденной области
            result_img = screenshot_img.copy()
            cv2.rectangle(result_img, top_left, bottom_right, (0, 255, 0), 2)
            cv2.putText(result_img, f'Confidence: {max_val:.4f}', (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
```

```
            cv2.imwrite('comparison_result.png', result_img)
            print(«Результат сравнения сохранен как 'comparison_result.png'»)
except Exception as e:
    print(f»Ошибка при сравнении изображений: {e}»)
    import traceback
    traceback.print_exc()
print(«\nПрограмма завершена!»)
```

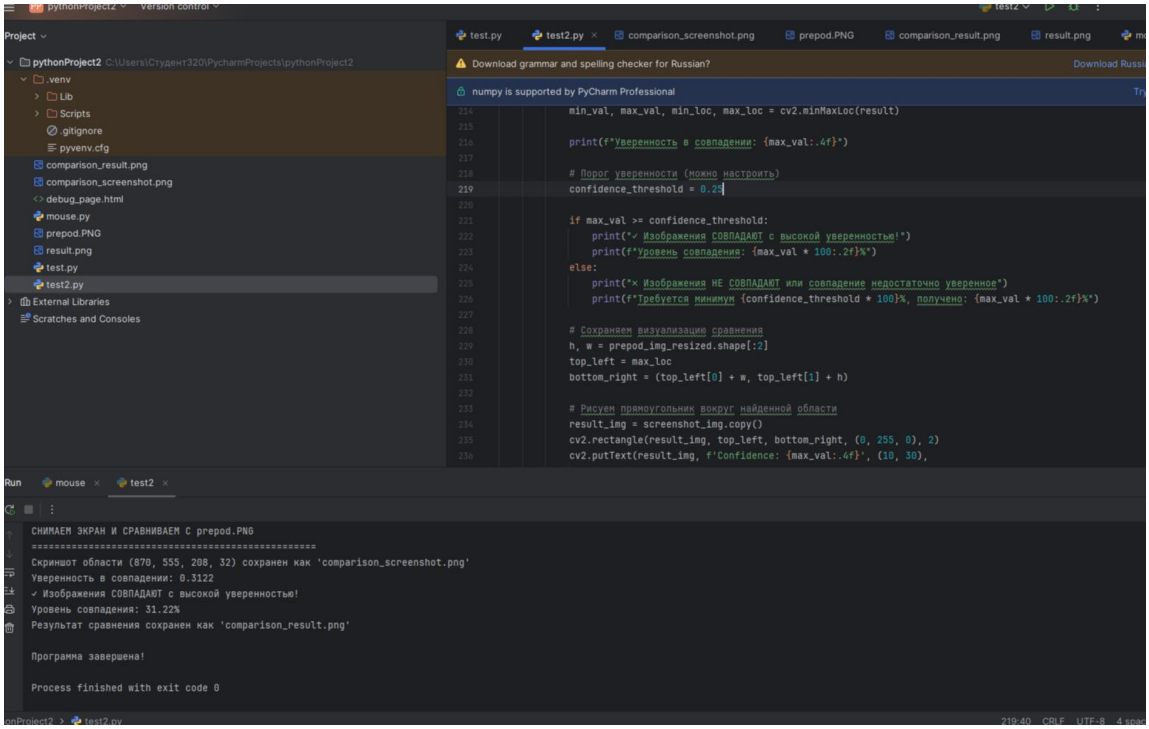

Рисунок 1 – Результат теста

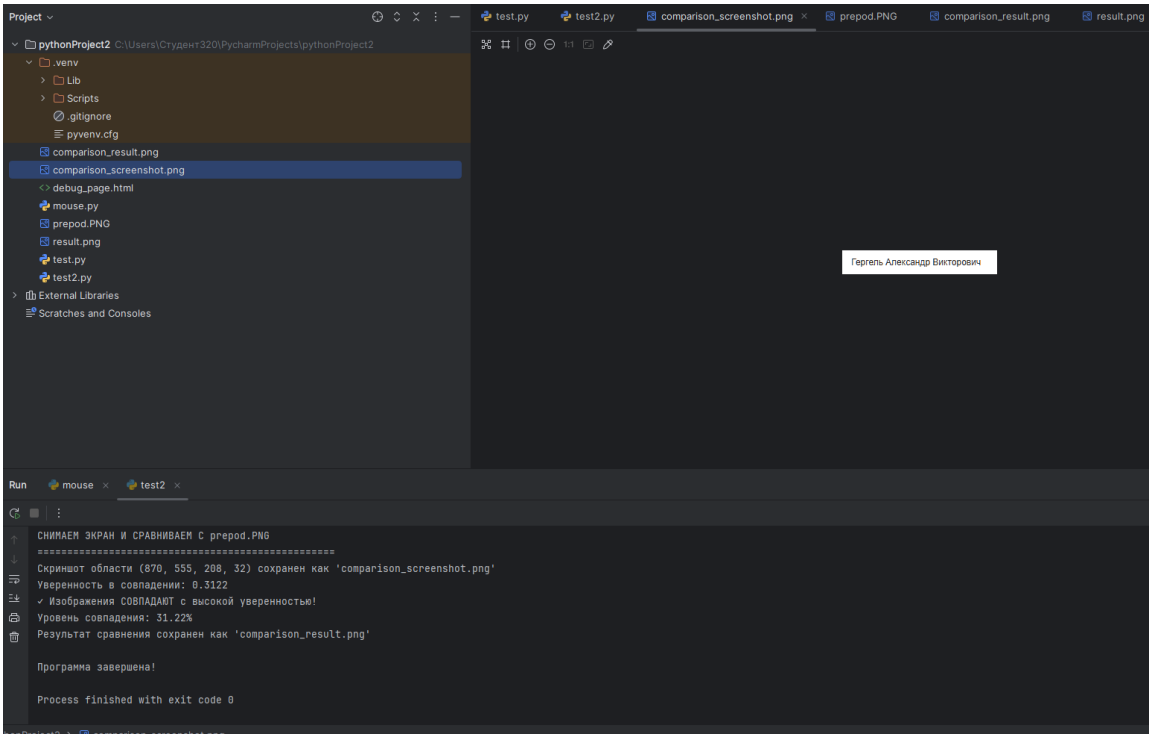

Рисунок 2 – Результат парсинга

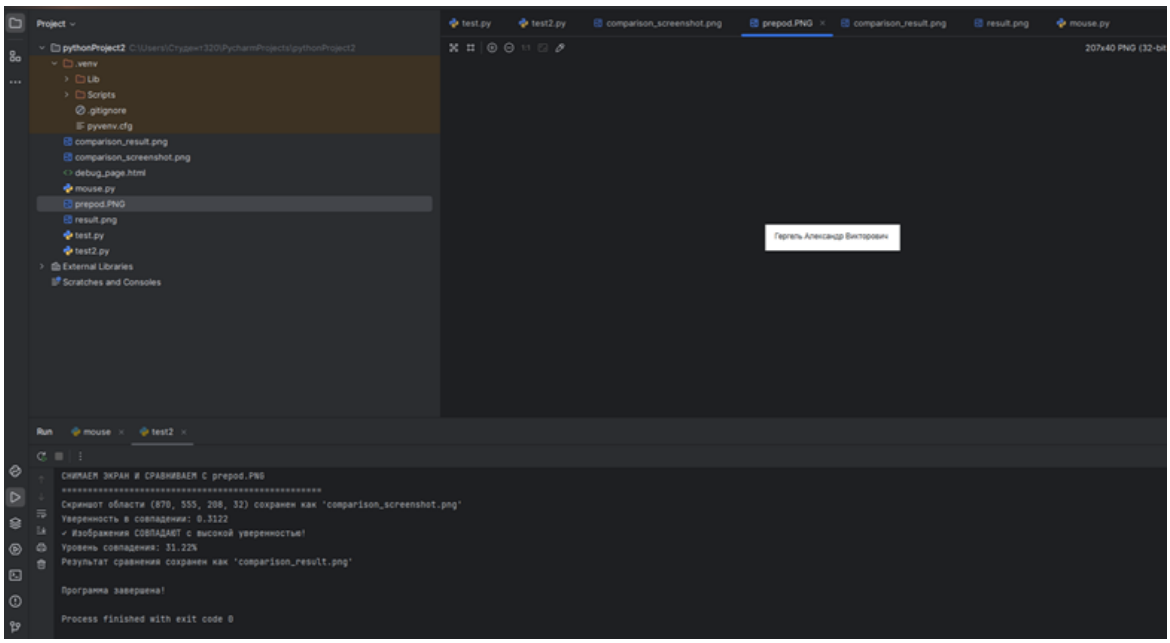

Рисунок 3 – Рисунок для проверки тестом

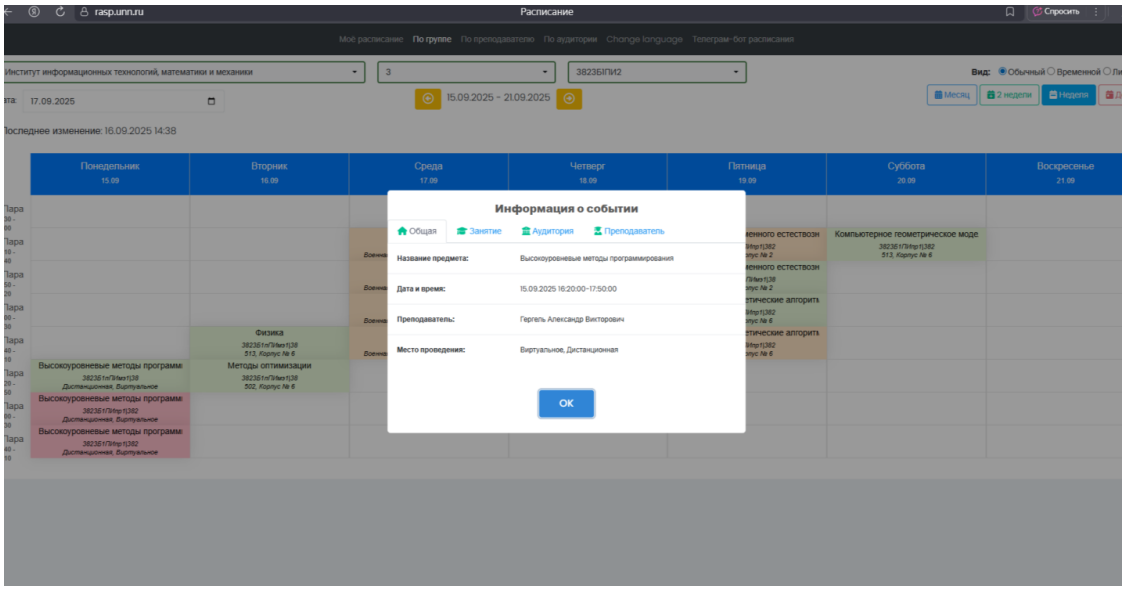

Рисунок 4 – Сайт и окно проверки