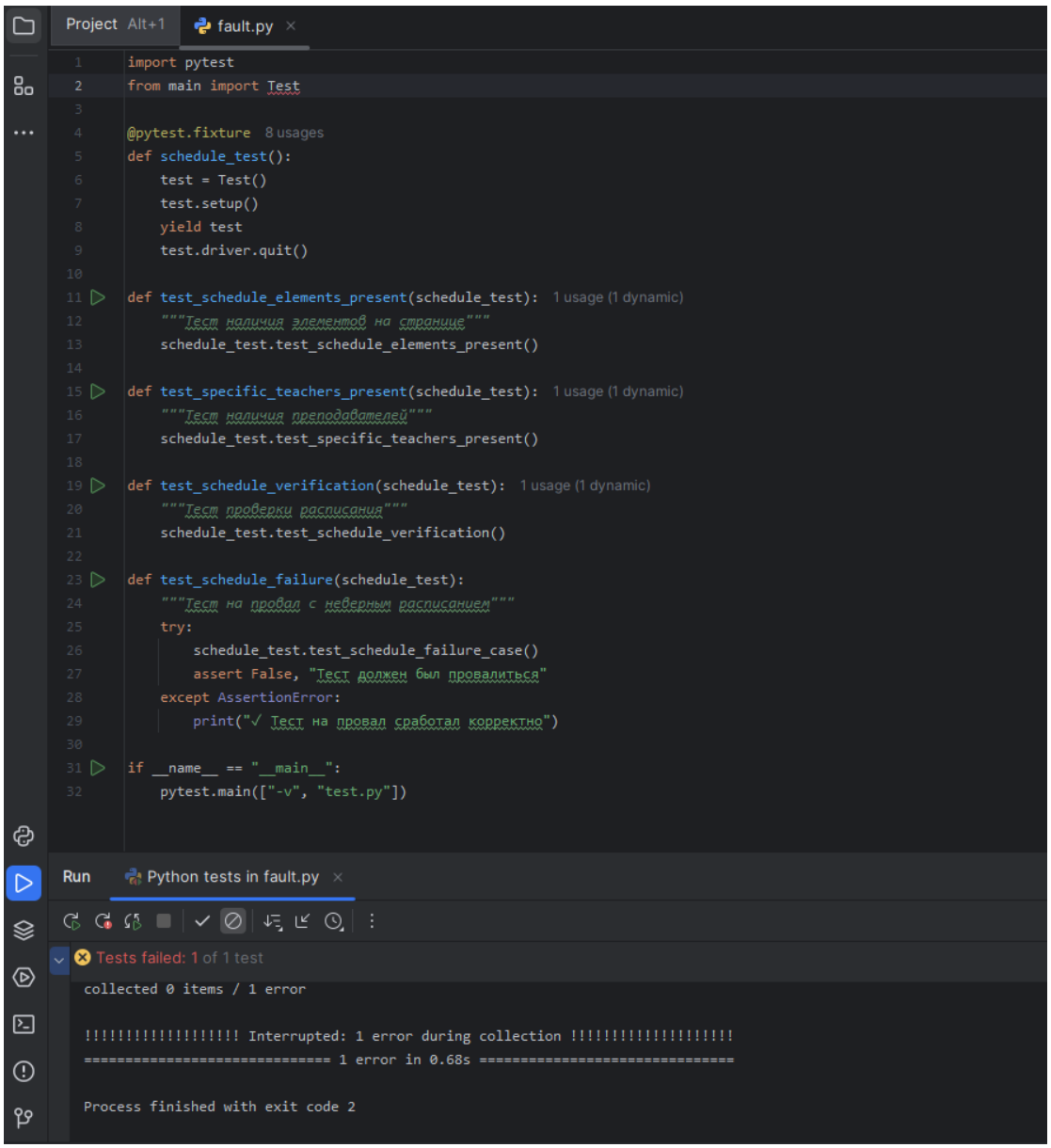
Практическое задание № 4

Задание:

Получаем ваше расписание на сайте https://api.nntu.ru/ и сверяем тестами pytest с заранее выбранным. Показываем что если заменяем свое расписание то тесты не проходят. Selenium — это библиотека для автоматизации веб-браузеров. Она позволяет управлять браузерами программно, что полезно для тестирования вебприложений, скрапинга данных или выполнения рутинных задач в браузере. В Python Selenium работает через библиотеку selenium, которая предоставляет API для взаимодействия с браузерами.

Решение:

Сначала покажем программу с ошибкой:



Теперь запустим программу без ошибки. Логика ее заключается в том, что мы запускаем программу и нас кидает на страницу расписаний. После чего программа с изначально выбранными параметрами нашей группы и формой обучения(Заочная 3 года и 10 мес; АЗИС 22-2) заполняет их на сайте и на выходе мы получаем все наше расписание со страницы.

```
Браузер запущен
Загружаем страницу расписания с предустановленными значениями...
Страница расписания открыта
Устанавливаем предварительные значения...
Предварительные значения установлены

=== ПРОВЕРКА ВЫБРАННЫХ ЗНАЧЕНИЙ ===
Выбранная форма обучения: Заочная 3 года 10 мес (значение: 3)
Выбранная группа: АЗИС 22-2 (значение: 804)
✓ Все значения выбраны корректно

Загружаем расписание...
Ошибка при нажатии кнопки: Message:

Кнопка нажата через JavaScript

================================================================================
РАСПИСАНИЕ ГРУППЫ АЗИС 22-2
================================================================================

ЧЕТВЕРГ, 6 НОЯБРЯ 2025
--------------------------------------------------------------------------------
  1 пара / 08:30-10:00 Инфокоммуникационные системы и сети / КР. Гуськова Юлия Александровна 220 прием КР Четная
  2 пара / 10:10-11:40 Инфокоммуникационные системы и сети / Экз. Гуськова Юлия Александровна 220 Четная
  3 пара / 12:10-13:40 Основы тестирования программного обеспечения / Экз. Комаров Александр Олегович 324 Четная
--------------------------------------------------------------------------------

ПЯТНИЦА, 7 НОЯБРЯ 2025
--------------------------------------------------------------------------------
  3 пара / 12:10-13:40 производственная (преддипломная практика) заочка / Жидкова Наталья Валерьевна 218 Четная
  4 пара / 13:50-15:20 Организационно-экономическое обоснование научных и технических решений / Зач. Гусева Ирина Борисовна 218 Четная
  5 пара / 15:30-17:00 Эксплуатация и модификация информационных систем / Экз. Жидкова Наталья Валерьевна 226 Четная
--------------------------------------------------------------------------------

ПОНЕДЕЛЬНИК, 10 НОЯБРЯ 2025
--------------------------------------------------------------------------------
  4 пара / 13:50-15:20 Анализ больших данных / Зач.Оц. Рябов Антон Владимирович 5 Четная
  5 пара / 15:30-17:00 Надежность и отказоустойчивость информационных систем / РГР Жидкова Наталья Валерьевна 226 прием РГР Четная
  1 пара вечер / 17:30-19:00 Надежность и отказоустойчивость информационных систем / Зач.Оц. Жидкова Наталья Валерьевна 226 Четная
================================================================================
Всего дней: 3
Всего пар: 9
================================================================================
```

Вывод расписания после выполнения программы

Код программы:

```
import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.action_chains import ActionChains
```

```
class UniversityScheduleTester:
    def __init__(self):
        self.setup_browser()

    def setup_browser(self):
        """Настройка браузера"""
        chrome_options = Options()
        chrome_options.add_argument("--start-maximized")
        chrome_options.add_argument("--disable-dev-shm-usage")
        chrome_options.add_argument("--disable-gpu")
        chrome_options.add_argument("--no-sandbox")
        chrome_options.add_argument("--disable-blink-
features=AutomationControlled")
        chrome_options.add_experimental_option("excludeSwitches", ["enable-
automation"])
        chrome_options.add_experimental_option('useAutomationExtension', False)

        self.service = Service(ChromeDriverManager().install())
        self.driver = webdriver.Chrome(service=self.service,
options=chrome_options)
        self.driver.execute_script("Object.defineProperty(navigator, 'webdriver', {get:
() => undefined})")
        self.wait = WebDriverWait(self.driver, 20)
        self.actions = ActionChains(self.driver)
        print("Браузер запущен")

    def close_browser(self):
        """Закрытие браузера"""
        if hasattr(self, 'driver') and self.driver:
            self.driver.quit()
            print("Браузер закрыт")

    def open_schedule_page_with_preselection(self):
        """Открытие страницы расписания с предустановленными
параметрами"""
        # Открываем страницу с параметрами в URL для предварительного
выбора
        url = 'https://api.nntu.ru/raspisanie'
        self.driver.get(url)
        print("Страница расписания открыта")
        time.sleep(3)

        # Ждем загрузки страницы и устанавливаем значения через JavaScript
```

```
        self.set_preselected_values()

    def set_preselected_values(self):
        """Установка предварительно выбранных значений через JavaScript"""
        try:
            print("Устанавливаем предварительные значения...")

            # Устанавливаем значение для формы обучения (Заочная 3 года 10 мес)
            department_script = """
                var departmentSelect = document.getElementById('studentAdvert__controls--department');
                if (departmentSelect) {
                    // Ищем опцию с заочной формой обучения
                    for (var i = 0; i < departmentSelect.options.length; i++) {
                        var option = departmentSelect.options[i];
                        if (option.text.includes('Заочная 3 года 10 мес') ||
                            option.text.includes('Заочная') && option.text.includes('3 года') && option.text.includes('10 мес')) {
                            departmentSelect.value = option.value;
                            var event = new Event('change', { bubbles: true });
                            departmentSelect.dispatchEvent(event);
                            console.log('Установлена форма обучения: ' + option.text);
                            break;
                        }
                    }
                }
            """
            self.driver.execute_script(department_script)
            time.sleep(2)

            # После выбора формы обучения ждем загрузки групп
            time.sleep(3)

            # Устанавливаем значение для группы АЗИС 22-2
            group_script = """
                var groupSelect = document.getElementById('studentAdvert__controls--groups');
                if (groupSelect) {
                    // Ищем опцию с группой АЗИС 22-2
                    for (var i = 0; i < groupSelect.options.length; i++) {
                        var option = groupSelect.options[i];
                        if (option.text.includes('АЗИС 22-2') ||
                            (option.text.includes('АЗИС') && option.text.includes('22-2'))) {
                            groupSelect.value = option.value;
```

```
                    var event = new Event('change', { bubbles: true });
                    groupSelect.dispatchEvent(event);
                    console.log('Установлена группа: ' + option.text);
                    break;
                }
            }
        }
        """
        self.driver.execute_script(group_script)
        time.sleep(2)

        print("Предварительные значения установлены")

    except Exception as e:
        print(f"Ошибка при установке предварительных значений: {e}")

def scroll_to_element(self, element):
    """Прокрутка к элементу"""
    self.driver.execute_script("arguments[0].scrollIntoView({block: 'center',
behavior: 'smooth'});", element)
    time.sleep(0.5)

def get_available_options(self, element_id):
    """Получение всех доступных опций из выпадающего списка"""
    try:
        select_element = self.wait.until(
            EC.presence_of_element_located((By.ID, element_id))
        )
        self.scroll_to_element(select_element)
        select = Select(select_element)
        options = []
        for option in select.options:
            if option.get_attribute("value") and option.get_attribute("value") !=
"null":
                options.append({
                    'value': option.get_attribute("value"),
                    'text': option.text.strip(),
                    'visible': option.is_displayed()
                })
        return options
    except Exception:
        return []

def select_option_by_value(self, element_id, value):
    """Выбор опции по значению с обработкой исключений"""
```

```
        try:
            select_element = self.wait.until(
                EC.element_to_be_clickable((By.ID, element_id))
            )
            self.scroll_to_element(select_element)

            time.sleep(1)

            # Используем JavaScript для установки значения
            self.driver.execute_script(f"""
                var select = document.getElementById('{element_id}');
                if (select) {{
                    select.value = '{value}';
                    var event = new Event('change', {{ bubbles: true }});
                    select.dispatchEvent(event);
                }}
            """)

            time.sleep(1)

            # Проверяем, что значение установилось
            current_value = self.driver.execute_script(f"""
                return document.getElementById('{element_id}').value;
            """)

            if current_value == value:
                return True
            else:
                # Пробуем стандартный способ
                try:
                    select = Select(select_element)
                    select.select_by_value(value)
                    return True
                except:
                    return False

        except Exception:
            return False

    def verify_selection(self):
        """Проверка корректности выбранных значений"""
        print("\n=== ПРОВЕРКА ВЫБРАННЫХ ЗНАЧЕНИЙ ===")

        try:
            # Проверяем выбранную форму обучения
```

```
        department_value = self.driver.execute_script("""
            var select = document.getElementById('studentAdvert__controls--
department');
            return select ? select.value : null;
        """)

        department_text = self.driver.execute_script("""
            var select = document.getElementById('studentAdvert__controls--
department');
            if (select && select.selectedIndex >= 0) {
                return select.options[select.selectedIndex].text;
            }
            return null;
        """)

        print(f"Выбранная форма обучения: {department_text} (значение:
{department_value})")

        # Проверяем выбранную группу
        group_value = self.driver.execute_script("""
            var select = document.getElementById('studentAdvert__controls--
groups');
            return select ? select.value : null;
        """)

        group_text = self.driver.execute_script("""
            var select = document.getElementById('studentAdvert__controls--
groups');
            if (select && select.selectedIndex >= 0) {
                return select.options[select.selectedIndex].text;
            }
            return null;
        """)

        print(f"Выбранная группа: {group_text} (значение: {group_value})")

        # Проверяем, что значения установлены корректно
        department_ok = department_value and department_value != "null"
        group_ok = group_value and group_value != "null"

        if department_ok and group_ok:
            print("✓ Все значения выбраны корректно")
            return True
        else:
```

```
            print("✗ Не все значения выбраны корректно")
            return False

        except Exception as e:
            print(f"Ошибка при проверке выбранных значений: {e}")
            return False

    def find_and_select_option(self, element_id, search_text, option_type):
        """Поиск и выбор опции по тексту (резервный метод)"""
        try:
            options = self.get_available_options(element_id)
            if not options:
                print(f"Нет доступных {option_type}")
                return False

            # Ищем точное совпадение
            for option in options:
                if search_text.lower() in option['text'].lower():
                    print(f"Найдена {option_type}: {option['text']}")
                    return self.select_option_by_value(element_id, option['value'])

            # Если точное совпадение не найдено, ищем частичное
            for option in options:
                if any(word.lower() in option['text'].lower() for word in
search_text.split() if len(word) > 2):
                    print(f"Найдено приблизительное совпадение для {option_type}:
{option['text']}")
                    return self.select_option_by_value(element_id, option['value'])

            print(f"{option_type} '{search_text}' не найдена")
            print(f"Доступные {option_type}:")
            for option in options[:10]:
                print(f"  - {option['text']}")
            if len(options) > 10:
                print(f"  ... и еще {len(options) - 10} вариантов")
            return False

        except Exception as e:
            print(f"Ошибка при поиске {option_type}: {e}")
            return False

    def automatic_selection_azis_22_2(self):
        """Автоматический выбор группы АЗИС 22-2 заочной формы
(резервный метод)"""
        print("\n=== РЕЗЕРВНЫЙ ВЫБОР ГРУППЫ АЗИС 22-2 ===")
```

```
        # Выбираем заочную форму обучения
        department_search = "Заочная 3 года 10 мес"
        print(f"Ищем форму обучения: {department_search}")

        if self.find_and_select_option("studentAdvert__controls--department",
department_search, "формы обучения"):
            print("Форма обучения выбрана")
        else:
            # Пробуем альтернативные варианты
            alternative_departments = ["Заочная", "заочная"]
            for alt_dept in alternative_departments:
                print(f"Пробуем альтернативу: {alt_dept}")
                if self.find_and_select_option("studentAdvert__controls--department",
alt_dept, "формы обучения"):
                    print("Форма обучения выбрана (альтернативный вариант)")
                    break
            else:
                print("Не удалось выбрать форму обучения")
                return False

        time.sleep(3)

        # Выбираем группу АЗИС 22-2
        group_search = "АЗИС 22-2"
        print(f"Ищем группу: {group_search}")

        if self.find_and_select_option("studentAdvert__controls--groups",
group_search, "группы"):
            print("Группа выбрана")
        else:
            # Пробуем найти похожие группы
            similar_groups = ["АЗИС", "22-2"]
            for similar in similar_groups:
                print(f"Пробуем найти группу по: {similar}")
                if self.find_and_select_option("studentAdvert__controls--groups",
similar, "группы"):
                    print(f"Группа выбрана (похожий вариант: {similar})")
                    break
            else:
                print("Не удалось найти группу АЗИС 22-2")
                return False

        time.sleep(3)
        return True
```

```
    def show_schedule(self):
        """Показать расписание"""
        try:
            # Ждем появления кнопки
            show_button = self.wait.until(
                EC.element_to_be_clickable((By.XPATH, "//button[contains(text(),
'Показать расписание')]"))
            )

            self.scroll_to_element(show_button)

            # Пробуем кликнуть через JavaScript
            self.driver.execute_script("arguments[0].click();", show_button)

            print("Нажата кнопка 'Показать расписание'")
            time.sleep(5)
            return True

        except Exception as e:
            print(f"Ошибка при нажатии кнопки: {e}")
            # Пробуем альтернативный способ
            try:
                self.driver.execute_script("""
                    var buttons = document.querySelectorAll('button.btn-primary');
                    for (var i = 0; i < buttons.length; i++) {
                        if (buttons[i].textContent.includes('Показать') ||
buttons[i].textContent.includes('показать')) {
                            buttons[i].click();
                            return true;
                        }
                    }
                    return false;
                """)
                print("Кнопка нажата через JavaScript")
                time.sleep(5)
                return True
            except:
                print("Не удалось нажать кнопку")
                return False

    def get_full_schedule(self):
        """Получение полного текста расписания"""
        try:
            # Ждем появления расписания
```

```
            self.wait.until(
                EC.presence_of_element_located((By.ID, "printable"))
            )

            time.sleep(3)

            # Получаем весь текст из блока printable
            printable = self.driver.find_element(By.ID, "printable")
            full_schedule = printable.text

            # Если текст короткий, пробуем получить через JavaScript
            if len(full_schedule) < 100:
                full_schedule = self.driver.execute_script("""
                    return document.getElementById('printable').innerText;
                """)

            return full_schedule

        except Exception as e:
            print(f"Ошибка при получении расписания: {e}")
            try:
                # Пробуем найти таблицу расписания
                tables = self.driver.find_elements(By.TAG_NAME, "table")
                if tables:
                    return tables[0].text
            except:
                pass

            return "Не удалось получить расписание"

    def format_schedule(self, schedule_text):
        """Форматирование расписания для красивого вывода"""
        if not schedule_text or len(schedule_text.strip()) < 50:
            return "Расписание не найдено или слишком короткое"

        lines = schedule_text.split('\n')
        formatted_schedule = []
        current_day = ""

        for line in lines:
            line = line.strip()
            if not line:
                continue

            # Проверяем, является ли строка заголовком дня
```

```
            day_keywords = ["понедельник", "вторник", "среда", "четверг",
"пятница", "суббота", "воскресенье"]
            if any(day in line.lower() for day in day_keywords):
                # Если у нас уже есть текущий день, добавляем разделитель
                if current_day:
                    formatted_schedule.append("—" * 80)
                current_day = line
                formatted_schedule.append(f"\n{line.upper()}")
                formatted_schedule.append("—" * 80)
            elif "пара дисциплина преподаватель" in line.lower():
                # Пропускаем заголовок таблицы
                continue
            elif line and not line.isspace():
                # Добавляем пару с отступом
                formatted_schedule.append(f"  {line}")

        return "\n".join(formatted_schedule)

    def display_schedule(self, schedule_text):
        """Отображение форматированного расписания"""
        if not schedule_text or len(schedule_text.strip()) < 50:
            print("Расписание не найдено или слишком короткое")
            return False

        formatted_schedule = self.format_schedule(schedule_text)

        print("\n" + "=" * 80)
        print("РАСПИСАНИЕ ГРУППЫ АЗИС 22-2")
        print("=" * 80)
        print(formatted_schedule)
        print("=" * 80)

        # Подсчет дней и пар
        lines = schedule_text.split('\n')
        days_count = sum(1 for line in lines if any(day in line.lower() for day in
                                        ["понедельник", "вторник", "среда", "четверг",
"пятница", "суббота",
                                         "воскресенье"]))
        pairs_count = sum(1 for line in lines if "пара" in line.lower() and
                          not "пара дисциплина преподаватель" in line.lower())

        print(f"Всего дней: {days_count}")
        print(f"Всего пар: {pairs_count}")
        print("=" * 80)
```

```
        return True


def main():
    """Основная функция"""
    tester = UniversityScheduleTester()

    try:
        # Открываем страницу с предустановленными значениями
        print("Загружаем страницу расписания с предустановленными значениями...")
        tester.open_schedule_page_with_preselection()

        # Проверяем, что значения установились корректно
        if not tester.verify_selection():
            print("Предварительные значения не установились, используем резервный метод...")
            if not tester.automatic_selection_azis_22_2():
                print("Не удалось выбрать параметры. Завершение работы.")
                return

        # Показываем расписание
        print("\nЗагружаем расписание...")
        if tester.show_schedule():
            # Получаем полное расписание
            schedule_text = tester.get_full_schedule()

            # Отображаем форматированное расписание
            tester.display_schedule(schedule_text)

        else:
            print("Не удалось загрузить расписание")

        print("\nГотово! Расписание загружено.")

    except Exception as e:
        print(f"Произошла ошибка: {e}")
        import traceback
        traceback.print_exc()
    except KeyboardInterrupt:
        print("\nПрограмма прервана пользователем")
    finally:
        tester.close_browser()
```

```
if __name__ == "__main__":
    main()
```