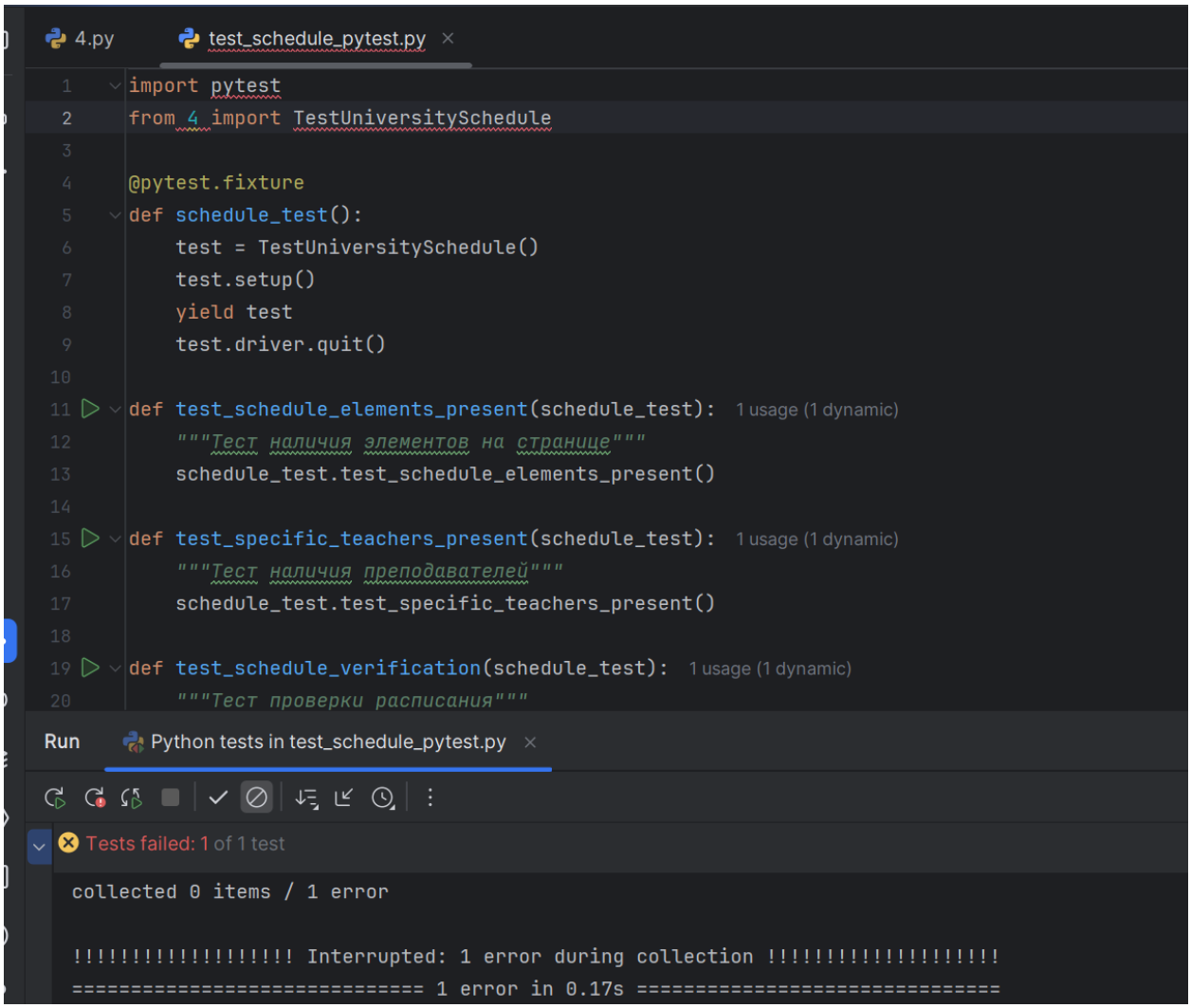
Практическая работа №4

Запуск pytest с ошибкой:



Запуск pytest без ошибки:

Мы запускаем программу и ждем пока браузер откроет нужную нам страницу. После открытия страницы, программа предлагает нам выбрать форму обучения и группу. Выбираем и получаем на выход расписание нашей группы, которое находиться на странице.

```
================================================
          ВЫБОР ФОРМЫ ОБУЧЕНИЯ И ГРУППЫ
================================================

ДОСТУПНЫЕ ФОРМЫ ОБУЧЕНИЯ
1. Очная
2. Очно-заочная
3. Переводчики
4. Заочная 3 года 10 мес
5. Заочная 5 лет

Выберите номер (1-5): 4
Выбрано: Заочная 3 года 10 мес
Форма обучения выбрана: Заочная 3 года 10 мес
```

```
================================================
          ВЫБОР ФОРМЫ ОБУЧЕНИЯ И ГРУППЫ
================================================

ДОСТУПНЫЕ ФОРМЫ ОБУЧЕНИЯ
1. Очная
2. Очно-заочная
3. Переводчики
4. Заочная 3 года 10 мес
5. Заочная 5 лет

Выберите номер (1-5):
```

1.Выбор формы обучения

```
ДОСТУПНЫЕ ГРУППЫ
1. АЗИ 25-1
2. АЗИС 25-1
3. АЗМ 25-1
4. АЗМ 25-2
5. АЗМ 25-3
6. АЗМ 25-4
7. АЗР 25-1
8. АЗИ 24-1
9. АЗИС 24-1
10. АЗМ 24-1
11. АЗМ 24-2
12. АЗМ 24-3
13. АЗР 24-1
14. АЗИ 23-1
15. АЗИ 23-3
16. АЗИС 23-1
17. АЗМ 23-1
18. АЗМ 23-2
19. АЗР 23-1
20. АЗИ 22-1
21. АЗИС 22-1
22. АЗИС 22-2
23. АЗМ 22-1
24. АЗМ 22-2
25. АЗМ 22-5

Выберите номер (1-30): 21
Выбрано: АЗИС 22-1
Группа выбрана: АЗИС 22-1

Загружаем расписание...
Кнопка нажата
```

2.Выбор группы

```
================================================================================
                        ПОЛНОЕ РАСПИСАНИЕ
================================================================================
  1. Понедельник, 22 Сентября 2025
  2. Пара Дисциплина Преподаватель Дистанционно/Ауд. Примечание Неделя
  3. 2 пара / 10:10-11:40 Стандартизация и сертификация в информационных системах / Пз. Мельникова Оксана Юрьевна 324 Четная
  4. 3 пара / 12:10-13:40 Стандартизация и сертификация в информационных системах / Пз. Мельникова Оксана Юрьевна 324 Четная
  5. 4 пара / 13:50-15:20 Стандартизация и сертификация в информационных системах / Пз. Мельникова Оксана Юрьевна 324 Четная
  6. 5 пара / 15:30-17:00 Стандартизация и сертификация в информационных системах / Зач. Мельникова Оксана Юрьевна 324 Четная
  7. Вторник, 23 Сентября 2025
  8. Пара Дисциплина Преподаватель Дистанционно/Ауд. Примечание Неделя
  9. 2 пара / 10:10-11:40 Инфокоммуникационные системы и сети / Лб. Гуськова Юлия Александровна 220 Четная
 10. 3 пара / 12:10-13:40 Инфокоммуникационные системы и сети / Лб. Гуськова Юлия Александровна 220 Четная
 11. 4 пара / 13:50-15:20 Инфокоммуникационные системы и сети / Лб. Гуськова Юлия Александровна 220 Четная
 12. 5 пара / 15:30-17:00 Инфокоммуникационные системы и сети / Лб. Гуськова Юлия Александровна 220 Четная
 13. Среда, 24 Сентября 2025
 14. Пара Дисциплина Преподаватель Дистанционно/Ауд. Примечание Неделя
 15. 1 пара / 08:30-10:00 Эксплуатация и модификация информационных систем / Лб. Жидкова Наталья Валерьевна 226 Четная
 16. 2 пара / 10:10-11:40 Эксплуатация и модификация информационных систем / Лб. Жидкова Наталья Валерьевна 226 Четная
 17. 3 пара / 12:10-13:40 Эксплуатация и модификация информационных систем / Лк. Жидкова Наталья Валерьевна 320 Четная
 18. Четверг, 25 Сентября 2025
 19. Пара Дисциплина Преподаватель Дистанционно/Ауд. Примечание Неделя
 20. 1 пара / 08:30-10:00 Надежность и отказоустойчивость информационных систем / Пз. Жидкова Наталья Валерьевна 226 Четная
 21. 2 пара / 10:10-11:40 Надежность и отказоустойчивость информационных систем / Пз. Жидкова Наталья Валерьевна 226 Четная
 22. Пятница, 26 Сентября 2025
 23. Пара Дисциплина Преподаватель Дистанционно/Ауд. Примечание Неделя
 24. 1 пара / 08:30-10:00 Анализ больших данных / Лб. Рябов Антон Владимирович 324 Четная
```

3.На выходе получаем наше расписание

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
from datetime import datetime


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(self):
        """Открытие страницы расписания"""
        self.driver.get('https://api.nntu.ru/raspisanie')
        print("Страница расписания открыта")
        time.sleep(3)

    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)

            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)
            current_value = self.driver.execute_script(f"""
                return document.getElementById('{element_id}').value;
            """)

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

        except Exception:
            return False

    def display_available_options(self, options, title):
        """Отображение доступных опций для выбора"""
        print(f"\n{title}")
        if not options:
            print("Нет доступных опций")
            return None

        for i, option in enumerate(options, 1):
```

```
            print(f"{i}. {option['text']}")

        while True:
            try:
                choice = input(f"\nВыберите номер (1-{len(options)}): ").strip()
                if choice == '':
                    print("Необходимо ввести номер")
                    continue

                choice_num = int(choice)
                if 1 <= choice_num <= len(options):
                    selected_option = options[choice_num - 1]
                    print(f"Выбрано: {selected_option['text']}")
                    return selected_option
                else:
                    print(f"Введите число от 1 до {len(options)}")
            except ValueError:
                print("Введите корректный номер")
            except KeyboardInterrupt:
                print("Операция прервана")
                return None

    def manual_selection(self):
        """Ручной выбор формы обучения и группы"""
        print("\n" + "=" * 50)
        print("          ВЫБОР ФОРМЫ ОБУЧЕНИЯ И ГРУППЫ")
        print("=" * 50)

        # Получаем доступные формы обучения
        departments = self.get_available_options("studentAdvert__controls--
department")
        selected_dept = self.display_available_options(departments, "ДОСТУПНЫЕ
ФОРМЫ ОБУЧЕНИЯ")

        if not selected_dept:
            return None, None

        # Выбираем формы обучения
        if self.select_option_by_value("studentAdvert__controls--department",
selected_dept['value']):
            print(f"Форма обучения выбрана: {selected_dept['text']}")
        else:
```

```
            print(f"Не удалось выбрать форму обучения")
            return None, None

        time.sleep(3)

        # Получаем доступные группы
        groups = self.get_available_options("studentAdvert__controls--groups")

        if not groups:
            print("Нет доступных групп для выбранной формы обучения")
            time.sleep(2)
            groups = self.get_available_options("studentAdvert__controls--groups")
            if not groups:
                return selected_dept, None

        selected_group = self.display_available_options(groups, "ДОСТУПНЫЕ
ГРУППЫ")

        if not selected_group:
            return selected_dept, None

        # Выбираем группу
        if self.select_option_by_value("studentAdvert__controls--groups",
selected_group['value']):
            print(f"Группа выбрана: {selected_group['text']}")
        else:
            print(f"Не удалось выбрать группу")
            return selected_dept, None

        time.sleep(3)
        return selected_dept, selected_group

    def show_schedule(self):
        """Показать расписание"""
        try:
            show_button = self.wait.until(
                EC.element_to_be_clickable((By.CSS_SELECTOR, "button.btn-
primary"))
            )

            self.scroll_to_element(show_button)
            self.actions.move_to_element(show_button).click().perform()
```

```
            time.sleep(1)
            show_button.click()

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

        except Exception:
            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].click();
                            break;
                        }
                    }
                """)
                print("Кнопка нажата")
                time.sleep(5)
                return True
            except:
                return False

    def get_full_schedule(self):
        """Получение полного текста расписания"""
        try:
            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:
            try:
                # Пробуем найти таблицу расписания
                tables = self.driver.find_elements(By.TAG_NAME, "table")
                if tables:
                    return tables[0].text
            except:
                pass

            # Пробуем получить весь текст страницы
            try:
                return self.driver.find_element(By.TAG_NAME, "body").text
            except:
                return "Не удалось получить расписание"

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

        print("\n" + "=" * 80)
        print("                    ПОЛНОЕ РАСПИСАНИЕ")
        print("=" * 80)

        # Разделяем расписание на строки и выводим с нумерацией
        lines = schedule_text.split('\n')
        for i, line in enumerate(lines, 1):
            if line.strip():  # Пропускаем пустые строки
                print(f"{i:3d}. {line}")

        print("=" * 80)
        print(f"Всего строк: {len([l for l in lines if l.strip()])}")
        print(f"Общий размер: {len(schedule_text)} символов")
        print("=" * 80)

        return True
```

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

    try:
        # Открываем страницу
        tester.open_schedule_page()

        # Ручной выбор параметров
        dept, group = tester.manual_selection()

        if not group:
            print("Не удалось выбрать группу. Завершение работы.")
            return

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

            # Отображаем полное расписание
            tester.display_schedule(schedule_text)

        input("\nНажмите Enter для выхода...")

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


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