

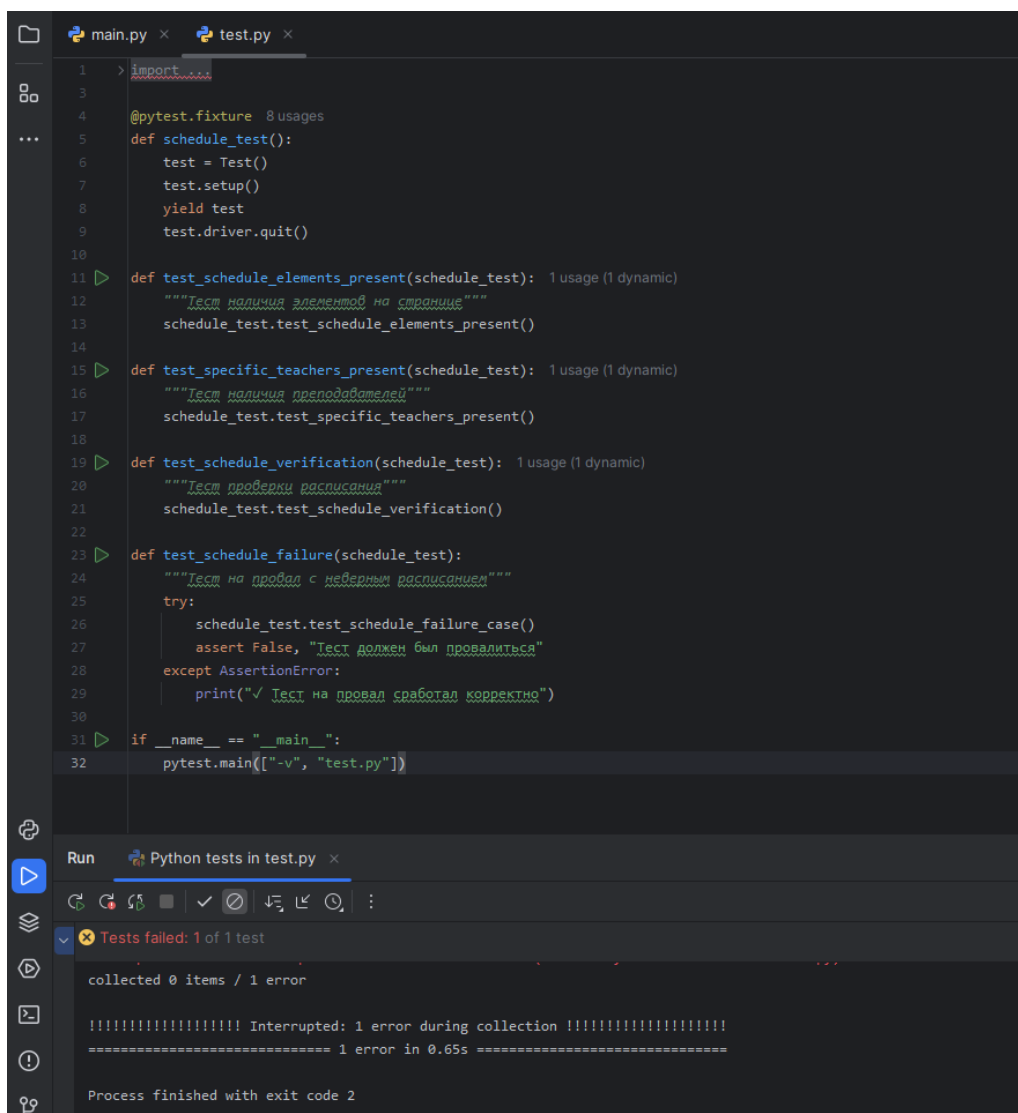
Практическая работа №4

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.

1.Запуск pytest с ошибкой



The screenshot shows a code editor with two tabs: `main.py` and `test.py`. The `test.py` file contains the following code:

```
1 > import ...
3
4 @pytest.fixture 8 usages
5 def schedule_test():
6     test = Test()
7     test.setup()
8     yield test
9     test.driver.quit()
10
11 def test_schedule_elements_present(schedule_test): 1 usage (1 dynamic)
12     """Тест наличия элементов на странице"""
13     schedule_test.test_schedule_elements_present()
14
15 def test_specific_teachers_present(schedule_test): 1 usage (1 dynamic)
16     """Тест наличия преподавателей"""
17     schedule_test.test_specific_teachers_present()
18
19 def test_schedule_verification(schedule_test): 1 usage (1 dynamic)
20     """Тест проверки расписания"""
21     schedule_test.test_schedule_verification()
22
23 def test_schedule_failure(schedule_test):
24     """Тест на провал с неверным расписанием"""
25     try:
26         schedule_test.test_schedule_failure_case()
27         assert False, "Тест должен был провалиться"
28     except AssertionError:
29         print("✓ Тест на провал сработал корректно")
30
31 if __name__ == "__main__":
32     pytest.main(["-v", "test.py"])
```

Below the code editor, the `Run` button is clicked, and the output window shows the following message:

```
Python tests in test.py x
Tests failed: 1 of 1 test
collected 0 items / 1 error
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
===== 1 error in 0.65s =====
Process finished with exit code 2
```

2.Запуск pytest без ошибки

Инструкция: При запуске нашей программы, открывается браузер на странице расписания с изначально выбранными параметрами, а именно нашей формой обучения(Заочная 3 года 10 мес) и нашей группой(АЗИС 22-2). После чего на выходе получаем актуальное расписание.

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

```
СРЕДА, 5 НОЯБРЯ 2025
```

```
5 пара / 15:30-17:00 Корпоративные информационные системы / Зач.Оц. Гуськова Юлия Александровна 220 Четная
1 пара вечер / 17:30-19:00 Теория цифровой обработки сигналов / Экз. Абаймов Анатолий Вячеславович 220 Четная
2 пара вечер / 19:10-20:40 Теория цифровой обработки сигналов / Экз. Абаймов Анатолий Вячеславович 220 Четная
```

```
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 320 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 320 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: 4
```

```
Всего пар: 12
=====
```

```
Готово! Расписание загружено.
```

```
Браузер закрыт
```

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(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)

        # 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 find_and_select_option(self, element_id, search_text, option_type):
    """Поиск и выбор опции по тексту"""
    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"""
    print("\n=== АВТОМАТИЧЕСКИЙ 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"Попробуем 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)

        # Попробуем 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}")
        # Попробуем 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;
            """)
        except:
            pass

```

```

        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
        print("Загружаем страницу расписания...")
        tester.open_schedule_page()

        # Автоматически выбираем группу АЗИС 22-2
        print("Выбираем параметры...")
        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()
```