OTUS

Python QA Engineer

Курс по активной прокачке навыков программирования на Python для QA-инженеров

Длительность курса: 146 академических часов

1 Введение в автоматизацию тестирования

1 **Автоматизация тестирования: введение.**

Цели занятия:

создать свой проект на github, в который будут выкладываться ДЗ; проанализировать виды, цели, инструменты и инфраструктуру для автотестов.

Краткое содержание:

Пирамида автоматизированного тестирования; краткая история поколений инструментов; актуальные технологии для автоматизации.

Домашние задания

1 Настраиваем окружение

Цель: Прислать свой первый Pull Request в рамках курса

1. Зарегистрироваться на гитхабе
2. Создать репо
3. Создать веточку
4. Залить в ветку первый код с выводом "hello, world"
5. Сделать pull request
6. Прислать pull request

Инструменты для прохождения курса:
1. docker https://docs.docker.com/get-docker/
2. PyCharm Community Edition (либо другая IDE, на ваш вкус)
3. git https://git-scm.com/
4. Python 3.6+
5. VirtualBox https://www.virtualbox.org/wiki/Downloads (опционально)

2 **Введение в Pytest**

Цели занятия:

запускать тесты;
писать параметризованные тесты;
пользоваться фикстурами;
добавить в проект библиотеку pytest.

Краткое содержание:

основные особенности фреймворка и использование фикстур;
фикстура;
фреймворк pytest.

---

3 **Погружение в Python. ООП**

Цели занятия:

разобрать реализацию основных понятий ООП в Python.

Краткое содержание:

концепция ООП;
реализация ООП в Python;
практические примеры.

Домашние задания

1 ООП и Pytest на практике

Цель: Научиться писать код в ООП стиле и покрывать его тестами.

Создать базовый класс геометрической фигуры (Figure).
Реализовать классы геометрических фигур Треугольник, Прямоугольник, Квадрат, Круг (Triangle, Rectangle, Square, Circle).
- Каждый класс должен располагаться в отдельном файле с соответствующим названием (например class Triangle => Triangle.py).
- Все файлы с классами должны находиться в папке src/ в корне репозитория.
- Треугольник должен задаваться тремя

сторонами, если треугольник создать нельзя то выбрасывать ошибку raise ValueError.

1 Часть.

Каждая фигура должна иметь атрибуты:
name - название фигуры,
area (вычисляемое!) - площадь,
perimeter (вычисляемое!) - периметр (сумма длин сторон или длину окружности)

Все вычисляемые свойства должны вычисляться по формулам для соответствующих геометрических фигур (никакого хардкода значений).
Каждая фигура должна реализовать метод add_area(figure) который должен принимать другую геометрическую фигуру и возвращать сумму площадей этих фигур.
Если метод передана не геометрическая фигура, то нужно выбрасывать ошибку (raise ValueError).

Пример работы с одним из классов фигуры:

```
>>> square = Square(10) # Так создаем квадрат со стороной 10
>>> square.area
100
>>> triangle = Triangle(13, 14, 15) # Так создаем треугольник со сторонами 13, 14, 15
>>> triangle.area
84
>>> triangle.add_area(square)
184
```

2. Часть.

Написать тесты с использованием pytest на эти классы.
Глубину покрытия и объем определить самостоятельно, но минимум проверить реализацию всех указанных требований для каждого класса.
- Все тесты должны располагаться в папке tests/ в корне репозитория.

4	**Погружение в Python: Функциональное программирование**	**Цели занятия:** разобрать парадигму функционального программирования **Краткое содержание:** методы функционального программирования; продвинутое использование функций.

2 Тестирование API

1 Работа с тестовыми данными

Цели занятия:

работать с файлами основных типов (csv, xml, json): познакомиться с менеджерами контекста.

Краткое содержание:

работа с различными типами данных в Python; контекстные менеджеры.

Домашние задания

1 Работа с тестовыми данными

Цель: Научиться работать с различными типами файлов.

Работа с тестовыми данными

1. Скачать файлы: https://github.com/konflic/front_example/blob/master/data/books.csv и https://github.com/konflic/front_example/blob/master/data/users.json.
2. Написать скрипт, который из двух данных файлов будет читать данные и на их основании создаст result.json файл со структурой: https://github.com/konflic/front_example/blob/master/data/reference.json.
3. Идея в том что нужно раздать все книги из csv файла пользователям из списка. Книги складываются в виде словарей в массив books у каждого пользователя.
4. Книг изначально больше чем пользователей, поэтому раздавать нужно по принципу "максимально поровну", т.е. если книг, например 10. а пользователей 3 то распределение будет таким - 4 3 3 (один получит оставшуюся книгу).
5. Итоговая структура должна соответствовать стандарту json и парситься соответствующей библиотекой.

2 **Тестирование API**

Цели занятия:

тестировать REST API-сервисы;
работать с библиотекой requests.

Краткое содержание:

подходы к реализации REST;
использование библиотеки requests, json для тестирования API.

---

3 **DDT в тестировании API**

Цели занятия:

работать с большим количеством тестовых данных;
шаблонизировать тесты;
использовать генераторы в тестах;
работать с библиотеками генерации данных;
работать с техникой pairwise;

Краткое содержание:

параметризация тестов на pytest;
подход pairwise;

Домашние задания

1 Тестирование API

Цель: Поучиться тестировать API сервисов на основе их документации.

Тестирование каждого api оформить в отдельном тестовом модуле.

1. Тестирование REST сервиса 1
Написать минимум 5 тестов для REST API сервиса: https://dog.ceo/dog-api/.
Как минимум 2 из 5 должны использовать параметризацию.
Документация к API есть на сайте.
Тесты должны успешно проходить.

2. Тестирование REST сервиса 2
Написать минимум 5 тестов для REST API сервиса:

https://www.openbrewerydb.org/.
Как минимум 2 из 5 должны использовать параметризацию.
Документация к API есть на сайте.
Тесты должны успешно проходить.

3. Тестирование REST сервиса 3.
Написать минимум 5 тестов для REST API сервиса: https://jsonplaceholder.typicode.com/.
Как минимум 2 из 5 должны использовать параметризацию.
Документация к API есть на сайте.
Тесты должны успешно проходить.

4. Реализуйте в отдельном модуле (файле) тестовую функцию которая будет принимать 2 параметра:
url - должно быть значение по умолчанию https://ya.ru
status_code - значение по умолчанию 200
Параметры должны быть реализованы через pytest.addoption.
Можно положить фикстуру и тестовую функцию в один файл.
Основная задача чтобы ваш тест проверял по переданному урлу статус ответа тот который передали,
т.е. по адресу https://ya.ru/sfhfhfhfhfhfhfhfh должен быть валидным ответ 404

пример запуска pytest test_module.py --url=https://mail.ru --status_code=200

---

4 **Консультация по домашним заданиям**

Краткое содержание:

Разбираем алгоритмы раздачи книг пользователя на примерах из домашних работ.

3 Тестирование UI

1 Основы Web-разработки

Цели занятия:

познакомиться с устройством веб-приложений, для того чтобы уметь искать более эффективне подходы к тестированию.

Краткое содержание:

Устройство современных веб-приложений;
Роль css, js и html в веб-приложениях;
Разделение на backend и frontend;
Мобильный веб и его особенности.

---

2 Введение в тестирование Web UI, Selenium WebDriver

Цели занятия:

запускать и останавливать браузеры с помощью Selenium;
готовить инфрастуктуру для запуска ui тестов.

Краткое содержание:

концепция веб-драйвера;
стандрт W3C Driver.

---

3 Поиск элементов

Цели занятия:

искать элементы с помощью Selenium, писать свои отказоустойчивые и стабильные селекторы.

Краткое содержание:

типы локаторов в Selenium;
особенности поиска элементов;
обработка ошибок при поиске элементов;
какие локаторы лучше использовать.

---

4 Ожидания элементов

Цели занятия:

научиться работать с ожиданиями элементов.

Краткое содержание:

виды ожиданий, написание кастомных ожиданий;
обсуждение реализации тесов для приложений с AJAX;
работа с исключениями в Python

Домашние задания

1 Написание простых автотестов и основы Selenium

Цель: Научиться настраивать окружение для Selenium тестов, написать тесты, настроить ожидания к проекту. Научиться писать простые selenium скрипты.

Часть 1
1.1. Написать фикстуру для запуска разных браузеров (firefox, chrome, opera, по желанию - safari, edge, yandex).
1.2. Выбор браузера должен осуществляться путем передачи аргумента командной строки pytest.
1.3. По завершению работы тестов должно осуществляться закрытие браузера.
1.4. Добавить опцию командной строки, которая указывает базовый URL opencart.

Часть 2
2.1 Написать тесты проверяющие элементарное наличие элементов на разных страницах приложения opencart
2.2 Реализовать минимум пять тестов (одни тест = одна страница приложения)
2.3 Использовать методы явного ожидания элементов

Покрыть нужно:
- Главную
- Каталог
- Карточку товара
- Страницу логина в админку /admin
- Страницу регистрации пользователя (/index.php?route=account/register)

* Какие именно элементы проверять определить самостоятельно, но не меньше 5 для каждой страницы

---

5 **Работа с окнами, iframes, cookies**

Цели занятия:

научиться работать с элементами iframe, вкладками браузера и куками веб-приложений.

Краткое содержание:

одновременная работа с несколькими вкладками;
работа с iframe элементами;
работа с куками в веб-приложениях;
загрузка файлов.

---

6 **WebElement и работа с ним**

Цели занятия:

проанализировать свойства объекта WebElement;
научиться выполнять сложные действия с помощью ActionChains.

Краткое содержание:

разница между атрибутом и свойством элемента;
класс ActionChains и его использование;
гибкий подход к работе с вводом текстовых данных.

---

7 **Паттерн PageObject**

Цели занятия:

применить паттерн PageObject для автотестов;
научиться выделять общие компоненты объектов страниц.

Краткое содержание:

архитектурные паттерны и применение их в автотестриовании.

Домашние задания

1 PageObject

Цель: Научиться реализовывать PageObject шаблон в автотестах.

1. Переписать уже имеющиеся тесты в проекта opencart на PageObject паттерн.
2. Добавить автотесты на следующие сценарии.
2.1. Добавление нового товара в разделе администратора.
2.2. Удаление товара из списка в разделе администартора.
2.3. Регистрация нового пользователя в магазине опенкарта.
2.4. Переключение валют из верхнего меню опенкарта.

---

8 **Логгирование и протоколирование**

Цели занятия:

обсудить особенности и сложности касающиеся построения систем логгирования, получение доступа к логам и событиям WebDriver'a.

Краткое содержание:

логировать действия Selenium;
использовать Listener для действий браузера;
использовать библиотеку logging и внедрять её в проект;
получать js логи браузера.

---

9 **Отчёты Allure**

Цели занятия:

строить отчёты по результатам тестирования.

Краткое содержание:

инструмент Allure для генерации отчётов;
основные настройки, библиотека для подключения в python.

Домашние задания

1 Логгирование и отчётность

Цель: Научиться внедрять в проект логгирование и отчётность allure

1. Настройте логирование внутри PageObject'ов. (черед модуль logging).
2. Добавьте аннотации для шагов и тестов для трансляции в отчёт Allure.
3. Настройте Selenoid, добавьте несколько браузеров и запустите на них тесты.
4. Предусмотрите возможность запуска тестов как на удаленных сервисах так и локально.
5. Предусмотрите снятие скриншота и добавление его в отчёт при падении тестов.

---

10 **Удаленный запуск (Grid)**

Цели занятия:

запускать тесты на удаленной машине;
запустить тесты в облаке.

Краткое содержание:

настройка вебдрайвер для запуска на удаленном сервере.

---

11 **Selenoid**

Цели занятия:

объяснить как Selenoid решает проблему запуска тестов.

Краткое содержание:

основные настройки;
особенности конфигурирования.

4 Мобильное тестирование

1	**Введение в Appium**	**Цели занятия:** Настраивать Appium и окружение (windows/mac)
2	**Нативные и грибридные приложения**	**Цели занятия:** Разрабатывать автотесты как для нативных, так и для гибридных приложений
3	**Тестирование Swipe жестов, Reporting, Listeners**	
4	**Запуск автотестов на реальных устройствах**	

5 Работа с бекендом

1 Архитектура Линукс

Цели занятия:

работать с инструментами диагностики неисправностей Linux;
работать с утилитами ping, netstat, ip.;
диагностировать проблемы на уровне сети.
находить причины неисправностей, которые возникли в ходе работы Linux.

Краткое содержание:

файловая система Linux и работа процессов;
использование инструментов для дебага, как strace, gdb.

2 **Работа с ОС Linux с помощью Python**

Цели занятия:

работать с операционной системой Linux средствами Python.

Краткое содержание:

использование модулей system, subprocess, sys из стандартной библиотеки Python.

Домашние задания

1 Работа с ОС

Цель: Попрактиковаться в работе с процессами ОС Linux

Для выполнения задания нужно написать парсер системных процессов команды 'ps aux' на языке Python с использованием стандартной библиотеки и модуля subprocess.
Парсер должен вывести в стандартный вывод в качестве результата работы следующую информацию (все цифры и данные для примера):

Отчёт о состоянии системы:
Пользователи системы: 'root', 'user1', ...
Процессов запущено: 833

Пользовательских процессов:
root: 533
user1: 231
...

Всего памяти используется: 553.3 mb
Всего CPU используется: 33.2%
Больше всего памяти использует: (%имя процесса, первые 20 символов если оно длиннее)
Больше всего CPU использует: (%имя процесса, первые 20 символов если оно длиннее)

Так же этот отчёт должен быть сохранён в отдельный txt файл с названием текущей даты и времени проверки.
Например, 10-12-2021-12:15-scan.txt

3 **Анализ логов веб-сервера**

Цели занятия:

выявлять ошибки в бекенде;
использовать утилиты grep, awk, find.

Краткое содержание:

поиск причины неисправности по логам работы веб-сервера.

Домашние задания

1 Анализ лога веб-сервера

Цель: Потренироваться писать парсеры для логов.

Написать скрипт анализа приложенного access.log файла

Формат записи в файле лога:
%h %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %D

%h - имя удаленного хоста
%t - время получения запроса
%r - тип запроса, его содержимое и версия
%s - код состояния HTTP
%b - количество отданных сервером байт
%{Referer} - URL-источник запроса
%{User-Agent} - HTTP-заголовок, содержащий информацию о запросе
%D - длительность запроса в миллисекундах

Требования к реализации
1. Должна быть возможность указать директорий где искать логи или конкретный файл
2. Должна быть возможность обработки всех логов внутри одного директория
3. Для access.log должна собираться следующая информация:
- общее количество выполненных запросов
- количество запросов по HTTP-методам: GET - 20, POST - 10 и т.п. (список необходимых методов можно посмотреть в RFC: https://datatracker.ietf.org/doc/html/rfc7231#section-4 и https://datatracker.ietf.org/doc/html/rfc5789#section-2)
- топ 3 IP адресов, с которых были сделаны

запросы
- топ 3 самых долгих запросов, должно быть видно метод, url, ip, длительность, дату и время запроса
4. Собранная статистика должна быть сохранена в json файл и выведена в терминал в свободном (но понятном!) формате
5. Должен быть README.md файл, который описывает как работает скрипт

---

4 **Работа с сетью I (SSH, FTP)**

Цели занятия:

настраивать автопроверку кодстайла;
использовать аннотации типов и модуль mypy;
работать с протоколом FTP для передачи файлов;
работать с протоколом SSH для управления сервером.

Краткое содержание:

работа с протоколами ssh и ftp;
flake8, pylint, black;
mypy для аннотаций типов.

---

5 **Работа с сетью II (socket)**

Цели занятия:

использовать библиотеку socket;
настраивать TCP соединение.

Краткое содержание:

Библиотека socket и низкоуровневое сетевое взаимодействие.

Домашние задания

1 Echo сервер

Цель: Попрактиковаться с использованием модуля socket и сетевым взаимодействием по протоколу HTTP

В этом задании реализуем собственный echo-сервер с использованием библиотеки socket. Для выполнения задания использовать только модули стандартной библиотеки.

Ваш сервер должен принимать запрос от клиента и отправлять ему:
1) заголовки, которые получил в запросе
2) метод, которым был сделан запрос
3) выставлять статус, который передал клиент в GET параметре status (т.е. если передали /?status=404 то ответ будет со статусом 404) если параметр не передан или не валидный то отдавать 200, если значение
4) (опционально) сервер не должен прекращать работу после ответа клиенту, а продолжать ожидать следующего соединения

Заголовки должны отображаться на странице в виде строк текста:

Request Method: GET
Request Source: ('127.0.0.1', 47296)
Response Status: 200 OK
header-name: header-value
header-name: header-value
...

Для того чтобы правильно выставить значение статуса нужно отправить числовое значение и фразу например 200 OK, 404 Not Found и т.д. самый оптимальный способ для поиска нужной фразы статуса это воспользоваться модулем http из стандартной библиотеки python https://docs.python.org/3/library/http.html

---

6 Работа с БД

Цели занятия:

изучить работу с базами данных и серверами управления базами данных используя Python и ORM SQLAlchemy.

Краткое содержание:

выполнение CRUD операций на различных БД и СУБД; использование ORM SQLAlchemy для работы с БД.

6 DevOps

1 Введение в Docker и контейнеризацию

Цели занятия:

собирать собственные Docker контейнеры.

Краткое содержание:

строение контейнеризация;
основные термины и определения экосистемы Docker;
основы сборки собственных контейнеров.

---

2 Оркестрация и взаимодействие контейнеров

Цели занятия:

рассмотреть проблематику организации приложений в распределенной системе контейнеров, поговорить про docker-compose и его применение.

Краткое содержание:

сетевое взаимодействие между контейнерами;
промежуточное хранение данных в контейнерах;
композиция контейнеров с помощью docker-compose.

Домашние задания

1 Написать Dockerfile для своего проекта

Цель: Научиться заворачивать свои проекты в Docker

1. Написать Dockerfile с помощью которого можно запустить ваши тесты на opencart
2. Запуск должен сохранить возможность передачи параметров при запуске.

---

3 **Непрерывная интеграция, Jenkins**

Цели занятия:

установить и настроить Jenkins CI.

Краткое содержание:

основы непрерывной интеграции в контексте тестирования.

---

4 **Подготовка тестового окружения**

Цели занятия:

подготовить тестовое окружение, собирать deb и whl пакеты.

Краткое содержание:

основы сборки пакетов;
основные практики подготовки тестового окружения.

Домашние задания

1 Запуск автотестов с использованием Jenkins

Цель: Научиться поднимать Jenkins и выполнять базовую конфигурацию джоб

1. Понять Jenkins любым способом (docker, jar ...)
2. Настроить джобу которая будет запускать ваши автотесты на opencart
2.1 Джоба должна подтягивать данные из githuib
2.2. Джоба должна генерировать отчёт allure
2.3 Джоба должна принимать в качестве параметров:
- Адрес executor (selenoid, browserstack, ip)
- Адрес приложения opencart
- Количество потоков
- Браузер
- Версию браузера
2.4 Желательно запускать тесты в docker (опционально)

7 Другие виды тестирования

1 **Robot Framework + ATDD/BDD**

Цели занятия:

примерно понимать принципы ATDD и BDD;
знакомство с Robot Framework.

Краткое содержание:

ATDD/BDD как подход;
Robot Framework как инструмент;
уместность применения.

---

2 **Скриншотное тестирование**

Цели занятия:

научиться проводить визуальное сравнение версий приложения с использованием возможностей selenium и библиотеки Pillow.

Краткое содержание:

библиотека Pillow для сравнения изображений;
внедрение скриншотного тестирования в проект.

---

3 **Модульное тестирование**

Цели занятия:

написать модульные тесты, используя библиотеку unittest.

Краткое содержание:

модульное тестирование и научиться писать модульные тесты.

---

4 **Нагрузочное тестирование**

Цели занятия:

написать нагрузочные тесты для веб-приложения.

Краткое содержание:

инструмент Locust для нагрузочного тестирования;
основы нагрузочного тестирования веб-приложений.

8 Проектный модуль

1 Выбор темы и организация проектной работы

Цели занятия:

выбрать и обсудить тему проектной работы;
спланировать работу над проектом;
ознакомиться с регламентом работы над проектом.

Краткое содержание:

правила работы над проектом и специфика проведения итоговой защиты;
требования к результату проекта и итоговой документации.

Домашние задания

1 Проект

Цель: Систематизировать и применить на практике знания полученные на курсе

1. Выбрать приложение для покрытия автотестами.
2. Написать минимум 10 автотестов.
3. Развернуть Jenkins и настроить автоматический запуск.
4. Настроить отчётность по результатам прогона.

---

2 Внедрение автотестов, цели и барьеры

Цели занятия:

определять цели и барьеры на своем пути автоматизации

Краткое содержание:

схемы для анализа целей и барьеров при внедрении автотестов, применение их на практике.

3	**Собеседование Test Automation (Python)**	**Цели занятия:** разбираться в требованиях к кандидатам; **Краткое содержание:** учимся ориентироваться в IT-компаниях; разбираем карьерный путь и в автоматизации и в python; разбираемся с типовыми тестовыми занятиями на вакансию "Test Automation.Python"
4	**Защита проектных работ**	**Цели занятия:** защитить проект и получить рекомендации экспертов. **Краткое содержание:** презентация проектов перед комиссией; вопросы и комментарии по проектам.