**Блок «API»**

*1. Автоматизация запросов средствами Postman*

ПУСТО

*2. Рассказать, каким образом применяется Mock-сервис при тестировании*

ПУСТО

*3. Рассказать, что такое Stub-сервис и его отличие от mock-сервиса*

ПУСТО

**Блок «Selenium»**

*Могу написать конфигурацию для запуска тестов на сервере*

Смотреть файл **Selenoid.docx** в папке «Автоматизация» + репозиторий «**simple-selenoid-test**»

**Блок «Нагрузочное тестирование: Яндекс танк»**

Танк заменил на JMeter. Нужно будет сделать тест-план для показа. Список полезных плагинов есть в файле «**Нагрузочное тестирование**» -> «**JMeter**» -> «**JMeter - плагины.docx**»

**Блок «Pytest»**

*1. Знаю среду и могу запустить тесты*

Тесты запускаются при условии, что имена тестовых функций начинаются на «test\_». Функции можно внести под класс. Тогда класс должен начинаться с ключевого слова «Test…». Например, «class TestUserLogin» и т.п.

Запуск производится командой «pytest»

*2. Знаю параметры запуска (-v, -r, -s, -x, -p и т.д.)*

-v – запуск тестов с бОльшей информативностью

-r chars – показать дополнительную тестовую информацию по пропущенным, зафейленным, ошибочным и остальным тестам

-s – выводит всё (режим –capture=no), в т.ч. и принты на консоль

-x – выйти немедленно при первой ошибке или первом зафейленом тесте

-p – early-load given plugin module name or entry point (multi-allowed) (ранняя загрузка модулей с плагинами или что-то такое)

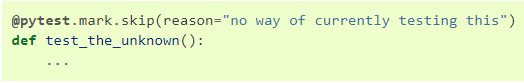
-k «выражение» – запуск тестов, названия которых входит в заданное выражение

*3. Знаю как правильно называть файлы, и как использовать фикстуры для пропуска, фейла тестов*

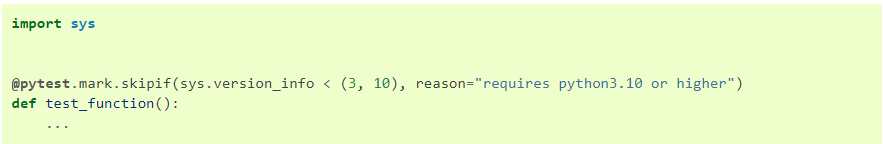
Чтобы Pytest распознал файлы с тестами, их наименования должны начинаться с «test\_». Например, **test\_register\_user.py.**

Декораторы pytest.mark:

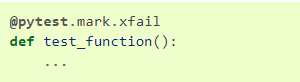
1. pytest.mark.skip – пропуск теста (можно указать причину):



2. pytest.mark.skipif – пропуск при заданном условии:



3. pytest.mark.xfail – пометить тест, который предположительно должен упасть



4. pytest.mark.parametrize – запустить тест с указанными параметрами

**Блок «Python II»**

*1. Async await*

Смотреть репозиторий «**Learning-Python**», проект «**async-await**»

*2. Знаю, что такое основы ООП - инкапсуляция, наследование, полиморфизм*

Смотреть файл «**ООП.docx**»

*3. Рассказать, что такое утиная типизация*

Утиная типизация непосредственно связана с динамической типизацией, которая используется в Python. Слоган утиной типизации: «*Если это выглядит как утка, плавает как утка и крякает как утка, то это, вероятно, и есть утка»*

Конкретный тип или класс объекта не важен, а важны лишь свойства и методы, которыми этот объект обладает. Другими словами, при работе с объектом его тип не проверяется, вместо этого проверяются свойства и методы этого объекта.

*4. Могу создавать классы и объекты классов*

Ответом будут показанные питоновские файлы с демонстрацией ООП

*5. Могу создать иерархию классов и представляю, на что влияют области видимости в классе. Могу объяснить все области видимости на примере*

Смотреть файл «**class\_hierarchy.py**». Там написано древо классов + показаны области видимости атрибутов класса («\_» и «\_\_»)

*6. Могу создать константу класса и рассказать, зачем она нужна*

Хз, что имелось ввиду. Если речь шла об обычных константах, заключённых внутри класса, то они нужны для использования внутри методов класса (походу)

*7. Могу создать статический метод в классе и вызвать его вне класса*

Смотреть файл «**class\_methods.py**».

*8. Паттерн Page Object*

Заключается в распределении страниц сайта по классам. Одна страница – один класс, в котором расположены локаторы конкретной страницы и методы для работы именно с этой страницей. Таким образом мы можем «собирать» наши тесты по кусочкам, используя методы из этих классов

*9. Могу рассказать, как работает приведение типов в Python*

ПУСТО

**Блок «Allure»**

*1. Знаю синтаксис в python для построения отчетов*

*2. Могу запустить отчет на компьютере*