

# Направление «Python-разработчик», KPMG

## Аннотация

Добро пожаловать на виртуальную стажировку **Shift + Enter** от KPMG. Предлагаем тебе познакомиться с ролью начинающего Python-разработчика и стать частью дружной IT-команды.

В твои обязанности как Junior Python developer входит:

- Умение быстро осваивать новый материал и вникать в суть заданий.
- Написание программного кода на языке Python для задач по анализу данных, машинному обучению, парсингу<sup>1</sup> сайтов и автоматизации тестирования.

## Развиваемые компетенции

По результатам выполнения заданий ты научишься:

1. Получать информацию с сайта с помощью собственной программы парсинга данных.
2. Писать юнит-тесты по проверке и выборке данных с сайта, а также по проверке базы данных.
3. Создавать базу данных с использованием SQLite в Python и добавлять в нее данные.

А также прокачаешь следующие скиллы: проектирование баз данных, основы юнит-тестирования, знание Python.

## Описание подзадач

Для программиста важно уметь не только писать код, но и его тестировать. Test Driven Development (TDD), или разработка через тестирование, – техника разработки программного обеспечения (ПО), которая позволяет улучшить покрытие кода тестами и снизить количество ошибок, попадающих в написанную и протестированную версию. В ходе стажировки ты напишешь программу для парсинга данных, поработаешь с проектированием базы данных и юнит-тестированием.

Выполнение всего блока заданий займет у тебя не более 90–115 минут.

## Рекомендуемый тайминг

1. 30–40 минут на первое задание.
2. 30–35 минут на второе задание.
3. 30–40 минут на третье задание.

## Информация о загрузке решения

Этот проект содержит несколько подзадач. Можно загрузить файл, содержащий решение только части заданий, но по возможности постарайся сделать их все.

Желаем удачи!

---

<sup>1</sup>Парсинг — это автоматизированный процесс сбора данных с сайтов.

## Задание 1. Напиши программу для парсинга данных

Утром ты получил письмо от своего руководителя. Он просит тебя подключиться к работе по написанию программы для парсинга.

Привет!

KPMG хочет оценить возможности по проникновению на новые рынки. Мы думаем, что анализ информации портала госзакупок поможет лучше понять условия проведения тендеров. Поэтому тебе нужно научиться парсить сайт госзакупок с использованием языка программирования Python (версии 3 и выше) и дополнительной библиотеки для парсинга сайтов. Некоторые разработчики используют регулярные выражения или встроенные библиотеки Python, но использование специализированных библиотек для парсинга сайтов – более удобный метод извлечения информации.

Чтобы создать заготовку для программы парсинга на Python, прошу тебя:

1. Определить, какие библиотеки Python будут нужны для реализации программы.  
Hints: обязательно прочитай статьи во вкладке «Полезные материалы».
2. Написать программу получения информации с [сайта госзакупок](#), которая считывала бы данные и сохраняла их в удобном для дальнейшей работы формате. Для лучшего понимания структуры HTML-страницы в современных браузерах код можно посмотреть, кликнув правой кнопкой на странице и нажав «Просмотреть исходный код» (или комбинацию клавиш Ctrl + U).  
Код также должен выводить номер закупки и начальную цену (см. вложение 1).  
Hints: для этого используй функцию find и HTML-код из вложения 1.

Оформи свое решение в виде папки с файлами. Не забудь приложить README-файл с описанием того, что содержится в каждом файле. Пожалуйста, пришли мне архив в ближайшие полтора часа.

До связи!

### Полезные материалы

- [Статья](#) про Web Scraping с помощью Python.
- [Статья](#) о парсинге сайтов.

### Формат конечного результата

Папка, содержащая файлы, включающие файл Python с кодом, файл README в .txt или .docx с пояснениями к коду.

### Форма загрузки результата

Пожалуйста, загрузи свой вариант ответа в формате zip-архива, используя инструмент «Загрузить решение». Необходимо сформировать единый zip-архив, содержащий решение одного или всех заданий по выбранной специальности.

### Пример решения

У тебя будет возможность ознакомиться с примером решения задания после отправки своей версии.

## Вложение 1. Дополнительная информация по заданию 1

Пример того, как выглядит карточка с информацией на портале госзакупок

223-ФЗ Закупка у единственного поставщика (подрядчика, исполнителя)		Начальная цена	
№ 32110886908    Работа комиссии		872 432,50 ₽	
Объект закупки			
мебель			
Заказчик			
ГОСУДАРСТВЕННОЕ УНИТАРНОЕ ПРЕДПРИЯТИЕ САХАЛИНСКОЙ ОБЛАСТИ		Размещено	Обновлено
"УГЛЕГОРСКОЕ ДОРОЖНОЕ РЕМОНТНО - СТРОИТЕЛЬНОЕ УПРАВЛЕНИЕ"		30.11.2021	01.12.2021

### HTML-код карточки

- Карточка  

- Номер карточки  

- Начальная цена

### Примечание

Перед тем как считывать данные с сайта, важно проверить статус HTTP-кода ответа сервера. В случае если статус не равен 200 (именно статус 200 информирует пользователя об успешной обработке запроса), вывести сообщение «Status code is not equal 200 – problem in loading site».

## Задание 2. Напиши юнит-тест для проверки своей программы

Чуть позже ты получил письмо от руководителя с новым заданием. Подход TDD (разработка через тестирование) хорошо подходит для юнит-тестирования. Так как твой код еще не покрыт тестами, руководитель отказывается принимать твою задачу. Возможно, в твоём решении есть баги, и оно не полностью работоспособно, поэтому твоя задача — написать юнит-тест, который протестирует алгоритм работы программы парсинга.

Привет!

Спасибо за твое решение! Чтобы мы с коллегами смогли взять твой код в работу, тебе нужно написать юнит-тест на Python для тестирования своей программы.

Юнит-тест представляет собой программу, проверяющую работу небольшой части кода. Разработчики регулярно обновляют ПО и вносят правки, поэтому существуют разные типы тестов. Представь пирамиду с основанием в виде юнит-тестов, далее следует слой с интеграционными тестами, затем — системные тесты, и на вершине пирамиды находятся end-to-end тесты. Обычно при разработке программы программист пишет юнит-тесты, а далее тестировщик пишет все остальные виды тестов. Но без юнит-тестов невозможен в целом механизм полного тестирования приложения: написание этих тестов является основой для других тестов, без них невозможно 100%-е покрытие кода тестами.

Юнит-тесты могут быть написаны с нуля без использования библиотек, однако проще воспользоваться уже готовыми фреймворками. В Python их несколько, но я предлагаю поработать с `unittest`. Шаблон структуры юнит-теста дан во вложении 2.

Тебе нужно:

1. Как и в предыдущем задании, считать данные с сайта госзакупок и сохранить их в удобном формате.
2. Провести проверку на успешность с помощью функции `assertGreater`, чтобы выбрать все ненулевые данные.
3. Провести еще одну проверку на успешность, используя `assertIsNotNone` для аргументов с номером карточки и ценой.

Оформи свое решение в виде файла Python и пришли мне его через час. Спасибо!

### Полезные материалы

- [Статья](#) о модуле `unittest` с примерами использования различных проверок.

### Формат конечного результата

Файл в формате `.py/.ipynb` (при необходимости можешь оставить комментарии в нем же).

### Форма загрузки результата

Пожалуйста, загрузи свой вариант ответа в формате zip-архива, используя инструмент «Загрузить решение». Необходимо сформировать единый



zip-архив, содержащий решение одного или всех заданий по выбранной специальности.

### Пример решения

У тебя будет возможность ознакомиться с примером решения задания после отправки своей версии.

## Вложение 2. Шаблон кода для юнит-теста

```
import unittest

class TestSomething(unittest.TestCase):
    def test_thing(self):
        assert True # здесь нужны проверки assertEquals для статуса ответа сервера и
        assertGreater для проверки, что номер карточки и начальная цена являются не пустыми
        значениями

if __name__ == '__main__':
    unittest.main()
```

## Задание 3. Создай базу данных по закупкам и напиши юнит-тест по ее проверке

Чуть позже ты получил сообщение от руководителя в мессенджере с постановкой финального на сегодня задания.

Привет!

Спасибо за отлично проделанную работу. Запуск файла с юнит-тестом для программы парсинга прошел успешно и без ошибок. Теперь тебе осталось сделать всего одну задачу – спроектировать базу данных по закупкам и проверить ее работу при помощи юнит-теста.

Первым этапом проектирования БД станет создание таблицы `cards`. Обычно для создания БД SQLite в Python используется объект `Connection`, который создается с помощью функции `connect` библиотеки `sqlite3`. Далее нужно создать саму таблицу `cards`, содержащую 2 колонки – `number` и `amount`, в которых находятся номер закупки и начальная цена объекта. Для твоего удобства интересующие нас значения уже импортированы с портала госзакупок в виде таблицы (см. вложение 3).

Hints: для добавления данных нужно использовать объект `cursor` и функцию `executemany` для выборки нескольких значений из БД. Подробно об этом написано в руководстве по SQLite, ссылка на которое дана во вкладке «Полезные материалы». Для выборки данных из БД не забудь использовать `fetchall`, а также проверить, что данные действительно vybrались с помощью команды `print`.

После этого напиши в отдельном Python-файле юнит-тест для проверки работы БД, как ты делал это для программы парсинга. Здесь важно написать запрос для выбора первых трех строк, а также проверки на успешность, связанные с проверкой того, что значения заполнены и не пустые (`assertIsNot(x, 0)` и `assertGreater(x, 0)`).

Как обычно, оформи свое решение в виде папки с файлами (Python-файлы и README).

Жду твое решение, спасибо за помощь!

### Полезные материалы

- [Руководство](#) по SQLite в Python.

### Формат конечного результата

Папка, содержащая файлы, включающие файл Python с кодом, файл README в `.txt` или `.docx` с пояснениями к коду.

### Форма загрузки результата

Пожалуйста, загрузи свой вариант ответа в формате zip-архива, используя инструмент «Загрузить решение». Необходимо сформировать единый zip-архив, содержащий решение одного или всех заданий по выбранной специальности.



### Пример решения

У тебя будет возможность ознакомиться с примером решения задания после отправки своей версии.

### Вложение 3. Значения для таблицы cards

Number (номер закупки) – тип данных integer	Amount (цена объекта в рублях) – тип данных double
0338100000521000100	448400.00
0338100000521000099	80500.00
0161100008221000004	35000.00
0347000000121000054	125051.45
32110590678	21109928.54
32110750824	5629302.00
0347000000121000053	144913.80
0161100008221000003	580000.00