Язык Python. Часть 2

Лекция 6

Пример json-ресурса

- Социальная сеть reddit.com
- Есть темы (subreddit): https://www.reddit.com/r/Python/
- Их можно получить как json
- https://www.reddit.com/r/Python/.json

Читаем json-документ

- Встроенный модуль
- Так и называется - json
- Ключевая функция для чтения - loads
- При успешном чтении возвращает объект
- Который который является словарем
- Некоторые значения которого - тоже словари

Скачаем и прочитаем json

```
1 import requests
2 import json
3
4 response = requests.get('https://www.reddit.com/r/Python/.js
5 data = json.loads(response.text)
6
7 print(data)
```

Возвращаемся в requests

- Мы разбирали GET-запрос
- Он нацелен на то, чтобы получить информацию
- Есть два типа запросов для отправки данных на сервер
- POST и PUT
- Они немного для разных ситуаций

Запрос POST

- Для ситуаций, когда повторное исполнение того же запроса
- Изменяет состояние по сравнению с однократным
- Пример: отправка нового сообщения в дискуссию
- Два раза исполнили - появилось два сообщения

Запрос POST

- Для ситуаций, когда повторное исполнение того же запроса
- НЕ изменяет состояние по сравнению с однократным
- Пример: исправление сообщения в дискуссии по его id
- Два раза исполнили - все равно, что один

Но это все зависит от разработчиков ресурса

- Иногда бывают неочевидные решения
- В REST API hh.ru добавление резюме - GET-запрос
- Что выглядит странным
- А исправление - PUT
- Что совершенно логично

Оформление в requests

- Соответствующие функции - put и post
- Нужно дополнительное действие - заполнить тело запроса
- Именованный параметр data
- Передаем словарь
- Такого же формата, как читаем из json.loads

Пример REST API

- hh.ru
- Много разных запросов
- Есть анонимные
- Есть с регистрацией, но бесплатные
- Есть платные - в основном, про резюме

Аутентификация

- Часто на интернет-ресурсах требуют логины и пароли
- Как минимум, их надо иметь
- А для автоматического скачивания - еще и знать, что с ними делать
- Чтобы они помогли

Принципы веб-аутентификации

- В http-протоколе нет понятия состояния
- Все запросы независимы друг от друга
- А аутентификация предполагает нечто противоположное
- Мы один раз предъявляем логин-пароль
- А в последующих запросах сервер должен понять, что наш пароль он уже знает

Принципы веб-аутентификации

- Теоретически - можно в каждом запросе передавать логин и пароль
- Но так никто не делает
- Потому что небезопасно
- Вместо этого выдается большое случайное число
- И клиент его передает в одном из HTTP-заголовков
- И периодически оно меняется

Аутентификация в requests

- Можно разобраться в деталях
- И выполнять нужные действия на стороне клиента самостоятельно
- requests это берет на себя
- Но есть нюансы

Аутентификация в requests

- Вариантов реализации указанных принципов м
- Не все автоматически поддерживаются в reques
- Но есть механизм расширения
- И есть плагины
- В любом случае надо понять, какой метод аутентификации использует сервер
- https://requests.readthedocs.io/en/latest/user/authe

Хранение и обработка данных

- То, что накачали - хорошо бы где-то хранить
- И уметь извлекать
- Существуют базы данных
- Можно хранить json в текстовых файлах
- Для базы данных нужна СУБД, json иерархичен, иногда - слишком

CSV

- Облегченный формат хранения данных
- В виде текстового файла
- Слегка отформатированного под таблицу
- Можно читать глазами
- Удобно обрабатывать автоматически

CSV

- Предполагается, что данные состоят из строчек
- А строчки/записи - из полей
- В каждой записи - фиксированное число полей
- Поля разделяются каким-то символом
- Например, запятой
- Отсюда и название - Comma Separated Values

CSV

- Иногда используют другие разделители
- Часто - табуляцию
- Потому что поле может быть текстом
- И там могут быть запятые как часть текста
- Это решаемая проблема, но доставляет неудобства
- А табуляция гораздо реже бывает частью текста

CSV

- В Python есть стандартный модуль для работы с csv
- Но там довольно низкоуровнево
- И есть сторонняя библиотека pandas
- Она в целом умеет работать с данными
- И в частности - с csv

Pandas

- https://pandas.pydata.org/
- https://pypi.org/project/pandas
- `python3 -m pip install pandas`
- Ключевая структура - DataFrame
- Что-то типа двумерной таблички
- Ее можно заполнить и сохранить в csv

Pandas

- В Python можно создать список
- Элементами которого будут одномерные списки одинаковой длины
- И тогда получится своего рода двумерная табличка
- Что в такой структуре неудобного
  - Уязвима к ошибкам
  - Нет гарантий однотипности значений в колонкам

- Неудобно добавлять строки
- Безымянные колонки, доступны только по номеру

Pandas

- Сравним с DataFrame
  - Следит на "прямоугольностью"
  - Можно указать тип данных для колонок
  - Удобные манипуляции в обоих измерениях
  - Именованные колонки

Пример создания

```
df = pd.DataFrame(
        {
            "Name": [
                "Braund, Mr. Owen Harris",
                "Allen, Mr. William Henry",
                "Bonnell, Miss. Elizabeth",
            ],
            "Age": [22, 35, 58],
            "Sex": ["male", "male", "female"],
        }
)
```