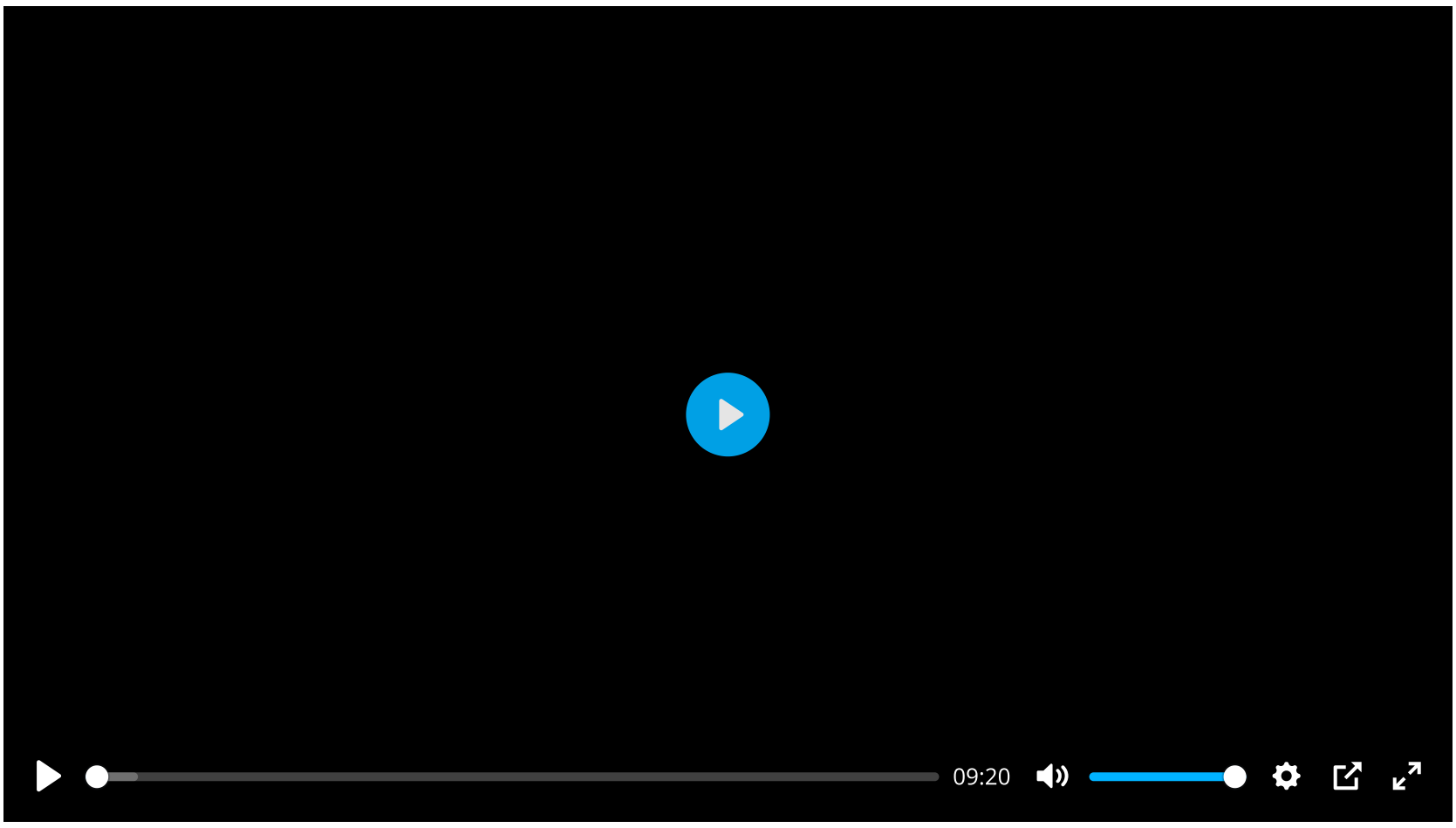

Текстовая расшифровка видео:

САМЫЕ ПОПУЛЯРНЫЕ КЛИЕНТЫ ДЛЯ HTTP

План:

- Urllib;
- Requests;
- HTTPX;
- Игра престолов

Urllib

Urllib – встроенный модуль, на который вы наткнетесь при поиске того, как сделать HTTP-запрос.

Выглядит он следующим образом:

```
1. import urllib.parse
2. import urllib.request
3.
4. # URL - адрес, получающий данные
5. url = 'http://www.someserver.com/cgi-bin/register.cgi'
6. # Отправляемые данные
7. values = {'name' : 'Michael Foord',
8.          'location' : 'Northampton',
9.          'language' : 'Python' }
10.
11. # Кодирование данных
12. data = urllib.parse.urlencode(values)
13. # данные должны быть байтами
14. data = data.encode('utf-8')
15. req = urllib.request.Request(url, data)
16. with urllib.request.urlopen(req) as response:
17.  the_page = response.read()
```

Документация по модулю

Что делаем: задаем URL, создаем запрос, делаем реквест и отправляем его через специальный метод.

В стандартной библиотеке модули сильно тормозятся в дальнейшей разработке. Urllib не рекомендуют использовать в реальной продуктовой разработке. Это типичный пример того, как встроенная библиотека сама себе рекомендует использовать внешнюю.

Вывод: если необходимо делать HTTP-запрос и у вас есть только Python и нет возможности поставить внешние зависимости, то Urllib поможет. В остальных случаях следует отдать предпочтения другим вариантам.

Requests

Requests – простой модуль, который получил популярность путем реализации логики HTTP-запроса в простом виде.

Рассмотрим пример:

```
import requests

# Search GitHub's repositories for requests
response = requests.get(
   'https://api.github.com/search/repositories',
   params={'q': 'requests+language:python'},
)

# Inspect some attributes of the `requests` repository
json_response = response.json()
repository = json_response['items'][0]
print(f'Repository name: {repository["name"]}')
print(f'Repository description: {repository["description"]}')
```

Что делаем: мы пишем «requests.get», далее, перекодируем автоматически его в Python-объект. «.json» возвращает Python-контейнеры. Также, здесь есть механизмы для выкачивания файлов.

Нюанс: библиотека Requests написана поверх библиотеки urllib3. Библиотека во многом зависит от реализации urllib3.

HTTPX

У HTTPX и Requests практически идентичный API. Однако HTTPX немного новее и быстрее, чем предыдущий пример:

```
import httpx

# Search GitHub's repositories for httpx
response = httpx.get(
    'https://api.github.com/search/repositories',
   params={'q': 'httpx+language:python'},
)

# Inspect some attributes of the first repository
json_response = response.json()
repository = json_response['items'][0]
print(f'Repository name: {repository["name"]}')
print(f'Repository description: {repository["description"]}')
```

Игра престолов

HTTPX поддерживает HTTP/2. В requests его поддержка завязана на пакет urllib3, в котором его пока отказались завозить. Также можно использовать низкоуровневый pycurl, он быстрее, но сложнее в использовании.

HTTPX не ходит по редиректам автоматически.

Уникальность HTTPX заключается в поддержке синхронного и асинхронного интерфейса в одной библиотеке.

Код, в случае HTTPX выглядит следующим образом:

```
1. import asyncio
2. import httpx
3.
4. async def main():
5.   async with httpx.AsyncClient() as client:
6.       response = await client.get('https://www.example.com/')
7.       print(response)
8.
9. asyncio.run(main())
```

В мире асинхронных функций мы используем асинхронную версию клиента «httpx.AsyncClient» через «async with» и в нем через «await» вытягиваем содержимое по URL.

Как вам урок? 😠 😐 🔥

Далее >