Библиотека requests

In [ ]:

```
import requests
```

In [ ]:

```
# метод get
res = requests.get('https://yandex.ru')
res
# res.status_code
```

In [ ]:

```
# браузер отрисовал бы страницу на основе данного текста
res.text
```

In [ ]:

```
# посмотрим куки
res.cookies
```

In [ ]:

```
# получаем плохой статус
bad_request = requests.get('https://yandex.ru/secret')
bad_request
```

In [ ]:

```
bad_request.text
```

In [ ]:

```
# попроуем сделать post запрос
post_req = requests.post('https://yandex.ru')
post_req
```

In [ ]:

```
# сформируем поисковый запрос, обратите внимание на его формат
URL = 'https://yandex.ru/search/?text=python&lr=54'
```

In [ ]:

```
req = requests.get(URL)
req
```

In [ ]:

```
req.text
```

In [ ]:

```
# в request можно передать параметры запроса и заголовки (headers) в виде словарей.
# сегодня не будем рассматривать примеры с необхожимостью передачи заголовка,
# но в практике вам это точно понадобится
URL = 'https://yandex.ru/search/'
params = {
    'text': 'python',
    'lr': 54
}
headers = {}
req = requests.get(URL, params)
req = requests.post(URL, params, headers)
req.text
```

In [ ]:

```

```

In [ ]:

```
# очень часто сайты могут ограничивать частые запросы к себе,
# поэтому нужно задерживать исполнение
import time
time.sleep(0.2)
```

Beautiful Soup

In [ ]:

```
# как разбирать всю эту разметку? Поможет BeautifulSoup.
from bs4 import BeautifulSoup
```

Практика 1. Напишем скрипт, который будет отбирать посты из нужных хабов на habr.com

In [ ]:

```
# определяем список хабов, которые нам интересны
DESIRED_HUBS = ['python', 'bigdata']
```

In [ ]:

```
# получаем страницу с самыми свежими постами
req = requests.get('https://habr.com/ru/all/')
soup = BeautifulSoup(req.text, 'html.parser')
```

In [ ]:

```
# извлекаем посты
posts = soup.find_all('article', class_='post')
posts
```

In [ ]:

```
for post in posts:
    post_id = post.parent.attrs.get('id')
   # если идентификатор не найден, это что-то странное, пропускаем
    if not post_id:
        continue
    post_id = int(post_id.split('_')[-1])
    print('post', post_id)
    hubs = post.find_all('a', class_='hub-link')
```

In [ ]:

```
# добавляем извлечение хабов из постов, чтобы отбирать только нужные
for post in posts:
    post_id = post.parent.attrs.get('id')
   # если идентификатор не найден, это что-то странное, пропускаем
    if not post_id:
        continue
    post_id = int(post_id.split('_')[-1])
    hubs = post.find_all('a', class_='hub-link')
    for hub in hubs:
            hub_lower = hub.text.lower()
           # ищем вхождение хотя бы одного желаемого хаба
            if any([hub_lower in desired for desired in DESIRED_HUBS]):
               # пост нам интересен - делаем с ним все что захотим:
               # можно отправит в телеграм уведомление, можно на почту и т.п.
                title_element = post.find('a', class_='post__title_link')
                print(title_element.text, title_element.attrs.get('href'))
               # так как пост уже нам подошел - дальше нет смысла проверять хабы
                break
```

Практика 2. Напишем скрипт, который будет собирать новости с сайта Коммерсанта

In [ ]:

```
URL = 'https://www.kommersant.ru/search/results'
params = {
    'search_query': 'python'
}
```

In [ ]:

```
res = requests.get(URL, params)
```

In [ ]:

```
res.text
```

In [ ]:

```
soup = BeautifulSoup(res.text, 'html.parser')
soup
```

In [ ]:

```
# добираемся до блоков с новостями
news_blocks = soup.find_all('div', class_='search_results_item')
news_blocks
```

In [ ]:

```
# добираемся до текста со ссылкой
articles_intro = list(map(lambda x: x.find('div', class_='article_intro'), news_blocks))
articles_intro
```

In [ ]:

```
# добираемся до ссылок
a_list = list(map(lambda x: x.find('a').get('href'), articles_intro))
a_list
```

In [ ]:

```
# формируем полноценные ссылки
all_refs = list(map(lambda x: 'https://www.kommersant.ru/' + x, a_list))
all_refs
```

In [ ]:

```
# объединим все в одну функцию
def get_all_links(url, query):
    all_refs = []
    params = {
        'search_query': query,
    }
    res = requests.get(URL, params)
    time.sleep(0.3)
    soup = BeautifulSoup(res.text, 'html.parser')
    news_blocks = soup.find_all('div', class_='search_results_item')
    articles_intro = list(map(lambda x: x.find('div', class_='article_intro'), news_blocks))
    a_list = list(map(lambda x: x.find('a').get('href'), articles_intro))
    all_refs = list(map(lambda x: 'https://www.kommersant.ru/' + x, a_list))

    return all_refs

all_links = get_all_links(URL, 'python')
all_links
```

In [ ]:

```
# но мы же собрали только одну страницу? Хотим ВСЕ новости
def get_all_links(url, query, pages):
    all_refs = []
    params = {
        'search_query': query
    }
    for i in range(pages):
        params['page'] = i + 1
        res = requests.get(URL, params)
        time.sleep(0.3)
        soup = BeautifulSoup(res.text, 'html.parser')
        news_blocks = soup.find_all('div', class_='search_results_item')
        articles_intro = list(map(lambda x: x.find('div', class_='article_intro'), news_blo
        a_list = list(map(lambda x: x.find('a').get('href'), articles_intro))
        all_refs += list(map(lambda x: 'https://www.kommersant.ru/' + x, a_list))
    return all_refs

all_links = get_all_links(URL, 'python', 3)
all_links
```

In [ ]:

```
import pandas as pd
```

In [ ]:

```
# собираем даты, заголовки и тексты новостей
# получаем ошибочку. Значит не у всех получаемых страниц одинаковая разметка
for link in all_links:
    soup = BeautifulSoup(requests.get(link).text, 'html.parser')
    time.sleep(0.3)
    date = pd.to_datetime(soup.find('time', class_='title__cake').get('datetime'), dayfirst
    print(date)
    title = soup.find('h1', class_='article_name').text
    print(title)
    text = soup.find('div', class_='article_text_wrapper').text
    print(text)
```

In [ ]:

```
# получаем ошибочку. Значит не у всех получаемых страниц одинаковая разметка
for link in all_links:
    soup = BeautifulSoup(requests.get(link).text, 'html.parser')
    if soup.find('div', class_='b-article__publish_date'):
        date = pd.to_datetime(soup.find('div', class_='b-article__publish_date').find('time
    elif soup.find('time', class_='title__cake'):
        date = pd.to_datetime(soup.find('time', class_='title__cake').get('datetime'), dayf
    print(date)
    if soup.find('h2', class_='article_name'):
        title = soup.find('h2', class_='article_name').text
    else:
        title = soup.find('h1', class_='article_name').text
    print(title)
    text = soup.find('div', class_='article_text_wrapper').text
    print(text)
```

In [ ]:

```
# запишем данные в датафрейм
kom_news = pd.DataFrame()
for link in all_links:
    soup = BeautifulSoup(requests.get(link).text, 'html.parser')
    time.sleep(0.3)
    if soup.find('div', class_='b-article__publish_date'):
        date = pd.to_datetime(soup.find('div', class_='b-article__publish_date').find('time
    elif soup.find('time', class_='title__cake'):
        date = pd.to_datetime(soup.find('time', class_='title__cake').get('datetime'), dayf
    if soup.find('h2', class_='article_name'):
        title = soup.find('h2', class_='article_name').text
    else:
        title = soup.find('h1', class_='article_name').text
    text = soup.find('div', class_='article_text_wrapper').text
    row = {'date': date, 'title': title, 'text': text}
    kom_news = pd.concat([kom_news, pd.DataFrame([row])])
kom_news
```

In [ ]:

```
# обернем в функцию
def get_kom_news(links):
    kom_news = pd.DataFrame()
    for link in all_links:
        soup = BeautifulSoup(requests.get(link).text, 'html.parser')
        if soup.find('div', class_='b-article__publish_date'):
            date = pd.to_datetime(soup.find('div', class_='b-article__publish_date').find('
        elif soup.find('time', class_='title__cake'):
            date = pd.to_datetime(soup.find('time', class_='title__cake').get('datetime'),
        if soup.find('h2', class_='article_name'):
            title = soup.find('h2', class_='article_name').text
        else:
            title = soup.find('h1', class_='article_name').text
        text = soup.find('div', class_='article_text_wrapper').text
        row = {'date': date, 'title': title, 'text': text}
        kom_news = pd.concat([kom_news, pd.DataFrame([row])])
    return kom_news
```

In [ ]:

```
kom_news = get_kom_news(all_links)
kom_news
```

API

Практика 3. Получим данные о песнях исполнителя при помощи API ITunes (https://developer.apple.com/library/archive/documentation/AudioV

In [ ]:

```
# https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/iTuneSear
URL = 'https://itunes.apple.com/search?term=jack+johnson'
```

In [ ]:

```
res = requests.get(URL)
```

In [ ]:

```
res.text
```

In [ ]:

```
res.json()
```

In [ ]:

```
params = {
    'term': 'шнуров',
    'limit': 200,
#     'offset': 2
}
```

In [ ]:

```
# pd.set_option('display.max_columns', 100)
res = requests.get(URL, params)
# res.json()

pd.DataFrame(res.json()['results'])
```

In [ ]:

```
params = {
    'term': 'лазарев',
    'limit': 60,
    'attribute': 'allArtistTerm',
    'country': 'ru'
}
```

In [ ]:

```
res = requests.get(URL, params)
res.json()

pd.DataFrame(res.json()['results'])
```

Практика 4. Соберем сообщения из новостной ленты ВК по нужному запросу

In [ ]:

```
# https://vk.com/dev/manuals
# https://vk.com/dev/newsfeed.search
NEWSFEED_REQUEST = 'https://api.vk.com/method/newsfeed.search?'
TOKEN = '9df7991c9df7991c9df7991c329d86910d99df79df7991cc363a27748dcf7ad91284ef6'
VERSION = '5.103'
SLEEP = 0.33
```

In [ ]:

```
# обращаем внимание, что максимальное количество постов,
# которые можно вытащить за раз, ограничено
params = {
    'access_token': TOKEN,
    'v': VERSION,
    'q': 'короновирус',
    'count': 200
}
```

In [ ]:

```
res = requests.get(NEWSFEED_REQUEST, params)
res
```

In [ ]:

```
res.text
```

In [ ]:

```
res.json()
```

In [ ]:

```
res.json()['response']['items']
```

In [ ]:

```
pd.DataFrame(res.json()['response']['items'])
```

In [ ]:

```
# соберем все доступные сообщения по запросу
newsfeed_df = pd.DataFrame()
while True:
    result = requests.get(NEWSFEED_REQUEST, params)
    time.sleep(0.33)
    newsfeed_df = pd.concat([newsfeed_df, pd.DataFrame(result.json()['response']['items'])]
    if 'next_from' in result.json()['response'].keys():
        params['start_from'] = result.json()['response']['next_from']
    else:
        break
newsfeed_df
```

In [ ]:

```

```