

# РАБОТА С THREADPOOLEXECUTOR

## ВВЕДЕНИЕ

### Задачи урока

- 01** Научиться работать с ThreadPoolExecutor для загрузки данных с сайта в нескольких потоках
- 02** Использовать асинхронные функции для эффективной обработки данных

...

```
import concurrent.futures
import asyncio
import pandas as pd
import numpy as np
import time
import requests
import datetime

def start():
    with concurrent.futures.ThreadPoolExecutor(max_workers=30) as pool:
        pool.map(export_bathc, np.array_split(range(0, 100000), 1000))

def export_bathc(categorys):
    asyncio.set_event_loop(asyncio.new_event_loop())
    return asyncio.get_event_loop().run_until_complete(asyncio.gather(*[
        asyncio.ensure_future(export_data(category))
        for category in categorys
    ]))

async def export_data(category):
    for skip in range(0, 99999, 100):
        if download_data(category, skip) == False: break

    return None

def download_data(category, skip, retry_count = 5):
    try:
```

```
req = f"""https://analitika.woysa.club/images/panel/json/download/niches.php?skip={skip}&price_min=0&price_max=1060225&up_vy_min=0&up_vy_max=108682515&up_vy_pr_min=0&up_vy_pr_max=2900&sum_min=1000&sum_max=82432725&feedbacks_min=0&feedbacks_max=32767&trend=false&sort=sum_sale&sort_dir=-1&id_cat={category}"""

data = requests.get(req)
df = pd.read_excel(data.content)
if df.empty: return False
df['category_id'], df['download_date'] = category,
datetime.datetime.now()

print(df)
except Exception as e:
    if retry_count == 0: return True
    time.sleep(10)
    download_data(category, skip, retry_count - 1)

if __name__ == '__main__':
    start()
```

## В примере мы реализуем загрузку данных с сайта в разных потоках

- 01 Массив категорий с 0 до 100000 делим на части по 1000 элементов
- 02 Передаем эти подмассивы в метод `export_bathc`, где уже асинхронно будем получать данные по каждой категории

```
...
def export_bathc(categorys):
    asyncio.set_event_loop(asyncio.new_event_loop())
    return
asyncio.get_event_loop().run_until_complete(asyncio.gather(*[
    asyncio.ensure_future(export_data(category))
    for category in categorys
]))
```

- 03 Так как в новом потоке нет `event loop`, то мы его задаем `asyncio.set_event_loop(asyncio.new_event_loop())`

04 Далее, выполняем асинхронно метод `export_data` по каждой категории

```
...
asyncio.get_event_loop().run_until_complete(asyncio.gather(*[
    asyncio.ensure_future(export_data(category))
    for category in categories
]))
```

05 В методе `export_data` мы проходимся по каждой странице ответа, пока они не будут пустыми, так как в каждой категории может быть несколько страниц с ответами. Таким образом, в метод `download_data` мы передаем категорию и сдвиг по которому мы хотим получить данные

```
...
async def export_data(category):
    for skip in range(0, 99999, 100):
        if download_data(category, skip) == False: break

    return None
```

06 И наконец, `download_data`, которая получает данные и печатает их на экране

```
...
def download_data(category, skip, retry_count = 5):
    try:
        req = f"""https://analitika.woysa.club/images/panel/json/download/niches.php?skip={skip}
&price_min=0&price_max=1060225&up_vy_min=0&up_vy_max=108682515&up_vy_pr_min=0&up_vy_pr_max=2900&sum_min=1000&sum_max=82432725&feedbacks_min=0&feedbacks_max=32767&trend=false&sort=sum_sale&sort_dir=-1&id_cat={category}"""
        data = requests.get(req)
        df = pd.read_excel(data.content)
        if df.empty: return False
        df[['category_id', 'download_date']] = category, datetime.datetime.now()

        print(df)
    except Exception as e:
        if retry_count == 0: return True
        time.sleep(10)
        download_data(category, skip, retry_count - 1)
```

## ИТОГИ

- ✓ Реализована загрузка данных с сайта в разных потоках с использованием `ThreadPoolExecutor`
- ✓ В методе `export_data` происходит обход каждой страницы ответа, пока они не будут пустыми для каждой категории
- ✓ Для обработки и вывода данных использовались библиотеки `asyncio, pandas, numpy, time, requests и datetime`
- ✓ Работа с потоками позволяет эффективно обрабатывать большие объемы данных

