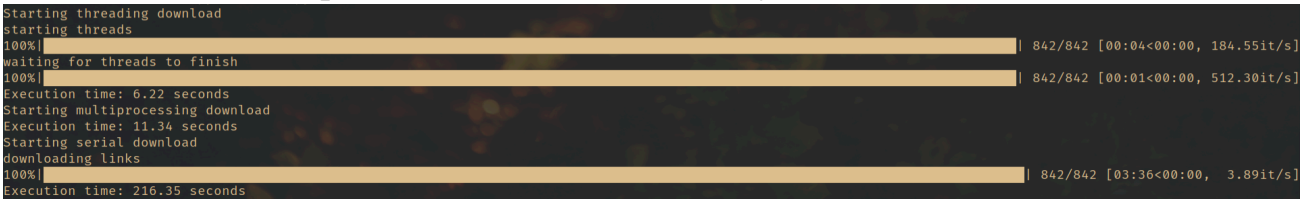
«Параллельные и распределённые вычисления» Лабораторная работа №7 Многопоточность в Python	Выполнил:	Зенин М.А.
	Преподаватель:	Катаев А.В.
	Дата выполнения:	
	Подпись преподавателя	

Цель: Изучение работы с параллельными вычислениями в языке Python.

Задача: Реализовать алгоритм на Python, продемонстрировать увеличение производительности в за счет распараллеливания вычислений, сравнить работу с threading и multitasking (не надо дублировать код алгоритма вычислений, используйте код повторно).

Скачивание статей гражданского кодекса с консультант плюс:



Threading: 6 сек
Multiprocessing: 11 сек
Последовательно: 216 сек

Количество процессов multiprocessing ограничено до 500. Иначе слишком много записей в файлы одновременно.

Код:

```
import requests
from bs4 import BeautifulSoup
from tqdm import tqdm
import threading
from multiprocessing import Process, Pool
import time

def extract_links(url):
        response = requests.get(url)

        if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')

        links = [a['href'] for a in soup.find_all('a', href=True) if not
a['href'].startswith(('http://', 'https://', '//www'))]

        return links
```

```
        else:
        print(f"Failed to retrieve the page. Status code: {response.status_code}")
        return []

def download_link(url):
        headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'
        }
        prefix = "https://www.consultant.ru"
        response = requests.get(prefix + url, headers=headers)

        if response.status_code == 200:
        with open(f"files/{url.replace("/", "-")}.html", "w") as file:
        file.write(response.text)
        else:
        print(url)
        print(f"Failed to retrieve the page. Status code: {response.status_code}")



def serial_download(links):
        print("downloading links")
        for link in tqdm(links):
        download_link(link)

def threading_download(links):
        threads = []

        print("starting threads")
        for link in tqdm(links):
        thread = threading.Thread(target=download_link, args=(link,))
        threads.append(thread)
        thread.start()

        print("waiting for threads to finish")
        for thread in tqdm(threads):
        thread.join()

def multiprocessing_download(links):
        with Pool(processes=500) as pool:
```

```
        pool.map(download_link, links)

url = 'https://www.consultant.ru/document/cons_doc_LAW_5142/'
links = extract_links(url)

print("Starting threading download")
start = time.time()
threading_download(links)
end = time.time()
took = end - start
print(f"Execution time: {took:.2f} seconds")

print("Starting multiprocessing download")
start = time.time()
multiprocessing_download(links)
end = time.time()
took = end - start
print(f"Execution time: {took:.2f} seconds")

print("Starting serial download")
start = time.time()
serial_download(links)
end = time.time()
took = end - start
print(f"Execution time: {took:.2f} seconds")
```