«Параллельные и	Выполнил:	Зенин М.А.
распределённые	Преподаватель:	Катаев А.В.
вычисления»	Дата выполнения:	
	Подпись	
Лабораторная работа №7	преподавателя	
Многопоточность в Python	1	

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.

```
File "/home/zen/homework/parallel/lab7/main.py", line 56, in multiprocessing_download
   with Pool(processes=800) as pool:
    return Pool(processes, initializer, initargs, maxtasksperchild,
                context=self.get_context())
 File "/usr/lib/python3.13/multiprocessing/pool.py", line 215, in __init__
   self._repopulate_pool()
   return self._repopulate_pool_static(self._ctx, self.Process,
                                        self._processes,
   ...<3 lines>...
                                        self._maxtasksperchild,
                                        self._wrap_exception)
 File "/usr/lib/python3.13/multiprocessing/pool.py", line 329, in _repopulate_pool_static
   w.start()
 File "/usr/lib/python3.13/multiprocessing/process.py", line 121, in start
   self._popen = self._Popen(self)
 File "/usr/lib/python3.13/multiprocessing/context.py", line 282, in _Popen
   return Popen(process_obj)
   self._launch(process_obj)
 File "/usr/lib/python3.13/multiprocessing/popen_fork.py", line 66, in _launch
   child_r, parent_w = os.pipe()
OSError: [Errno 24] Too many open files
```

## 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")