In [11]:
```
import multiprocessing as mp
import numpy as np
import csv
```

Параллельные вычисления

Материалы:

- Макрушин С.В. Лекция 10: Параллельные вычисления
- https://docs.python.org/3/library/multiprocessing.html (https://docs.python.org/3/library/multiprocessing.html)

Задачи для совместного разбора

1. Посчитайте, сколько раз встречается каждый из символов (заглавные и строчные символы не различаются) в файле `Dostoevskiy Fedor. Prestuplenie i nakazanie - BooksCafe.Net.txt` и в файле `Dostoevskiy Fedor. Igrok - BooksCafe.Net.txt` .

In [1]:
```
def count_char_in_file(filename):
    char_dict = {}

    with open("10_multiprocessing_data/" + filename) as book:
        for char in book.read().lower():
            if char in char_dict.keys():
                char_dict[char] += 1
            else:
                char_dict[char] = 1

    return char_dict


def main(files):
    result = {}
    for file in files:
        result[file] = count_char_in_file(file)
    return result
```

In [2]:
```
files = [
    "Dostoevskiy Fedor. Prestuplenie i nakazanie - BooksCafe.Net.txt",
    "Dostoevskiy Fedor. Igrok - BooksCafe.Net.txt"
    ]

main(files)
```

Out[2]:
```
{'Dostoevskiy Fedor. Prestuplenie i nakazanie - BooksCafe.Net.txt': {'с': 50084,
  'п': 25652,
  'а': 73555,
  'и': 62030,
  'б': 16016,
  'о': 106740,
  ',': 26973,
  ' ': 182305,
  'ч': 16492,
  'т': 59813,
  'к': 30802,
  'л': 42328,
  'н': 60920,
  'г': 16174,
  'у': 27309,
  'в': 43700,
  'е': 80972,
  'й': 9747,
  'э': 3203,
```

2. Решить задачу 1, распараллелив вычисления с помощью модуля `multiprocessing` . Для обработки каждого файла создать свой собственный процесс.

In [8]:
```
%%file count_char_in_file.py
def count_char_in_file(filename, output):
    char_dict = {}

    with open("10_multiprocessing_data/" + filename) as book:
        for char in book.read().lower():
            if char in char_dict.keys():
                char_dict[char] += 1
            else:
                char_dict[char] = 1

    output.put(({filename: char_dict}))
```

```
Overwriting count_char_in_file.py
```

In [9]:
```
import count_char_in_file
```

In [10]:
```
files = [
    "Dostoevskiy Fedor. Prestuplenie i nakazanie - BooksCafe.Net.txt",
    "Dostoevskiy Fedor. Igrok - BooksCafe.Net.txt"
    ]
```

In [11]:
```
output = mp.Queue()

processes = [mp.Process(target=count_char_in_file.count_char_in_file, args=(filename,

for p in processes:
    p.start()

for p in processes:
    p.join()

results = [output.get() for p in processes]

results
```

```
'\n': 2734,
'u': 285,
'r': 308,
'd': 192,
'v': 87,
'i': 369,
'y': 8,
'_': 4,
'-': 900,
'1': 46,
'0': 22,
'9': 36,
'6': 42,
'm': 401,
'l': 571,
'ж': 2297,
'д': 6681,
'х': 1535,
'ф': 634,
'м': 7106
```

Лабораторная работа 10

1. Разбейте файл `recipes_full.csv` на несколько (например, 8) примерно одинаковых по объему файлов с названиями `id_tag_nsteps_*.csv`. Каждый файл содержит 3 столбца: `id`, `tag` и `n_steps`, разделенных символом `;`. Для разбора строк используйте `csv.reader`.

Важно: вы не можете загружать в память весь файл сразу. Посмотреть на первые несколько строк файла вы можете, написав код, который считывает эти строки.

Подсказка: примерное кол-во строк в файле - 2.3 млн.

```
id;tag;n_steps
137739;60-minutes-or-less;11
137739;time-to-make;11
137739;course;11
```

['name', 'id', 'minutes', 'contributor_id', 'submitted', 'tags', 'n_steps', 'steps', 'description', 'ingredients', 'n_ingredients']

Нам нужны id, tags, n_steps - 1, 5, 6 индексы

In [12]:
```
def get_col(lst):
    return lst[1], lst[5], lst[6]
```

In [13]:
```
with open(r"C:\Users\Red\Downloads\recipes_full\recipes_full.csv", encoding="utf-8")
    reader = csv.reader(csv_file)
    header = get_col(next(reader))

    print(header)

    n_rows = sum(1 for row in reader)

    print(n_rows)
```

```
('id', 'tags', 'n_steps')
2231637
```

In [30]:
```
with open(r"C:\Users\Red\Downloads\recipes_full\recipes_full.csv", encoding="utf-8")
    reader = csv.reader(csv_file)
    for row in range(3):
        print(get_col(next(reader)))
```

```
('id', 'tags', 'n_steps')
('683970', "['mexican', 'healthy-2', 'orange-roughy', 'chicken-thighs-legs', 'freez
er', 'whitefish', 'pork-sausage']", '4')
('1089012', "['brunch', 'ham-and-bean-soup', 'colombian', 'savory-pies', 'refrigera
tor', 'australian', 'served-cold', 'spaghetti']", '1')
```

Получили 2231637 строк в cvs => 2231637 / 8 = 278 954,625

In [14]:
```
with open(r"C:\Users\Red\Downloads\recipes_full\recipes_full.csv", encoding="utf-8")
    reader = csv.reader(csv_file)
    header = get_col(next(reader))

    rows_per_file = 278955

    for i in range(8):
        output_filename =  f"id_tag_nsteps_{i}.csv"
        with open("C:\\Users\\Red\\Downloads\\recipes_full\\" + output_filename, 'w',
            writer = csv.writer(fout, delimiter=';')
            writer.writerow(header)

            for j in range(rows_per_file):
                try:
                    row = get_col(next(reader))
                    writer.writerow(row)
                except StopIteration:
                    break
```

In [45]:
```
with open(r"C:\Users\Red\Downloads\recipes_full\id_tag_nsteps_0.csv", encoding="utf-8
    reader = csv.reader(csv_file)
    for row in range(3):
        print(next(reader))
```

```
['id;tags;n_steps']
["683970;['mexican'", " 'healthy-2'", " 'orange-roughy'", " 'chicken-thighs-legs'",
" 'freezer'", " 'whitefish'", " 'pork-sausage'];4"]
["1089012;['brunch'", " 'ham-and-bean-soup'", " 'colombian'", " 'savory-pies'", "
'refrigerator'", " 'australian'", " 'served-cold'", " 'spaghetti'];1"]
```

2. Напишите функцию, которая принимает на вход название файла, созданного в результате решения задачи 1, считает среднее значение количества шагов для каждого тэга и

возвращает результат в виде словаря.

In [1]:
```
def read_csv(filename):
    tags_dict = {}

    with open(filename, encoding="utf-8") as csv_file:
        reader = csv.reader(csv_file, delimiter=';')

        header = next(reader)

        for row in reader:
            tags = eval(row[1])
            n_steps = int(row[2])

            for tag in tags:
                if tag in tags_dict.keys():
                    tags_dict[tag].append(n_steps)
                else:
                    tags_dict[tag] = [n_steps]

        return dict(map(lambda x: (x[0], np.mean(x[1])), tags_dict.items()))
```

In [104]:
```
read_csv(r"C:\Users\Red\Downloads\recipes_full\id_tag_nsteps_0.csv")
```

Out[104]:
```
{'mexican': 5.302503052503052,
 'healthy-2': 6.328114004222378,
 'orange-roughy': 3.4451697127937337,
 'chicken-thighs-legs': 4.184877440797673,
 'freezer': 3.9144921718185466,
 'whitefish': 3.5132206328565236,
 'pork-sausage': 4.285714285714286,
 'brunch': 6.900977198697069,
 'ham-and-bean-soup': 3.5126970227670755,
 'colombian': 3.5766456266907123,
 'savory-pies': 4.294961081523965,
 'refrigerator': 4.74200503054258,
 'australian': 4.25384024577573,
 'served-cold': 4.940726577437858,
 'spaghetti': 4.1286239281339325,
 'passover': 3.6305130513051305,
 'quick-breads': 4.969485903814262,
 'californian': 3.7421907538525616,
 'namibian': 3.495145631067961,
```

3. Напишите функцию, которая считает среднее значение количества шагов для каждого тэга по всем файлам, полученным в задаче 1, и возвращает результат в виде словаря. Не используйте параллельных вычислений. При реализации выделите функцию, которая объединяет результаты обработки отдельных файлов. Модифицируйте код из задачи 2 таким образом, чтобы иметь возможность получить результат, имея результаты обработки отдельных файлов. Определите, за какое время задача решается для всех файлов.

In [118]:
```
def all_files_tags():
    results = []

    for i in range(8):
        results.append(read_csv(f"C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nst

    return merge_results(results)
```

In [4]:
```
def merge_results(results_list):
    merged = {}
    for dic in results_list:
        for k, v in dic.items():
            if k in merged.keys():
                merged[k].append(v)
            else:
                merged[k] = [v]

    return dict(map(lambda x: (x[0], np.mean(x[1])), merged.items()))
```

In [120]:
```
%time all_res = all_files_tags()
```

```
CPU times: total: 36.3 s
Wall time: 1min 13s
```

In [122]:
```
all_res
```

```
 'costa-rican': 3.513651715958942,
 'marinara-sauce': 3.5229598888569664,
 'macaroni-and-cheese': 3.53191502882974,
 'bisques-cream-soups': 4.132200032659291,
 'cantonese': 3.563640890295793,
 'scandinavian': 3.9535886791411188,
 'french': 4.472903084762207,
 'hanukkah': 3.7727942199561832,
 'chinese-new-year': 3.51816053468362,
 'pumpkin-bread': 3.5090501246924055,
 '1-day-or-more': 4.459913147015779,
 'amish-mennonite': 3.561238944348162,
 'mixer': 4.3016811638550285,
 'blueberries': 3.9648641975885166,
 'spanish': 3.862667315241983,
 'pressure-cooker': 3.667358858376653,
 'birthday': 3.624406285033168,
 'low-fat': 5.944633279430655,
 'african': 4.371693978010992,
```

4. Решите задачу 3, распараллелив вычисления с помощью модуля `multiprocessing` . Для обработки каждого файла создайте свой собственный процесс. Определите, за какое время задача решается для всех файлов.

In [2]:
```
%%file read_csv_proc_file.py
import numpy
import csv
import logging



def read_csv_proc(filename, output):
    logging.basicConfig(filename='read_csv_proc.log', level=logging.DEBUG)

    logging.debug(f"function was entered, {filename = }")
    tags_dict = {}

    with open(filename, encoding="utf-8") as csv_file:
        reader = csv.reader(csv_file, delimiter=';')
        logging.debug("file opened")
        header = next(reader)

        for row in reader:
            tags = eval(row[1])
            n_steps = int(row[2])

            for tag in tags:
                if tag in tags_dict.keys():
                    tags_dict[tag].append(n_steps)
                else:
                    tags_dict[tag] = [n_steps]
    logging.debug("reading ended")
    result = dict(map(lambda x: (x[0], numpy.mean(x[1])), tags_dict.items()))

    output.put(({filename: result}))
    logging.debug(f"function ended, {filename = }")
```

```
Overwriting read_csv_proc_file.py
```

In [1]:
```
import multiprocessing as mp
import numpy as np
import csv
```

In [2]:
```
files = [f"C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_{i}.csv" for i in r
files
```

Out[2]:
```
['C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_0.csv',
 'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_1.csv',
 'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_2.csv',
 'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_3.csv',
 'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_4.csv',
 'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_5.csv',
 'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_6.csv',
 'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_7.csv']
```

In [3]:
```
import read_csv_proc_file
```

In [9]:
```
%%time

output = mp.Queue()
processes = [mp.Process(target=read_csv_proc_file.read_csv_proc, args=(filename, outp

for p in processes:
    p.start()

results = [output.get() for p in processes]
```

```
CPU times: total: 15.6 ms
Wall time: 10.5 s
```

In [11]:
```
results
```

```
    'greens': 4.6458886932018515,
   'caribbean': 3.9367768595041324,
   'austrian': 3.5867768595041323,
   'drop-cookies': 4.397701149425288,
   'micro-melanesia': 3.524902555218709,
   'meatloaf': 3.581547879317884,
   'gumbo': 3.568758344459279,
   'mussels': 3.616774193548387,
   'dips-lunch-snacks': 3.5754633715798763,
   'veggie-burgers': 3.4867408041060735,
   'chicken-livers': 3.5140267587397496,
   'pears': 3.8105443634804974,
   'snacks-sweet': 3.4694960212201593,
   'lasagne': 3.4872711031710586,
   '1-day-or-more': 4.486750917244191,
   'norwegian': 3.526407682234832,
   'turkey': 4.514832162373146,
   'moose': 3.497333333333333}},
 {'C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_4.csv': {'stocks': 3.60
19677996422184,
```

In [16]:
```
%%time
results_list = []
for k, v in results[0].items():
    results_list.append(v)

merged = merge_results(results_list)
merged
```

```
CPU times: total: 0 ns
Wall time: 0 ns
```

Out[16]:
```
{'time-to-make': 9.339157042740105,
 'course': 9.333062907751073,
 'main-ingredient': 9.373848753696663,
 'cuisine': 9.179081520544935,
 'preparation': 9.35765342262863,
 'north-american': 8.186776464971954,
 'casseroles': 5.349056603773585,
 'main-dish': 9.523406906372374,
 'eggs-dairy': 7.576754741449181,
 'poultry': 7.775295914471172,
 'mexican': 5.184485006518905,
 'oven': 8.453335524153795,
 'grains': 5.19782034346103,
 'cheese': 6.531529354916646,
 'turkey': 4.460485133020344,
 'one-dish-meal': 7.097025171624714,
 'meat': 9.055418322414349,
```

Время выполнения без распараллеливания:

```
CPU times: total: 36.3 s
Wall time: 1min 13s
```

С распараллеливанием:

```
CPU times: total: 15.6 ms
Wall time: 10.5 s
```

5. (*) Решите задачу 3, распараллелив вычисления с помощью модуля `multiprocessing`. Создайте фиксированное количество процессов (равное половине количества ядер на компьютере). При помощи очереди передайте названия файлов для обработки процессам и при помощи другой очереди заберите от них ответы.

In [8]:
```
%%file read_csv_parallel.py
import logging
import multiprocessing as mp
import csv
import numpy


def read_csv_parallel(working_q, output_q):
    logging.basicConfig(filename='read_csv_proc.log', level=logging.DEBUG)

    while True:
        if working_q.empty():
            logging.debug("working_q is empty")
            break
        filename = working_q.get()
        logging.debug(f"got file {filename[-5]}")



        tags_dict = {}

        with open(filename, encoding="utf-8") as csv_file:
            reader = csv.reader(csv_file, delimiter=';')
            logging.debug(f"opened {filename[-5]}")
            header = next(reader)

            for row in reader:
                tags = eval(row[1])
                n_steps = int(row[2])

                for tag in tags:
                    if tag in tags_dict.keys():
                        tags_dict[tag].append(n_steps)
                    else:
                        tags_dict[tag] = [n_steps]
        logging.debug(f"ended {filename[-5]}")
        result = dict(map(lambda x: (x[0], numpy.mean(x[1])), tags_dict.items()))


        output_q.put(result)
        logging.debug(f"put {filename[-5]}")
```

```
Overwriting read_csv_parallel.py
```

In [9]:
```
import multiprocessing as mp
import read_csv_parallel
```

In [10]:
```
%%time

working_q = mp.Queue()
output_q = mp.Queue()

files = [f"C:\\Users\\Red\\Downloads\\recipes_full\\id_tag_nsteps_{i}.csv" for i in r

for i in files:
    working_q.put(i)

processes = [mp.Process(target=read_csv_parallel.read_csv_parallel,args=(working_q, c

for proc in processes:
    proc.start()

results_bank = []

while len(results_bank) < 8:
    results_bank.append(output_q.get())

merge_results(results_bank)
```

```
CPU times: total: 0 ns
Wall time: 18.1 s
```

Out[10]:
```
{'stocks': 3.628102544697785,
 'ethiopian': 3.5094618656633587,
 'time-to-make': 9.27861745373393,
 'melons': 3.616346619601838,
 'fall': 5.55601949143708,
 'tomatoes': 4.8644227716622686,
 'course': 9.274718384210226,
 'main-ingredient': 9.315238256444898,
 'preparation': 9.293519974373277,
 'occasion': 9.13623677986151,
 'appetizers': 6.230953726886838,
 'lunch': 6.678987484403059,
 'snacks': 4.974766234391118,
 'beef': 6.936718513461723,
 'oven': 8.349075937046113,
 'holiday-event': 8.307959334192633,
 'kid-friendly': 7.0949402721473405
```