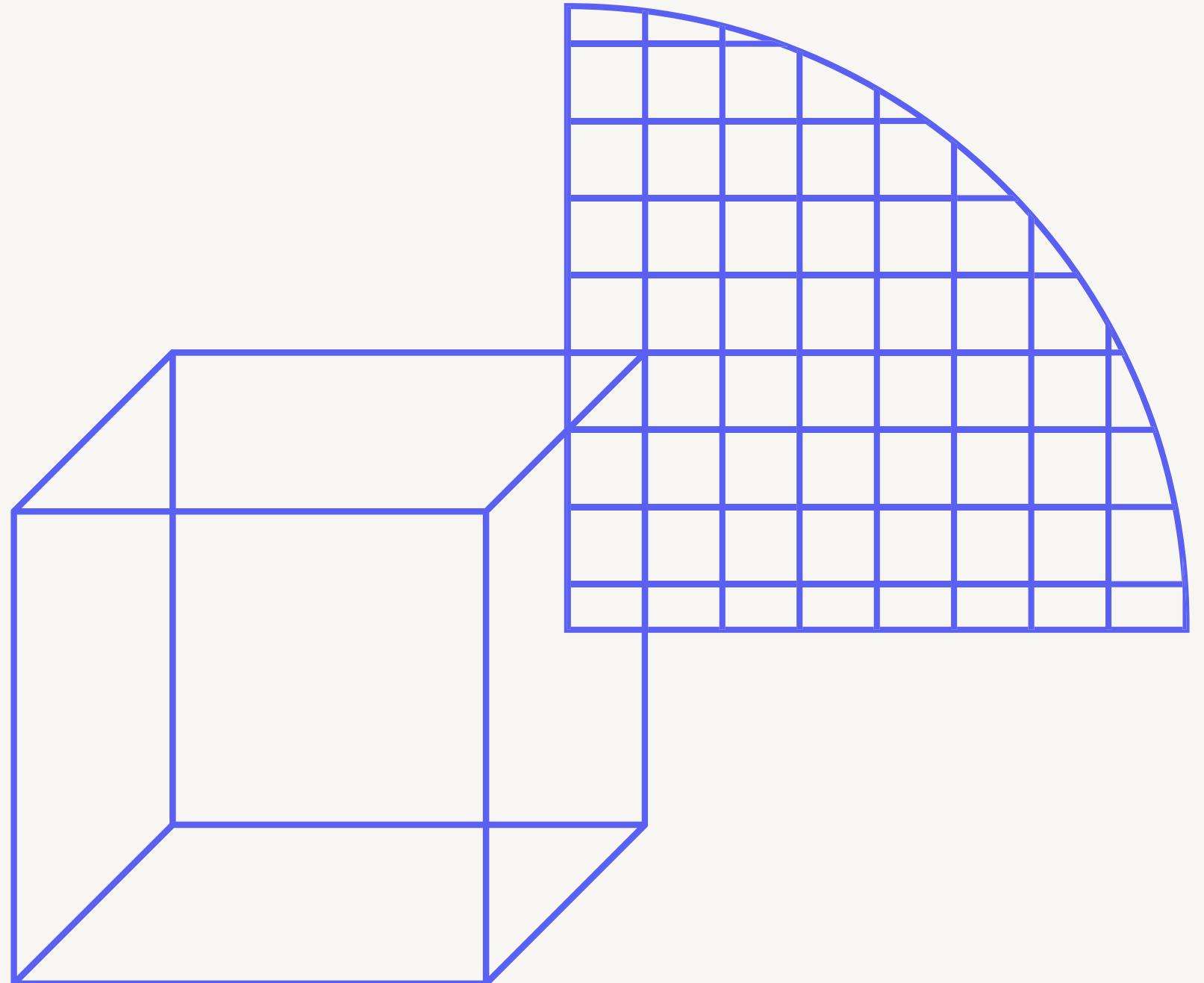


ПРОЕКТ ПО РУТНОН

Алгоритм для поиска лучшей кофейни
(на примере сети "Surf Coffee")



ИДЕЯ ПРОЕКТА



Написать **программу**, которая при вводе любой сети кофеен ищет информацию про **каждое** заведение, собирает интересующую нас, как студентов, информацию и **помогает выбрать лучшее место** для отдыха / работы

Основные критерии

Критерии, которые мы выделили для себя:



Местоположение (адрес, район, метро, расстояние от метро)



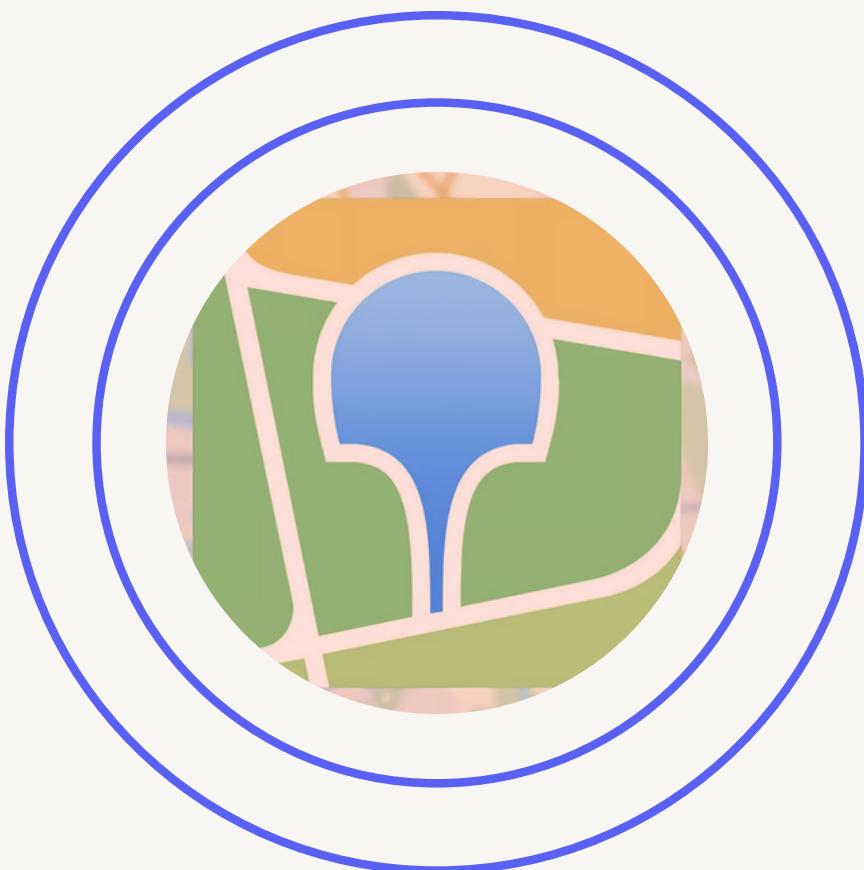
"Качество" (рейтинг, загруженность, время работы)



Бюджет (средний чек)

Выбор платформы

Почему "2 ГИС"?



1

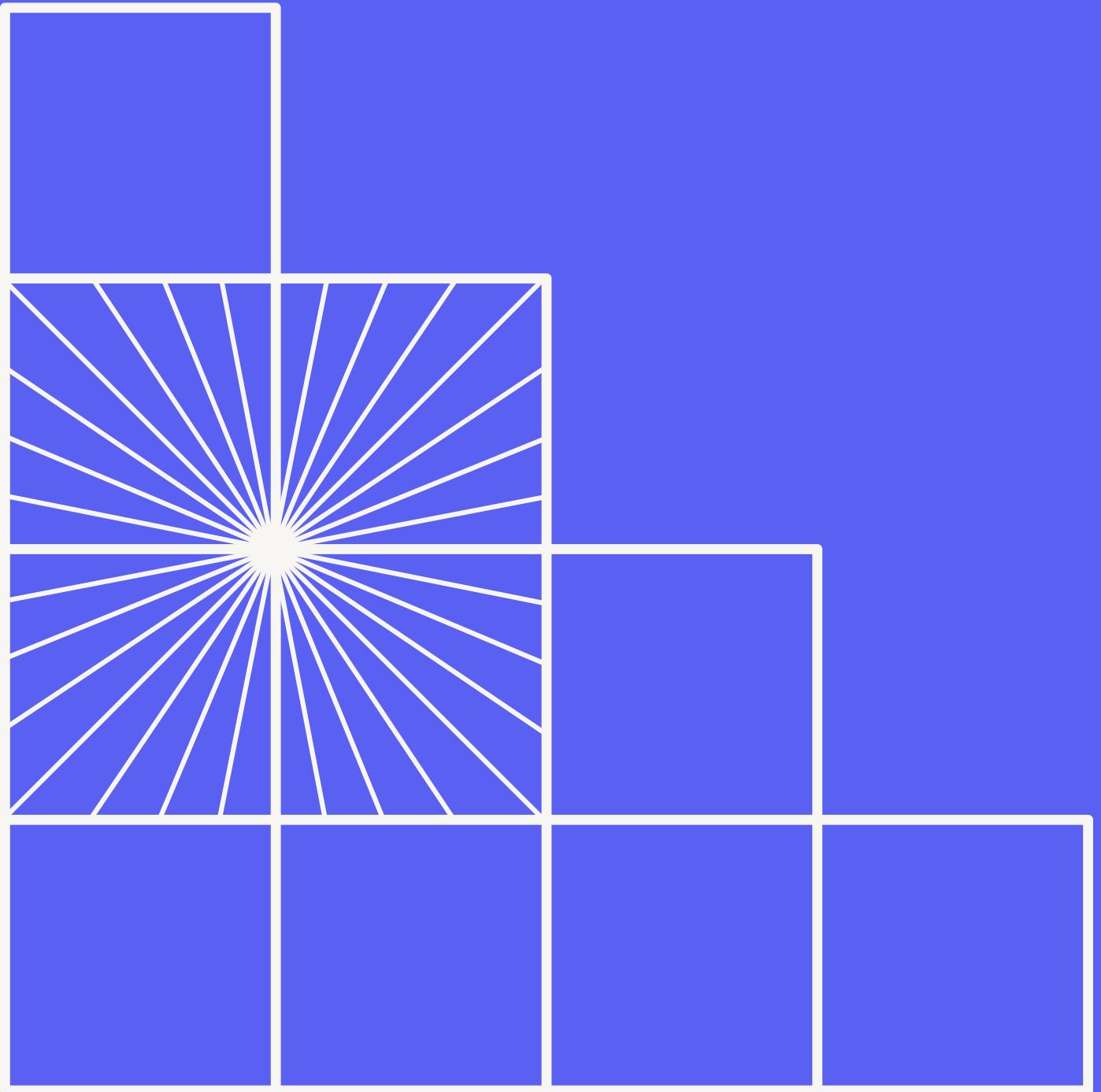
Дает более
полную
информацию, чем
конкуренты
(Яндекс.Карты,
Google Maps)

2

При поиске
выдает
информацию
сразу про все
кофейни сети (а не
только рядом)

Часть I

Первичный сбор данных



```
1 s = Service('C:/Python/chromedriver')
2 br = wb.Chrome(service = s)
3
4 url = 'https://2gis.ru/moscow/'
5 br.get(url)
```

1

Открываем браузер в автоматическом режиме, переходим на стартовую страницу "2 ГИС"

```
1 field = br.find_element(By.CSS_SELECTOR, '#root > div > div > div._1sf34doj > div._1u4plm2 > div:nth-child(2)\' \
2                               > div > div > div:nth-child(1) > div > div > div > div > div > div._ubuirc > form > div > input')
3 field.send_keys('Surf Coffee \n')
4 br.implicitly_wait(2)
```

2

Передаем поисковой запрос на сайт и немного ждем...

```
1 url0 = 'https://2gis.ru'
2 nextUrls = set()
3 soup = getSoupFromCurrentPage1()
4 for link in soup.find_all('a', attrs = {'class' : '_12164l30'}):
5     nextUrls.add(url0 + link.get('href'))
```

3

Собираем ссылки на поисковые страницы в специальное множество

```
1 nextUrls
```

```
{'https://2gis.ru/moscow/search/Surf%20Coffee%20/page/2',
 'https://2gis.ru/moscow/search/Surf%20Coffee%20/page/3',
 'https://2gis.ru/moscow/search/Surf%20Coffee%20/page/4'}
```

Посмотрим, как выглядит получившееся множество

```

1 places = []

2

3 for place in getPlacesFromSoup(soup):
4     places.append(getLinkToPlace(place))

5

6 for url in nextUrls:
7     soup = getSoupFromCurrentPage2(url)
8     sleep(3)
9     for place in getPlacesFromSoup(soup):
10        if len(place.find_all('a', attrs = {'class' : '_1soh43j8'})) == 0:
11            places.append(getLinkToPlace(place))
12        else:
13            pass

```

4

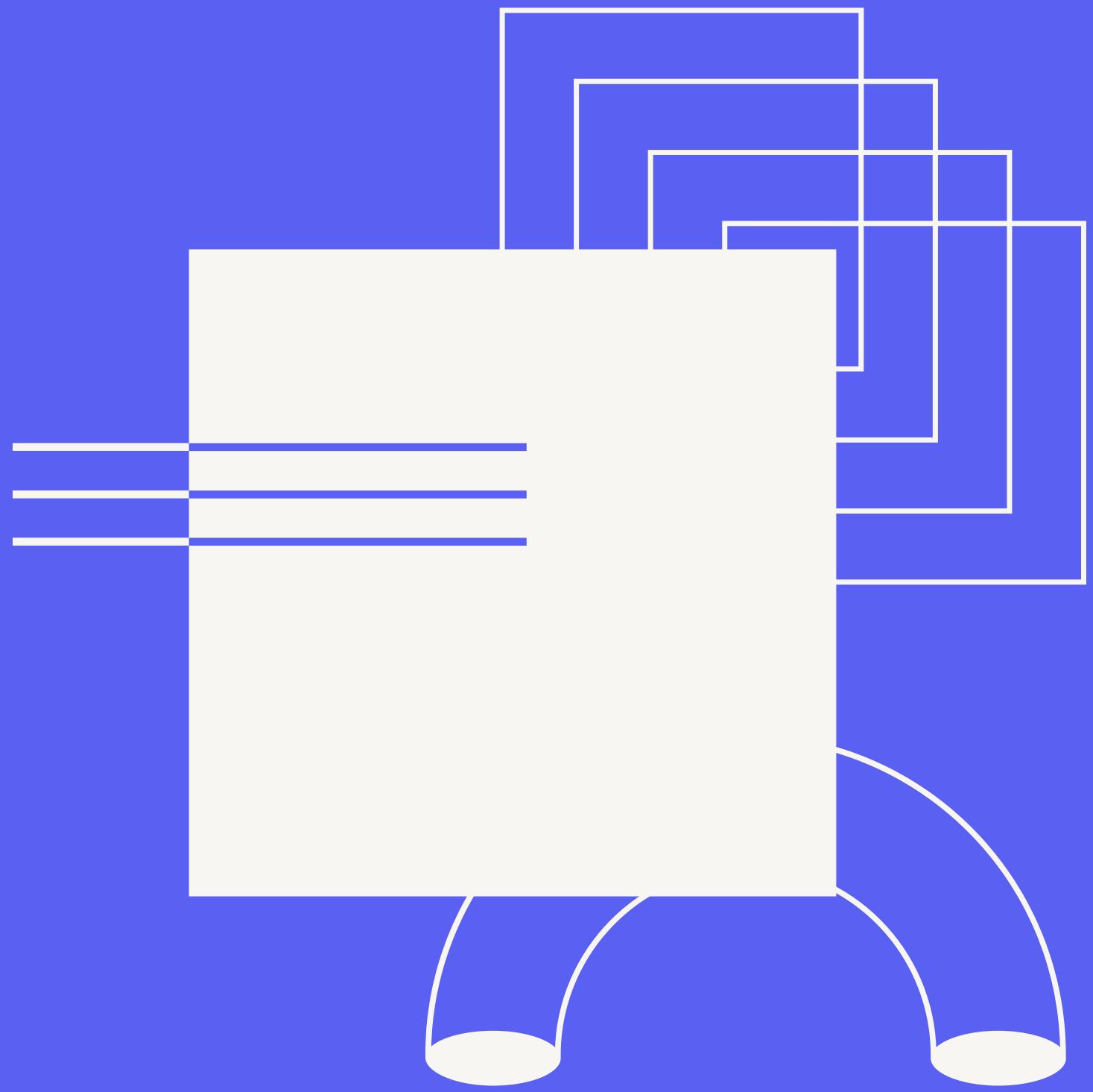
Проходим по каждой поисковой странице (первая уже открыта) и собираем ссылки на все кофейни сети в список. Не берем ссылку, если филиал закрылся

```

1 ## определяем функцию, отвечающую за поиск тегов с кофейнями на странице
2 def getPlacesFromSoup(soup):
3     """
4         Функция требует на вход параметр: преобразованный код страницы ("суп").
5         Функция ищет все теги 'div' со значением '_1hf7139' атрибута 'class',
6         в которых находится информация о заведениях, размещенная на поисковой странице.
7         Возвращает список с тегами, в которые входят все заведения, расположенные на текущей поисковой странице.
8         Требует импорта модуля `bs4`
9     """
10    places = soup.find_all('div', attrs = {'class' : '_1hf7139'})
11    return places
12
13 ## определяем функцию, отвечающую за забор ссылки на кофейню
14 def getLinkToPlace(soup):
15     """
16         Функция требует на вход параметр: преобразованный код страницы ("суп").
17         Функция ищет относительную ссылку на заведение, забирает ее и соединяет с "базовой" ссылкой на домен.
18         Функция возвращает полную ссылку на заведение (строка).
19         Требует импорта модуля `bs4`
20     """
21
22     url2GIS = 'https://2gis.ru'
23     placeUrlShort = soup.find_all('a', attrs = {'class' : '_1rehek'})[0].get('href')
24     placeUrlNew = url2GIS + placeUrlShort
25     return placeUrlNew

```

*используемые собственные функции, представляющие интерес



Часть II

Сбор основной информации

1

```
1 df = {}
2 n = 1 # Счетчик на случай повторений
3
4 for place in places:
5     soup = getSoupFromCurrentPage2(place)
6     sleep(3)
7
8     ## Анализ первой страницы с основной информацией о кафе
9
10    # Название
11    name = findInfoFromG1Soup_span('_oqoid')[0].text.upper()
12    print(f'Сейчас анализируется место под названием {name}') # Проверка
13
14    # Комментарий
15    if len(findInfoFromG1Soup_div('_avvjvo')) != 0:
16        extra = findInfoFromG1Soup_div('_avvjvo')[0].text
17    else:
18        extra = 'Нет доп. информации'
19
20    # Рейтинг
21    if len(findInfoFromG1Soup_span('_1n8h0vx')) != 0:
22        ranking = float(findInfoFromG1Soup_span('_1n8h0vx')[0].text)
23    else:
24        ranking = None
```

Открываем страницу филиала,
забираем ее код, немного ждем

2

Работаем с первым блоком
информации на основной странице
кофейни

Забираем информацию про:

- название
- доп. информацию
- рейтинг

...

```

26 # Переходим в блок со доп. информацией
27 for elem in findInfoFromG1Soup_div('_599hh'):
28     # Адрес
29     if len(elem.find_all('a', attrs = {'class' : '_2lcm958'})) != 0:
30         adress = ' '.join(elem.find_all('a', attrs = {'class' : '_2lcm958'})[0].text.split())
31     else:
32         adress = None
33
34     # Район
35     if len(findInfoFromSoup_div(elem, '_1p8iqzw')[0].text.split(', ')[0]) != 0:
36         area = findInfoFromSoup_div(elem, '_1p8iqzw')[0].text.split(', ')[0]
37     else:
38         area = None
39
40     # Рабочее время сегодня
41     if len(findInfoFromSoup_div(elem, '_18zamfw')) != 0:
42         if 'Сегодня' in findInfoFromSoup_div(elem, '_18zamfw')[0].text:
43             workTimeToday = findInfoFromSoup_div(elem, '_18zamfw')[0].text[:24].strip()
44         elif 'Ежедневно' in findInfoFromSoup_div(elem, '_18zamfw')[0].text:
45             workTimeToday = findInfoFromSoup_div(elem, '_18zamfw')[0].text[:26].strip()
46     else:
47         workTimeToday = None
48
49     # Загруженность сегодня
50     if len(findInfoFromSoup_div(elem, '_mi65ux')) != 0:
51         occupancy = []
52         for info in findInfoFromSoup_div(elem, '_mi65ux'):
53             t = 0
54             for time in findInfoFromSoup_div(info, '_yxu57z'):
55                 if time.get('style').endswith('transform'):
56                     x = str(int(float(time.get('style').split('(')[1].split(')')[0])*100)) + '%'
57                     occupancy.append((t, x))
58                     t += 1
59     else:
60         occupancy = None

```

- ...
- адрес
 - район
 - рабочее время (сегодня)
 - загруженность (сегодня)

3

```

62 ## Переходим на страницу с другой информацией
63 ## Кликаем на слово "Инфо"
64
65 WebDriverWait(br, 20).until(EC.element_to_be_clickable((By.LINK_TEXT, "Инфо"))).click()
66 br.implicitly_wait(2)
67 soup = getSoupFromCurrentPage1()
68 sleep(1)
69
70 # Средний чек
71 for info in findInfoFromGlSoup_span('_14quei'):
72     for tags in findInfoFromSoup_span(info, '_er2xx9'):
73         if 'чек' in tags.text:
74             meanCheque = int(tags.text.split()[2])
75             break
76         else:
77             meanCheque = None
78     break
79
80 # Станции метро
81 metros = []
82 for info in findInfoFromGlSoup_div('_599hh'):
83     for header in findInfoFromSoup_span(info, '_btwk9a2'):
84         if 'Транспорт' in header.text:
85             for metro in findInfoFromSoup_div(info, '_172gbf8')[1]:
86                 for i in range(len(metro.find_all('a', attrs = {'class' : '_1rehek'}))):
87                     metros.append((metro.find_all('a', attrs = {'class' : '_1rehek'})[i].text))
88
89 # Время до станции
90 metroMinutes = []
91 metroMetres = []
92 for info in findInfoFromGlSoup_div('_599hh'):
93     for header in findInfoFromSoup_span(info, '_btwk9a2'):
94         if 'Транспорт' in header.text:
95             for metro in findInfoFromSoup_div(info, '_172gbf8')[1]:
96                 for i in range(len(findInfoFromSoup_span(metro, '_5fyrv3'))):
97                     data = findInfoFromSoup_span(metro, '_5fyrv3')[i].text
98                     if 'мин' in data:
99                         metroMinutes.append(int(data.split()[0]))
100                    else:
101                        metroMetres.append(checkMetres(float(data.split()[0]))) # проверка на метры/км

```

Переходим (кликаем) на страницу с другой информацией и забираем ее код

4

Работаем со вторым блоком информации на дополнительной странице кофейни

Забираем информацию про:

- средний чек
- станции метро по близости
- время и расстояние от метро

5

```
105 if name not in df:  
106     df[name] = [  
107         ranking, adress, area, metros, metroMinutes, metroMetres,  
108         meanCheque, workTimeToday, occupancy, extra  
109     ]  
110 else:  
111     df[name + f' {n}'] = [  
112         ranking, adress, area, metros, metroMinutes, metroMetres,  
113         meanCheque, workTimeToday, occupancy, extra  
114     ]  
115  
116     print(f'Место {name} успешно проанализировано и записано \n') # Закончили  
117     n += 1  
118  
119     # Переход к новой ссылке...
```

Записываем собранную информацию в словарь, проверяя, нет ли там информации про кофейню с таким же названием (чтобы не потерять данные)

Печатаем пользователю сообщение, что анализ кофейни (указываем название) завершен

6

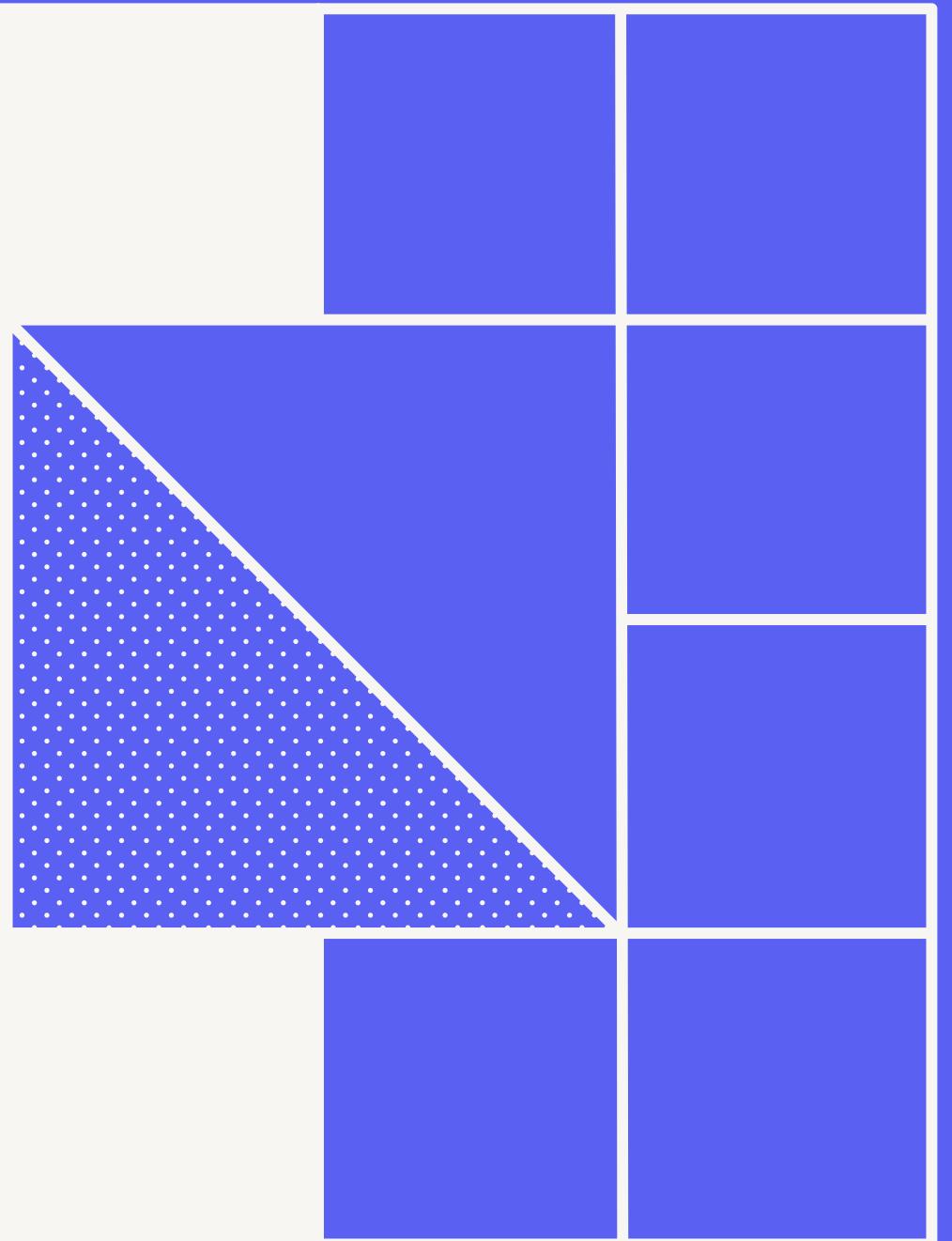
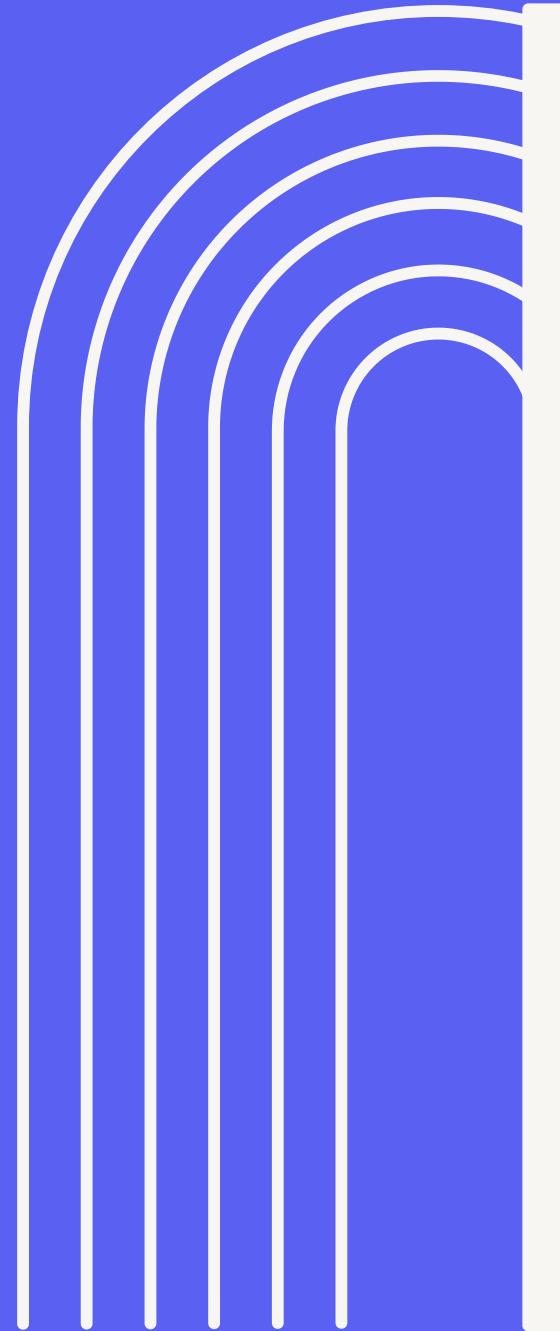
Переходим к новой ссылке



Часть III

Представление

информации



```

1 coffee_df = pd.DataFrame.from_dict(df, orient='index', columns=[  

2     'Рейтинг', 'Адрес', 'Район', 'Метро', 'Минут до метро',  

3     'Метров до метро', 'Средний чек', 'Рабочее время сегодня',  

4     'Загруженность сегодня', 'Доп. инф.'  

5 ])  

6  

7 coffee_df

```

	Рейтинг	Адрес	Район	Метро	Минут до метро	Метров до метро	Средний чек	Рабочее время сегодня	Загруженность сегодня	Доп. инф.
SURF COFFEE	NaN	Измайловский проезд, 5а	Измайлово район	[Измайловская]	[21]	[2000.0]	NaN	None	None	Скоро открытие
SURF COFFEE × SOLYANKA	4.4	Солянка, 1/2	Басманный район	[Китай-город, Китай-город, Лубянка]	[2, 2, 11]	[200.0, 250.0, 1000.0]	300.0	Сегодня с 08:00 до 22:30	None	Нет доп. информации
SURF COFFEE × MAYAK	4.5	Тверская, 25/12	Тверской район	[Маяковская, Тверская, Пушкинская]	[3, 5, 5]	[300.0, 450.0, 450.0]	300.0	Сегодня с 08:00 до 22:00	[(5, 30%), (6, 50%), (7, 50%), (8, 60%), (9, 6...	Нет доп. информации
SURF COFFEE X ARBAT	3.8	улица Новый Арбат, 7 ст А	Арбат район	[Арбатская, Арбатская, Александровский Сад]	[5, 5, 10]	[450.0, 500.0, 900.0]	350.0	Сегодня с 08:00 до 23:00	None	Нет доп. информации
SURF COFFEE × SECRET SPOT	4.7	Конный переулок, 12	Якиманка район	[Шаболовская, Октябрьская, Октябрьская]	[7, 13, 14]	[600.0, 1000.0, 1000.0]	300.0	Сегодня с 08:00 до 22:00	None	Нет доп. информации

1

Создаем красивую таблицу

```
1 coffee_df.to_csv('coffee_df.csv', encoding='utf-8-sig')
```

```
1 coffee_df.to_excel('coffee_df2.xlsx', encoding='utf-8-sig')
```

2

Выгружаем данные в формате csv или xlsx

```

1 print(coffee_df['Район'].value_count)

Тверской район          9
Басманный район        8
Замоскворечье район    3
Хамовники район        3
Пресненский район      3
Хорошёвский район      3
Якиманка район         2
Измайлово район        1
Красносельский район   1
Ясенево район           1
Красногорск              1
Аэропорт район          1
Крылатское район        1
Мещанский район          1
Очаково-Матвеевское район 1
Гагаринский район        1
Таганский район          1
Щукино район             1
Проспект Вернадского района 1
Бутырский район          1
Арбат район               1
Балашиха-1 м-н            1
Name: Район, dtype: int64

```

3 ВЫВОДИМ описательные статистики

```

1 print('Среднее значение среднего чека в кофейнях:', round(coffee_df['Средний чек'].mean()))
2 print('Среднее значение рейтинга в кофейнях:', round(coffee_df['Рейтинг'].mean(), 2))
3
4 # Ищем среднее время от метро в минутах
5 meanMetro = []
6 for i in coffee_df['Минут до метро']:
7     meanMetro.append(round(mean(i), 1))
8
9 print('Среднее значение минут от метро до кофейни:', round(mean(meanMetro), 1))
10
11 # Ищем среднее число метров от метро
12 meanMeters = []
13 for i in coffee_df['Метров до метро']:
14     meanMeters.append(round(mean(i), 1))
15
16 print('Среднее значение метров от метро до кофейни:', round(mean(meanMeters)))

```

Среднее значение среднего чека в кофейнях: 296

```

1 print(coffee_df['Рабочее время сегодня'].mode())
2 print('\n')
3 print(coffee_df['Рабочее время сегодня'].value_count)

0    Сегодня с 10:00 до 22:00
dtype: object

Сегодня с 10:00 до 22:00      15
Сегодня с 09:00 до 22:00      13
Сегодня с 10:00 до 23:00       8
Сегодня с 09:00 до 23:00       2
Сегодня с 10:00 до 21:00       2
Сегодня с 10:00 до 22:30       1
Ежедневно с 10:00 до 22:00     1
Сегодня с 11:00 до 21:00       1
Ежедневно с 09:30 до 22:00     1
Name: Рабочее время сегодня, dtype: int64

```

```

1 print('Квантиль уровня 0.25 для рейтинга:', coffee_df['Рейтинг'].quantile(0.25))
2 print('Квантиль уровня 0.75 для рейтинга:', coffee_df['Рейтинг'].quantile(0.75))
3 print('Квантиль уровня 0.25 для среднего чека:', coffee_df['Средний чек'].quantile(0.25))
4 print('Квантиль уровня 0.75 для среднего чека:', coffee_df['Средний чек'].quantile(0.75))

```

Квантиль уровня 0.25 для рейтинга: 4.0
 Квантиль уровня 0.75 для рейтинга: 4.95
 Квантиль уровня 0.25 для среднего чека: 254.75
 Квантиль уровня 0.75 для среднего чека: 350.0

```

1 modeMetro = []
2 for i in coffee_df['Метро']:
3     setMet = set(i)
4     modeMetro.extend(list(setMet))
5
6 print(my_mode(modeMetro))

```

['Театральная', 'Китай-город', 'Лубянка', 'Ма

```

1 print('Квантиль уровня 0.2 для времени от метро в минутах:', coffee_df['Минут до метро'].quantile(0.2))
2 print('Квантиль уровня 0.8 для времени от метро в минутах:', coffee_df['Минут до метро'].quantile(0.8))
3 print('Квантиль уровня 0.2 для числа метров от метро:', coffee_df['Метров до метро'].quantile(0.2))
4 print('Квантиль уровня 0.8 для числа метров от метро:', coffee_df['Метров до метро'].quantile(0.8))

```

Квантиль уровня 0.2 для времени от метро в минутах: 7.0
 Квантиль уровня 0.8 для времени от метро в минутах: 21.0
 Квантиль уровня 0.2 для числа метров от метро: 100.0
 Квантиль уровня 0.8 для числа метров от метро: 200.0

```

1 print('Медиана для среднего чека:', coffee_df['Средний чек'].median())
2 print('Медиана для рейтинга:', coffee_df['Рейтинг'].median())
3
4 # Ищем медианное время от метро в минутах
5 medianMetro = []
6 for i in coffee_df['Минут до метро']:
7     medianMetro.append(i)
8
9 print('Медиана для времени от метро в минутах:', median(medianMetro))
10
11 # Ищем медианное число метров от метро
12 medianMeters = []
13 for i in coffee_df['Метров до метро']:
14     medianMeters.append(i)
15
16 print('Медиана для расстояния от метро в метрах:', median(medianMeters))

Медиана для среднего чека: 300.0
Медиана для рейтинга: 4.5
Медиана для времени от метро в минутах: 7.0
Медиана для расстояния от метро в метрах: 700.0

```

Визуализируем результат

```

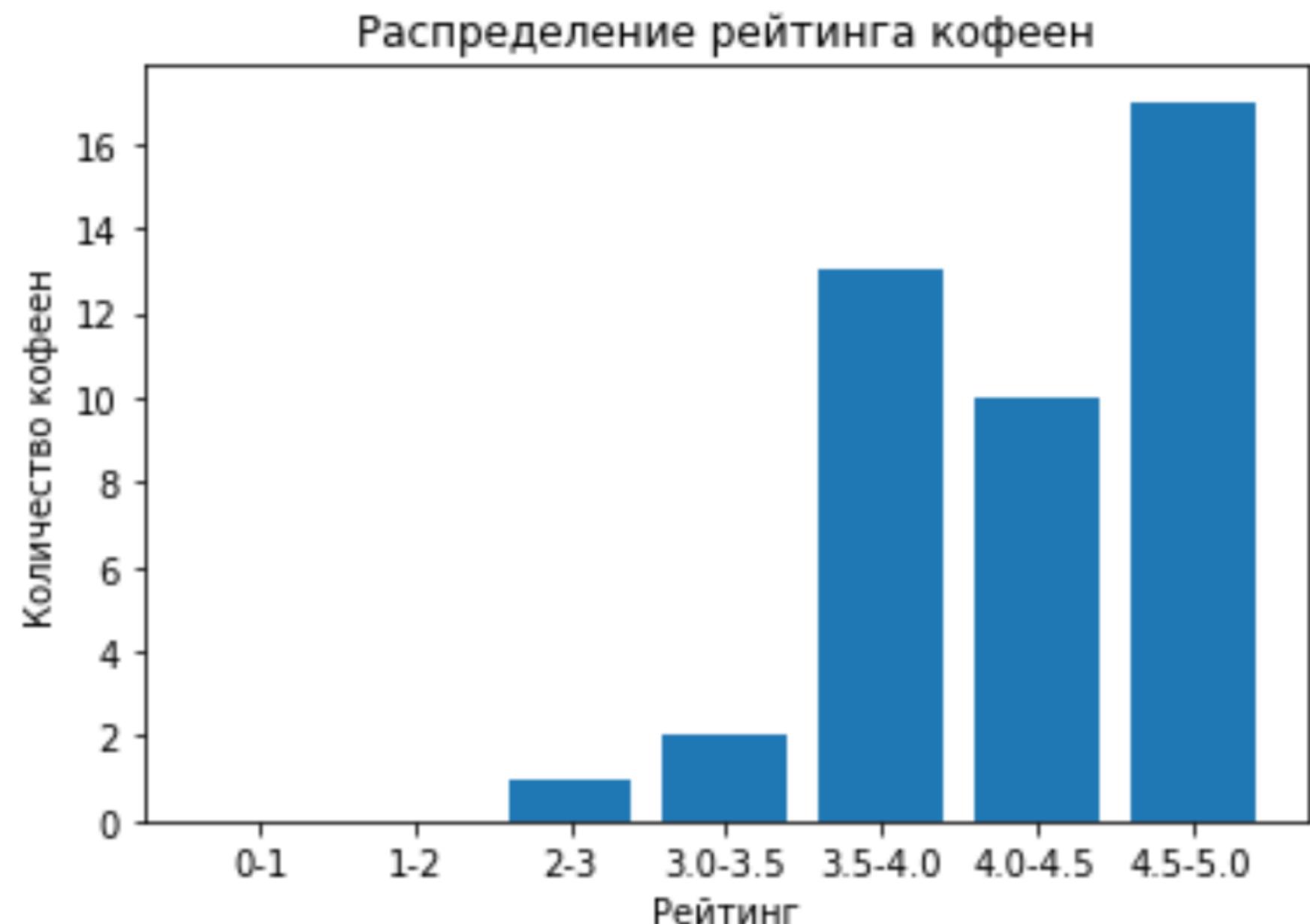
1 data_0={}
2 b=[]
3 for a in coffee_df['Рейтинг']:
4     b.append(a)
5 for i in b:
6     if i not in data_0:
7         data_0[i] = 1
8     else:
9         data_0[i] += 1
10
11 data = {'0-1' : 0, '1-2' : 0, '2-3' : 0, '3.0-3.5' : 0, '3.5-4.0' : 0,
12      '4.0-4.5' : 0, '4.5-5.0' : 0}
13
14 for k, v in data_0.items():
15     if k <= 1.0:
16         data['0-1'] += v
17     elif k > 1.0 and k <= 2.0:
18         data['1-2'] += v
19     elif k > 2.0 and k <= 3.0:
20         data['2-3'] += v
21     elif k > 3.0 and k <= 3.5:
22         data['3.0-3.5'] += v
23     elif k > 3.5 and k <= 4.0:
24         data['3.5-4.0'] += v
25     elif k > 4.0 and k <= 4.5:
26         data['4.0-4.5'] += v
27     elif k > 4.5 and k <= 5.0:
28         data['4.5-5.0'] += v

```

```

1 plt.bar(data.keys(), data.values())
2
3 plt.title('Распределение рейтинга кофеен')
4
5 plt.xlabel('Рейтинг')
6 plt.ylabel('Количество кофеен')
7
8 plt.show()

```



```

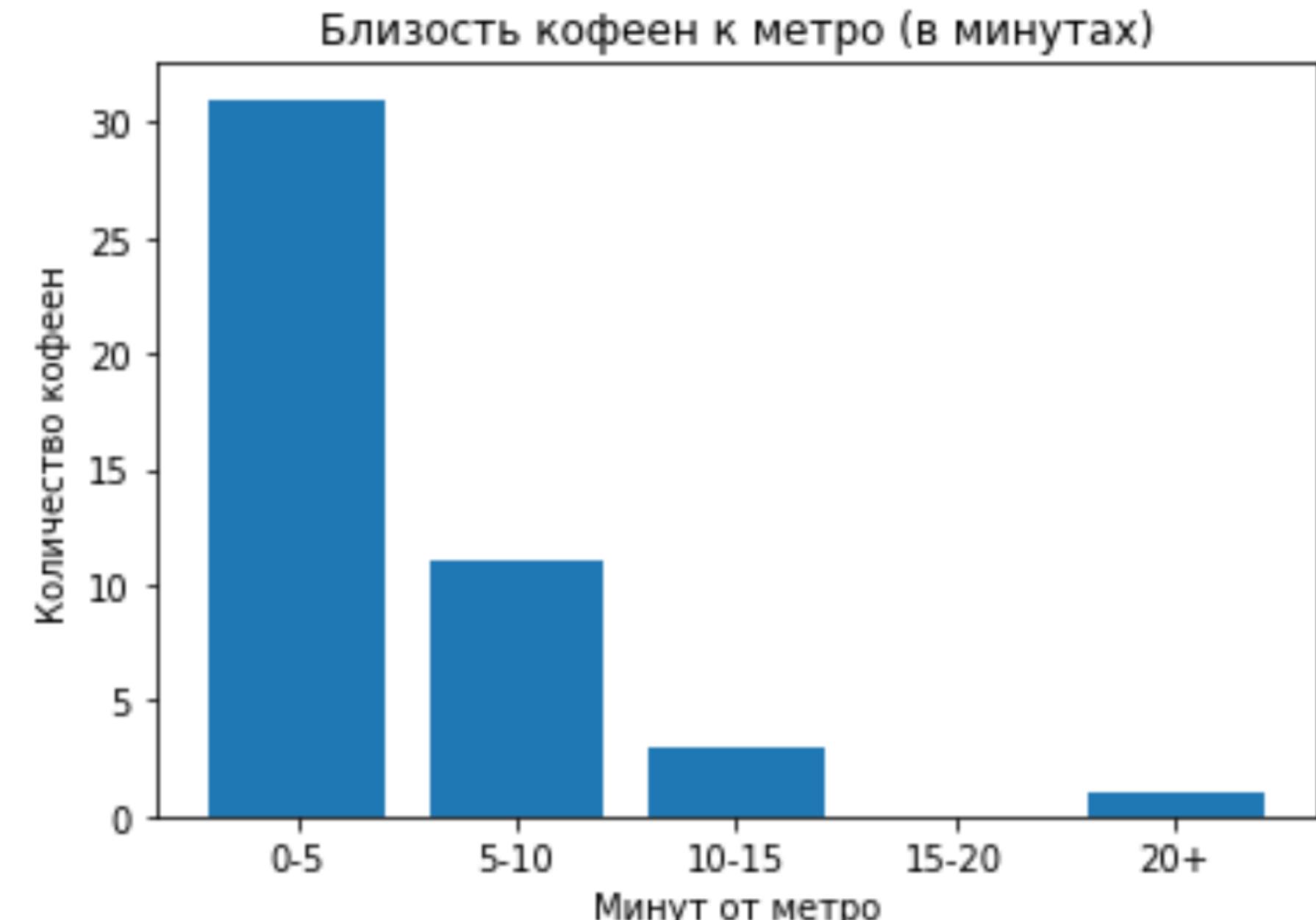
1 data1_0={}
2 for x in coffee_df['Минут до метро']:
3     x = min(x)
4     if x not in data1_0:
5         data1_0[x] = 1
6     else:
7         data1_0[x] += 1
8
9 data1 = {'0-5' : 0, '5-10' : 0, '10-15' : 0,
10      '15-20' : 0, '20+' : 0}
11
12 for k, v in data1_0.items():
13     if k <= 5:
14         data1['0-5'] += v
15     if k > 5 and k <= 10:
16         data1['5-10'] += v
17     if k > 10 and k <= 15:
18         data1['10-15'] += v
19     if k > 15 and k <= 20:
20         data1['15-20'] += v
21     if k > 20:
22         data1['20+'] += v

```

```

1 plt.bar(data1.keys(), data1.values())
2
3 plt.title("Близость кофеен к метро (в минутах")
4
5 plt.xlabel('Минут от метро')
6 plt.ylabel('Количество кофеен')
7
8 plt.show()

```



```

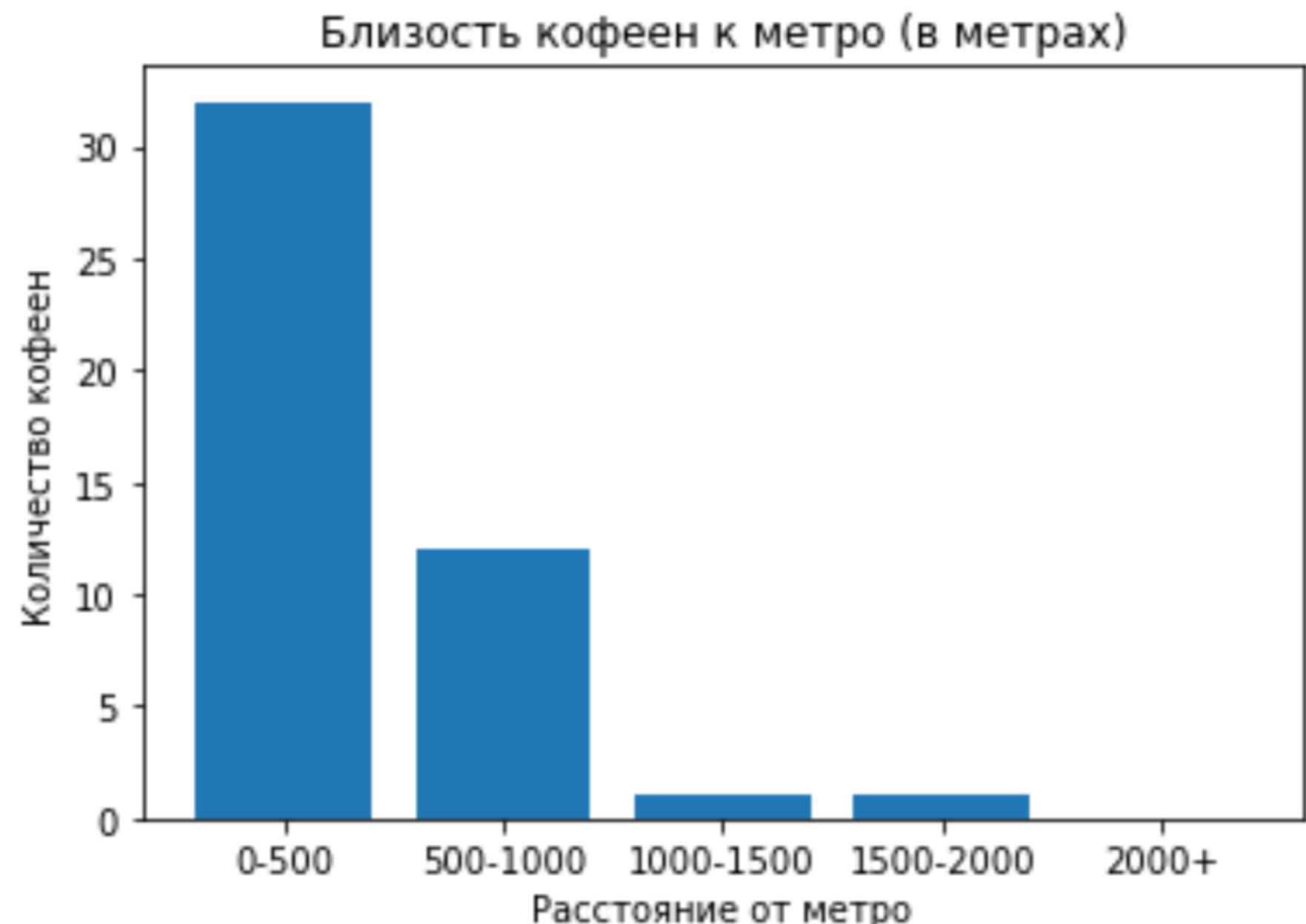
1 data2_0={}
2 for x in coffee_df['Метров до метро']:
3     x = min(x)
4     if x not in data2_0:
5         data2_0[int(x)] = 1
6     else:
7         data2_0[int(x)] += 1
8
9 data2 = {'0-500' : 0, '500-1000' : 0, '1000-1500' : 0,
10      '1500-2000' : 0, '2000+' : 0}
11
12 for k, v in data2_0.items():
13     if k <= 500:
14         data2['0-500'] += v
15     if k > 500 and k <= 1000:
16         data2['500-1000'] += v
17     if k > 1000 and k <= 1500:
18         data2['1000-1500'] += v
19     if k > 1500 and k <= 2000:
20         data2['1500-2000'] += v
21     if k > 2000:
22         data2['2000+'] += v

```

```

1 plt.bar(data2.keys(), data2.values())
2
3 plt.title("Близость кофеен к метро (в метрах")
4
5 plt.xlabel('Расстояние от метро')
6 plt.ylabel('Количество кофеен')
7
8 plt.show()

```



```
1 coffee_df2 = coffee_df['Средний чек']
2 coffee_df2 = pd.DataFrame(coffee_df2)
3 coffee_df2 = coffee_df2.dropna(axis=0, how=
4 print('До исключения:', len(coffee_df))
5 print('После исключения:', len(coffee_df2))
6
7 data4={}
8 for i in coffee_df2['Средний чек']:
9     if i not in data4:
10         data4[i] = 1
11     else:
12         data4[i] += 1
```

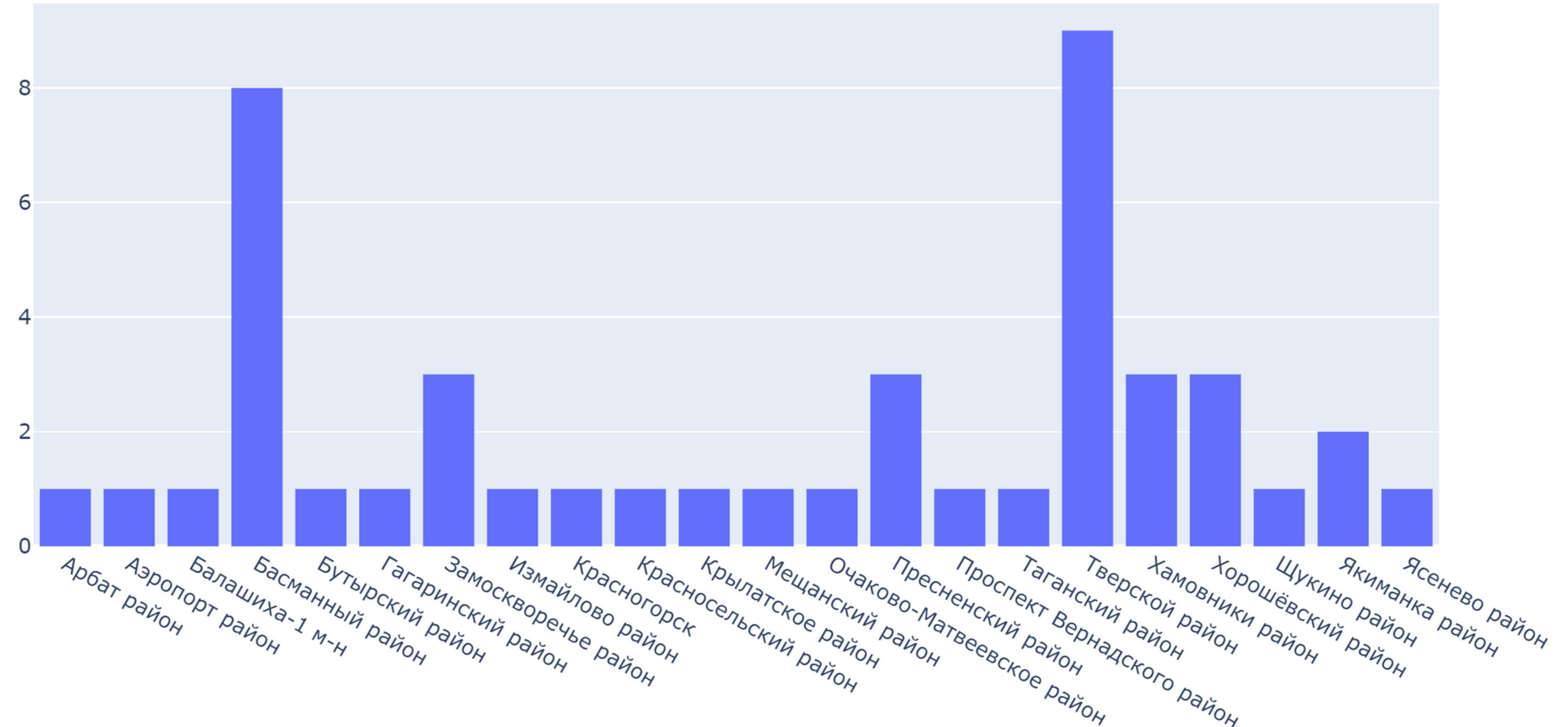
До исключения: 46

После исключения: 30

```
1 fig, ax1 = plt.subplots()
2 ax1.pie(data4.values(), labels=data4.keys()
3 plt.title("Средний чек кофеен")
4 plt.show()
```



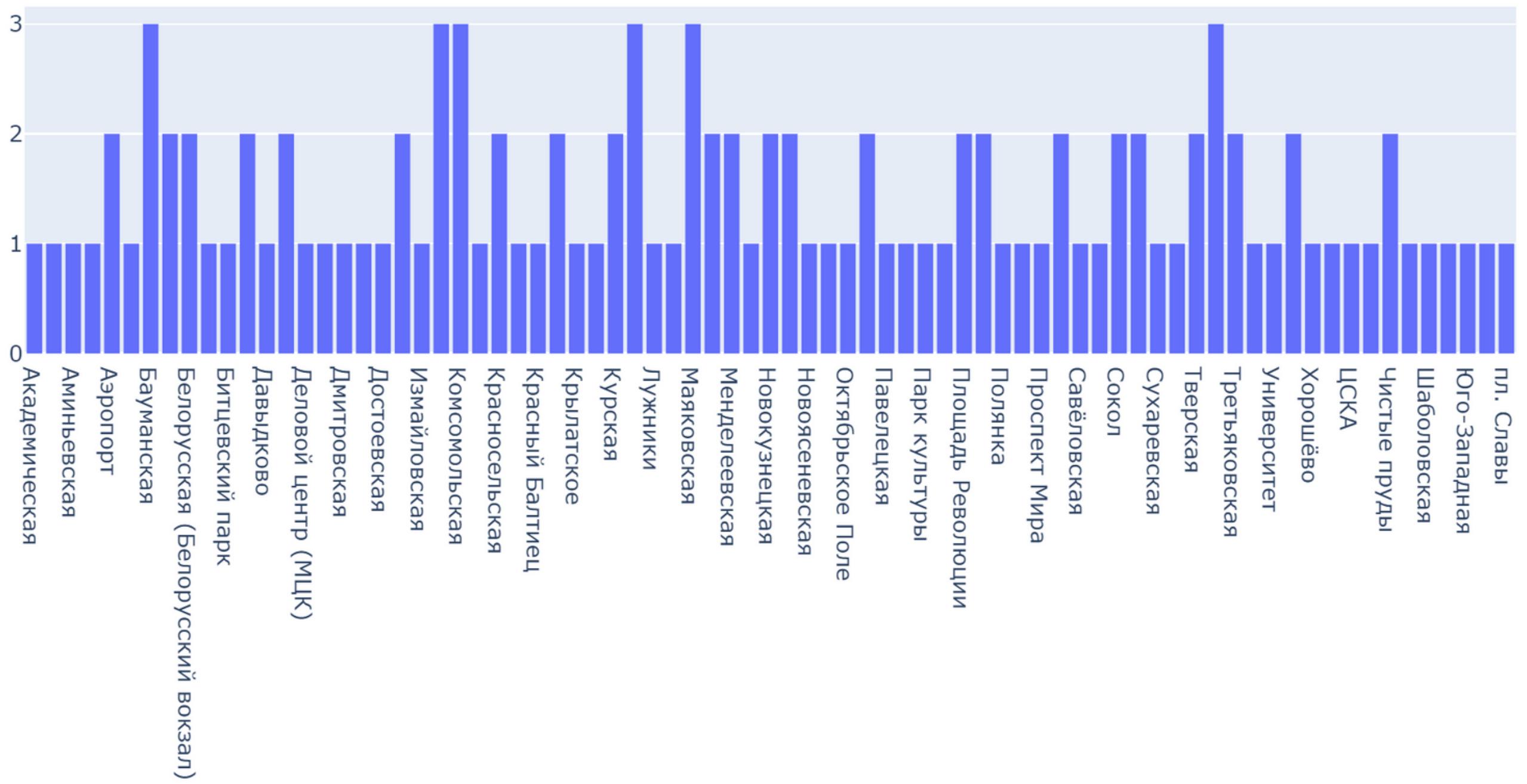
```
1 fig = go.Figure(data = [go.Histogram(x = sorted(coffee_df['Район']))])
2 fig.show()
```



```

1 allMetro = []
2 for i in coffee_df['Метро']:
3     setMet = set(i)
4     allMetro.extend(list(setMet))
5
6 fig = go.Figure(data = [go.Histogram(x = sorted(allMetro))])
7 fig.show()

```



Спасибо за внимание!