Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНЖЕНЕРНЫЙ БИЗНЕС И МЕНЕДЖМЕНТ»

КАФЕДРА «ПРОМЫШЛЕННАЯ ЛОГИСТИКА» (ИБМ-3)

Домашнее задание

По дисциплине:

«Парадигмы и конструкции языков программирования»

Студент ИБМ3- 34Б _____ П. Я. Головастикова
(Подпись, дата)

Руководитель _____ Ю. Е. Гапанюк
(Подпись, дата)

2025 г.

Задание:

1. Осуществите разбор данных сайта с использованием библиотеки BeautifulSoup.
2. Сформируйте датасет табличного типа с использованием Python и сохраните датасет в формате CSV.
3. Проведите разведочный анализ данных для сформированного датасета.

```
4. import requests
   from bs4 import BeautifulSoup
   import pandas as pd
   import time
   import re
   from datetime import datetime
   import matplotlib.pyplot as plt
   import seaborn as sns

   class GitHubScraper:
       def __init__(self):
           self.session = requests.Session()
           self.session.headers.update({
               'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36'
           })

       def scrape_trending_repos(self, language='python', since='weekly'):
           url = f'https://github.com/trending/{language}?since={since}'

           try:
               response = self.session.get(url)
               response.raise_for_status()

               soup = BeautifulSoup(response.content, 'html.parser')
               repos = []
               repo_elements = soup.find_all('article', class_='Box-row')

               for repo in repo_elements:
                   repo_data = self._parse_repo_element(repo)
                   if repo_data:
                       repos.append(repo_data)
                   time.sleep(0.5)

               return repos

           except Exception as e:
               print(f"Ошибка при парсинге: {e}")
               return []

       def _parse_repo_element(self, repo_element):
           try:
               title_element = repo_element.find('h2', class_='h3')
               if not title_element:
                   return None

               full_name =
   title_element.get_text(strip=True).replace('\n', '').replace(' ', '')
               author, name = full_name.split('/') if '/' in full_name
   else ('Unknown', full_name)
```

```
            desc_element = repo_element.find('p', class_='col-9')
            description = desc_element.get_text(strip=True) if
desc_element else 'No description'

            lang_element = repo_element.find('span',
itemprop='programmingLanguage')
            language = lang_element.get_text(strip=True) if
lang_element else 'Unknown'

            stars_element = repo_element.find('a',
href=re.compile(r'/stargazers'))
            stars =
self._parse_number(stars_element.get_text(strip=True) if stars_element
else '0')

            forks_element = repo_element.find('a',
href=re.compile(r'/forks'))
            forks =
self._parse_number(forks_element.get_text(strip=True) if forks_element
else '0')

            period_stars_element = repo_element.find('span', class_='d-
inline-block')
            period_stars =
self._parse_number(period_stars_element.get_text(strip=True).split()[0]
                                                if period_stars_element
else '0')

            return {
                'author': author,
                'name': name,
                'full_name': full_name,
                'description': description,
                'language': language,
                'stars': stars,
                'forks': forks,
                'period_stars': period_stars,
                'scraped_at': datetime.now().strftime('%Y-%m-%d
%H:%M:%S')
            }

        except Exception as e:
            print(f"Ошибка при парсинге элемента: {e}")
            return None

    def _parse_number(self, text):
        try:
            text = text.lower().replace(',', '')
            if 'k' in text:
                return int(float(text.replace('k', '')) * 1000)
            return int(text)
        except:
            return 0


scraper = GitHubScraper()

languages = ['python', 'javascript', 'java', 'go', 'rust']
all_repos = []

for lang in languages:
    print(f"Собираем данные для {lang}...")
    repos = scraper.scrape_trending_repos(language=lang)
```

```
        all_repos.extend(repos)
        time.sleep(2)

    print(f"Собрано {len(all_repos)} репозиториев")
```

В первой части кода происходит разбор данных с сайта Github с помощью библиотеки BeautifulSoup. Программа делает парсинг трендовых репризиториев, сортируя по языкам программирования. Внутри каждого репризитория смотрим имя владельца, его описание, язык программирования, звезды и форки, звезды за период.

```
 df = pd.DataFrame(all_repos)

csv_filename = 'github_trending_repos.csv'
df.to_csv(csv_filename, index=False, encoding='utf-8')
```

В этой части кода мы создаем датафрейм, который сохраняем в cvs файл.

```
numeric_cols = ['stars', 'forks', 'period_stars']
plt.style.use('default')
sns.set_palette("husl")

fig, axes = plt.subplots(2, 3, figsize=(18, 12))
fig.suptitle('Анализ Trending репозиториев GitHub', fontsize=16,
fontweight='bold')

axes[0, 0].hist(df['stars'], bins=20, edgecolor='black', alpha=0.7)
axes[0, 0].set_title('Распределение количества звезд')
axes[0, 0].set_xlabel('Звезды')
axes[0, 0].set_ylabel('Частота')

axes[0, 1].hist(df['forks'], bins=20, edgecolor='black', alpha=0.7)
axes[0, 1].set_title('Распределение количества форков')
axes[0, 1].set_xlabel('Форки')
axes[0, 1].set_ylabel('Частота')

top_languages = df['language'].value_counts().head(10)
axes[0, 2].bar(top_languages.index, top_languages.values)
axes[0, 2].set_title('Топ 10 языков программирования')
axes[0, 2].tick_params(axis='x', rotation=45)

axes[1, 0].scatter(df['stars'], df['forks'], alpha=0.6)
axes[1, 0].set_title('Соотношение звезд и форков')
axes[1, 0].set_xlabel('Звезды')
axes[1, 0].set_ylabel('Форки')

language_stats = df.groupby('language')[['stars',
'forks']].mean().sort_values('stars', ascending=False).head(10)
language_stats.plot(kind='bar', ax=axes[1, 1])
axes[1, 1].set_title('Средние звезды и форки по языкам')
axes[1, 1].tick_params(axis='x', rotation=45)

correlation_matrix = df[numeric_cols].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0,
ax=axes[1, 2])
axes[1, 2].set_title('Корреляционная матрица')

plt.tight_layout()
plt.show()

print("\n7. АНАЛИЗ ВЫБРОСОВ:")
Q1 = df['stars'].quantile(0.25)
```

```
Q3 = df['stars'].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df['stars'] < (Q1 - 1.5 * IQR)) | (df['stars'] > (Q3 + 1.5 *
IQR))]
print(f"Количество выбросов по звездам: {len(outliers)}")
```

Дальше я провела разведочный анализ данных, для каждого пункта реализовав графическое представление. Первые две гистограммы отвечают за количество распределения звезд и форков. Следующая гистограмма показывает тир-лист популярных языков программирования. Нижние графики отражают соотношения звезд и форков. А последний график реализует корреляционную матрицу (взаимосвязь разных компонентов в наборе данных).