

# Отчёт по практической работе №6

## Работа с облачными системами и визуализации данных

### Цель работы:

Приобрести практические навыки в области создания интеллектуальных систем с использованием современных средств разработки.

### Ход работы:

Снова откроем Google Colab, но теперь создадим полностью новый блокнот и скопируем код из методички, сразу отформатируем текст для улучшения читаемости по аналогии с моими прошлыми работами.

```
import folium
print('Folium installed and imported!')

# Определим карту мира
world_map = folium.Map()

# Отобразим карту мира
world_map

# Определим карту мира, центрированную на Канаде с низким уровнем масштабирования
world_map = folium.Map(location=[56.130, -106.35], zoom_start=4)

# Отобразим карту мира
world_map

# Создаем карту Stamen Toner мира, центрированную на Канаде
world_map = folium.Map(location=[56.130, -106.35], zoom_start=4, tiles='OpenStreetMap')

# Отобразим карту
world_map

# Создаем карту Stamen Toner мира, центрированную на Канаде
world_map = folium.Map(location=[56.130, -106.35], zoom_start=4, tiles='cartodbpositron')

# Отобразим карту
world_map

# Создаем карту мира с Mapbox Bright стилом
world_map = folium.Map(tiles='Cartodb dark_matter')

# Отобразим карту
world_map

# Загружаем данные о преступлениях
df_incidents = pd.read_csv('https://github.com/shihao-wen/TDM-Data-Science-Professional-Certificate/blob/master/6.120Data320Visualization/Final320Assignment/Police_Department_Incidents_-_Previous_Year_2016.csv?raw=true')

print('Dataset downloaded and read into a pandas dataframe!')

# Показываем первые 5 строк данных
df_incidents.head()

# Получаем первые 100 преступлений из dataframe df_incidents
limit = 100
df_incidents = df_incidents.iloc[0:limit, :]

# Координаты Сан-Франциско (широта и долгота)
latitude = 37.77
longitude = -122.42

# Создаем карту и отображаем её
sanfran_map = folium.Map(location=[latitude, longitude], zoom_start=12)
incidents = folium.map.FeatureGroup()

# Проходим через первые 100 преступлений и добавляем каждое на карту
for lat, lng in zip(df_incidents.Y, df_incidents.X):
    incidents.add_child(
        folium.features.CircleMarker(
            [lat, lng],
            radius=5, # определен размер кругов на карте
            color='yellow',
            fill=True,
            fill_color='blue',
            fill_opacity=0.6
        )
    )

# Добавляем инциденты на карту
sanfran_map.add_child(incidents)
```

Рис 6.1.

После этих действий мне выдало ошибку 404, в ссылке на данные с GitHub была допущена ошибка, после исправления получим интерактивную карту

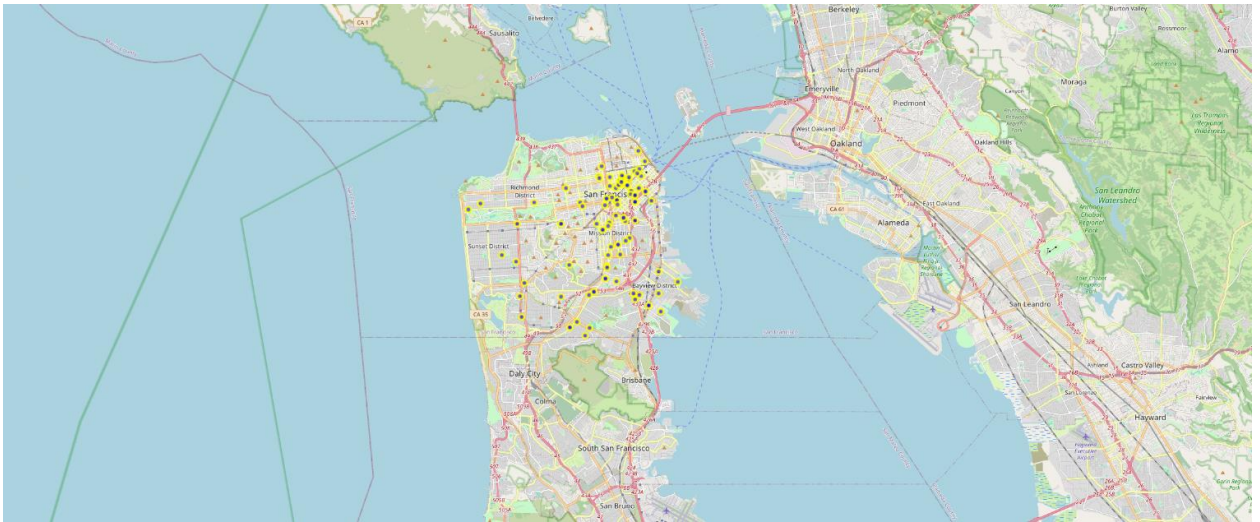


Рис 6.2.

### **Вывод:**

В ходе работы мы приобрели навык в отображении интерактивных карт, что позволяем нам отображать инциденты прямо на ней, что очевидно имеет преимущество по сравнению с графиками.

## Отчёт по практической работе №5

### Визуализация данных средствами Matplotlib. Диаграммы

#### Цель работы:

Получить навыки использования библиотеки визуализации данных Matplotlib с использованием языка Python.

#### Ход работы:

Зайдём по ссылке из методички и откроем четвёртую практическую работу, дополним её кодом из методички. Снова отформатируем код для улучшения читаемости.

```
# Проверяем данные для 2013 года
df_can['2013'].head()

# np.histogram возвращает два значения
count, bin_edges = np.histogram(df_can['2013'])

# Выводим частоту появления данных
print(count) # подсчет частоты появления данных

# Выводим количество столбцов, по умолчанию - 10
print(bin_edges)

# Строим гистограмму для данных за 2013 год
df_can['2013'].plot(kind='hist', figsize=(8, 5))

# Добавляем название графика
plt.title('Histogram of Immigration from 195 Countries in 2013')

# Добавляем наименование оси Y
plt.ylabel('Number of Countries')

# Добавляем наименование оси X
plt.xlabel('Number of Immigrants')

# Отображаем график
plt.show()

# Шаг 1: получаем данные для Исландии
df_iceland = df_can.loc['Iceland', years]
df_iceland.head()

# Шаг 2: строим горизонтальную столбчатую диаграмму
df_iceland.plot(kind='barh', figsize=(10, 6))

# Добавляем наименование оси X
plt.xlabel('Year')

# Добавляем наименование оси Y
plt.ylabel('Number of Immigrants')

# Добавляем название графика
plt.title('Icelandic immigrants to Canada from 1980 to 2013')

# Отображаем график
plt.show()
```

Рис 5.1.

Код сразу же не выдал ошибок и построил диаграммы.

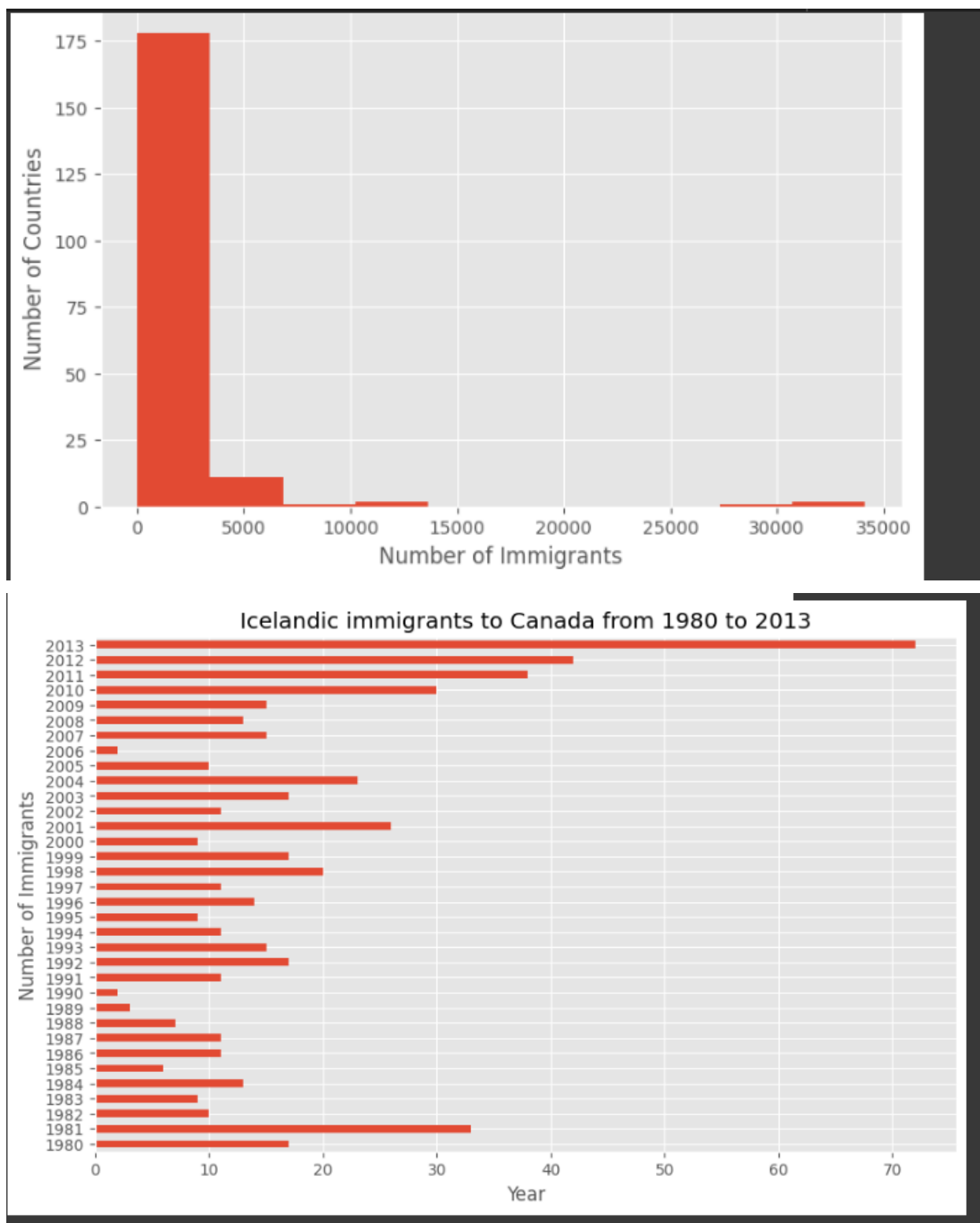


Рис 5.2.

Поменяем параметр `kind` на значение `bar`. Отображение получилось не правильным из-за большого количества надписей, которые вытеснили собой график

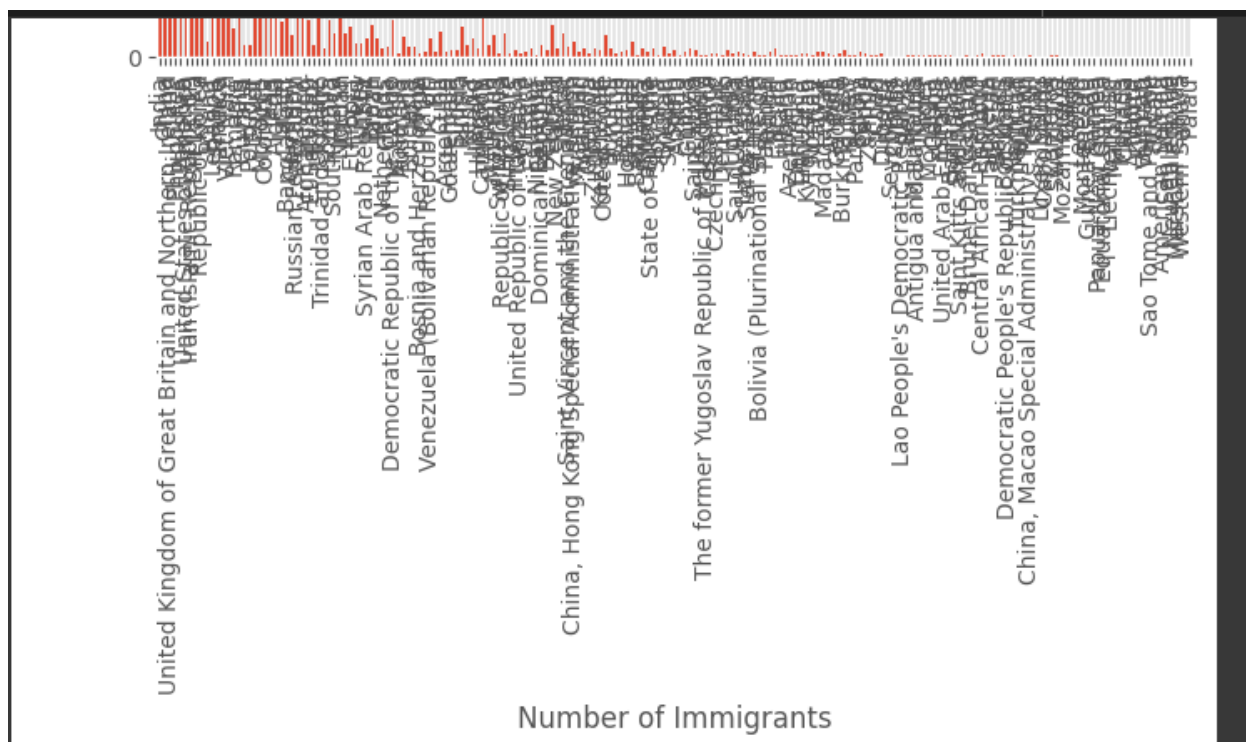


Рис 5.3.

## Вывод:

В данной работе мы получили опыт работы с диаграммами и получили информацию о значениях параметра “kind”. Это очень удобно для графического отображения почти любой информации, в частности подходят очень многие статистические данные, к тому же это сильно облегчает их представления другим людям.

## Отчёт по практической работе №4

### Визуализация данных средствами Matplotlib. Основы

#### Цель работы:

Получить навыки использования библиотеки визуализации данных Matplotlib с использованием языка Python.

#### Ход работы:

Согласно указаниям в методичке создадим блокнот на сайте Google Colab

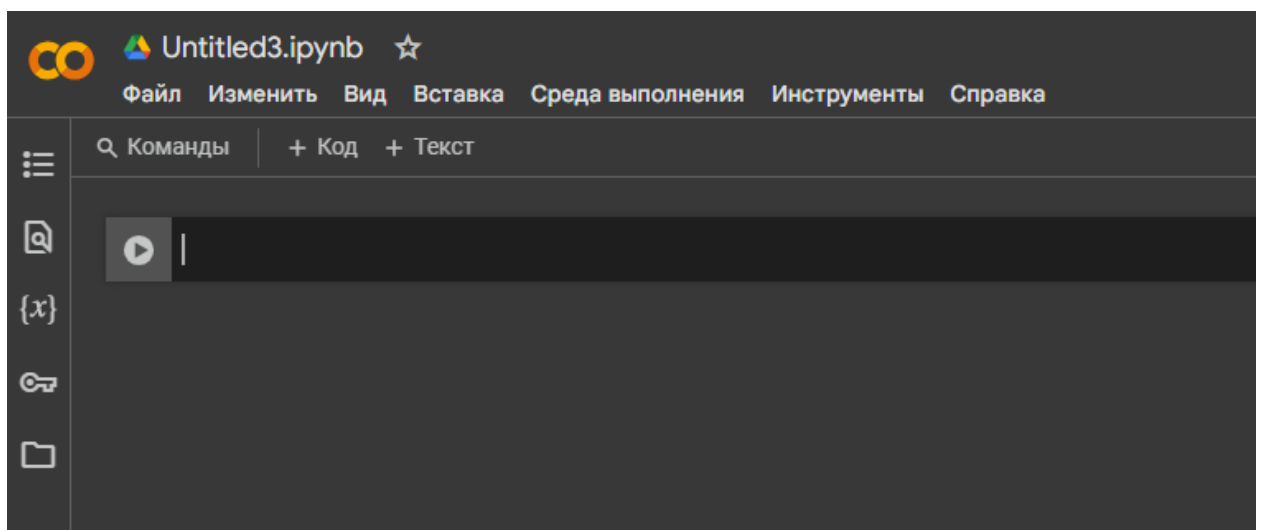


Рис 4.1.

Скопируем код из методички для проверки работы

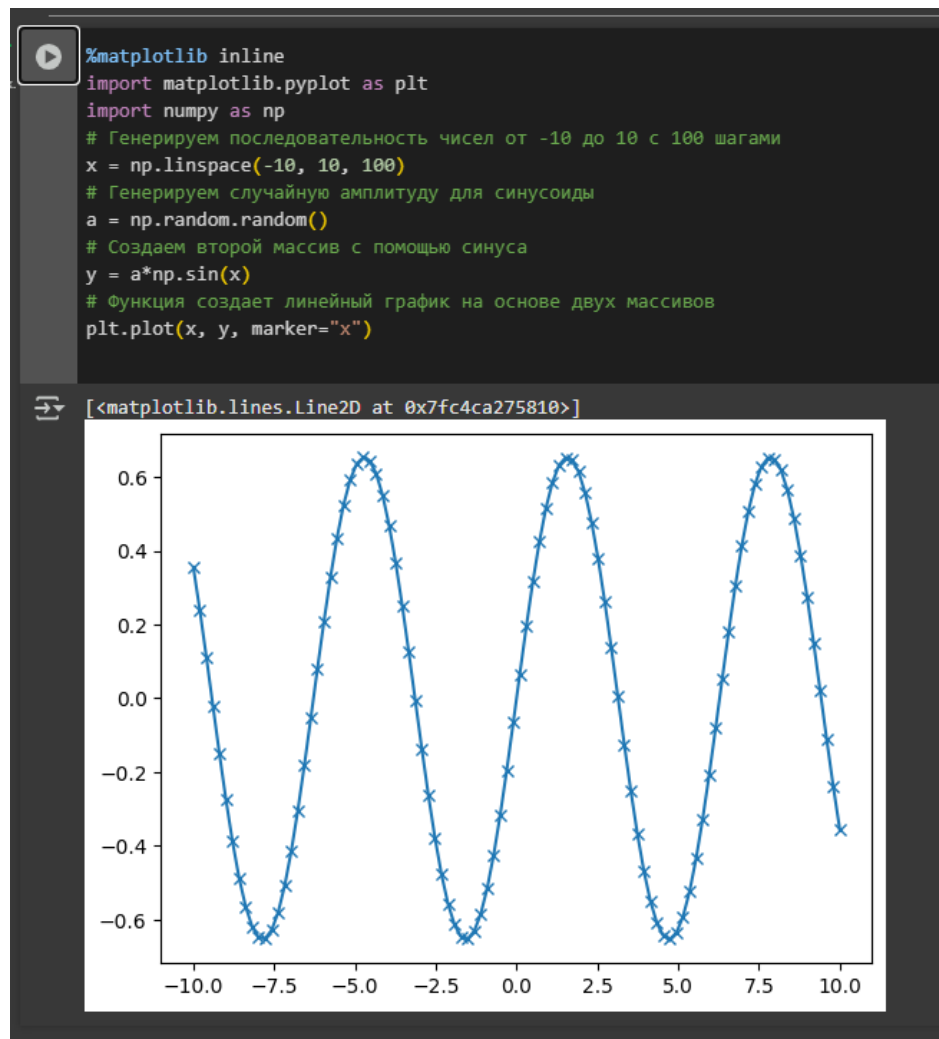


Рис 4.2.

После того как график выведен и отработал правильно перейдём к основной части. После копирования кода он выдал ошибку поэтому проверим код и отформатируем его, добавив комментарии

```

# Загружаем данные из Excel
df_can = pd.read_excel(
    'https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DV0101EN/labs/Data_Files/Canada.xlsx',
    sheet_name='Canada by Citizenship',
    skiprows=range(20),
    skipfooter=2
)
print('Данные загружены и записаны в dataframe!')

# Выводим первые строки таблицы
df_can.head()

# Выводим размерность таблицы
print(df_can.shape)

# Удаляем лишние столбцы
df_can.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis=1, inplace=True)
df_can.head()

# Переименовываем столбцы
df_can.rename(columns={'OdName': 'Country', 'AreaName': 'Continent', 'RegName': 'Region'}, inplace=True)
df_can.head()

# Проверим, все ли столбцы строковые
all(isinstance(column, str) for column in df_can.columns)

# Преобразуем столбцы в строки
df_can.columns = list(map(str, df_can.columns))

# Проверим снова
all(isinstance(column, str) for column in df_can.columns)

# Устанавливаем индекс по стране
df_can.set_index('Country', inplace=True)
df_can.head()

# Добавляем столбец Total с суммой по строкам
# Преобразуем все данные в числовые, где возможно (нечисловые заменятся на NaN)
df_can[years] = df_can[years].apply(pd.to_numeric, errors='coerce')

# Теперь можно добавить столбец Total с суммой по строкам
df_can['Total'] = df_can[years].sum(axis=1)
df_can.head()

df_can.head()

# Создаём список годов для сортировки
years = list(map(str, range(1980, 2014)))

# Сортируем данные по столбцу Total
df_can.sort_values(['Total'], ascending=False, axis=0, inplace=True)
df_top5 = df_can.head()

# Транспонируем таблицу для топ-5 стран
df_top5 = df_top5[years].transpose()
df_top5.head()

# Устанавливаем стиль графика ggplot
mpl.style.use('ggplot')

# Проверим версию Matplotlib
print('Matplotlib version: ', mpl.__version__) # >= 2.0.0

```

Рис 4.3

Код выдавал ошибки поэтому были внесены изменения

```

# Преобразуем все данные в числовые, где возможно (нечисловые заменятся на NaN)
df_can[years] = df_can[years].apply(pd.to_numeric, errors='coerce')

# Теперь можно добавить столбец Total с суммой по строкам
df_can['Total'] = df_can[years].sum(axis=1)
df_can.head()

```

Рис 4.4.

После этих изменений получили вот такой график



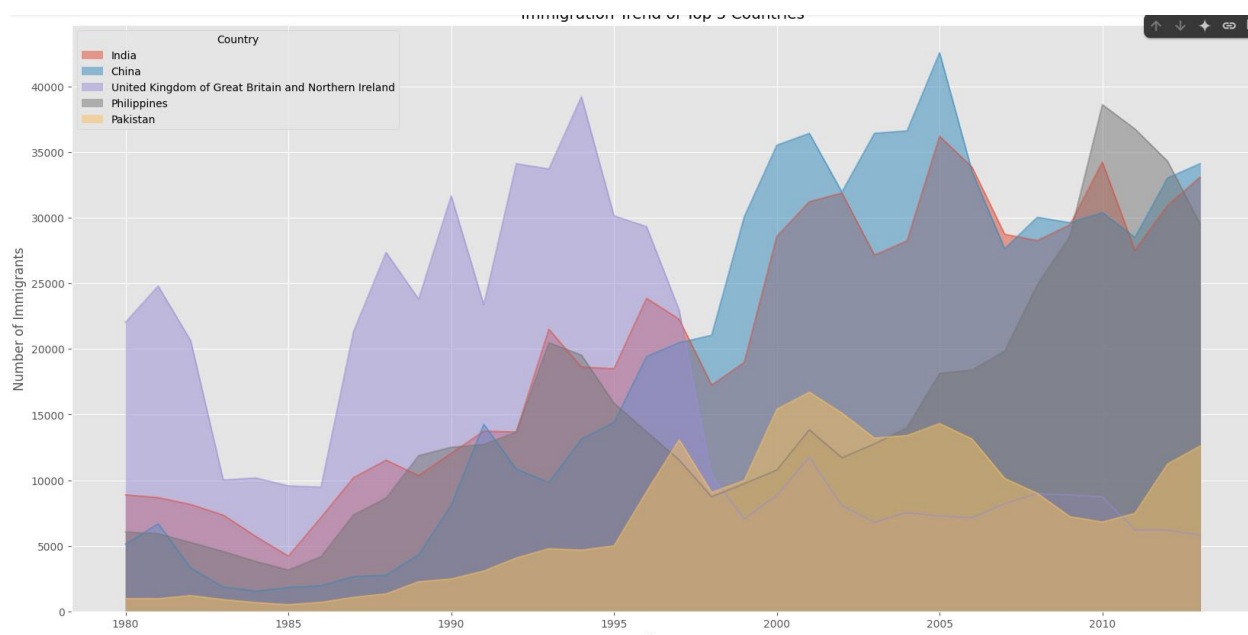


Рис 4.5.

Так же сохраним данную работу так как она понадобится нам в работе №5

### **Вывод:**

В данной работе мы изучили основы построения графиков с помощью Matplotlib. Так же решили проблему с табуляциями и дополнили программу комментариями для улучшения читаемости текста.

## Отчёт по практической работе №3

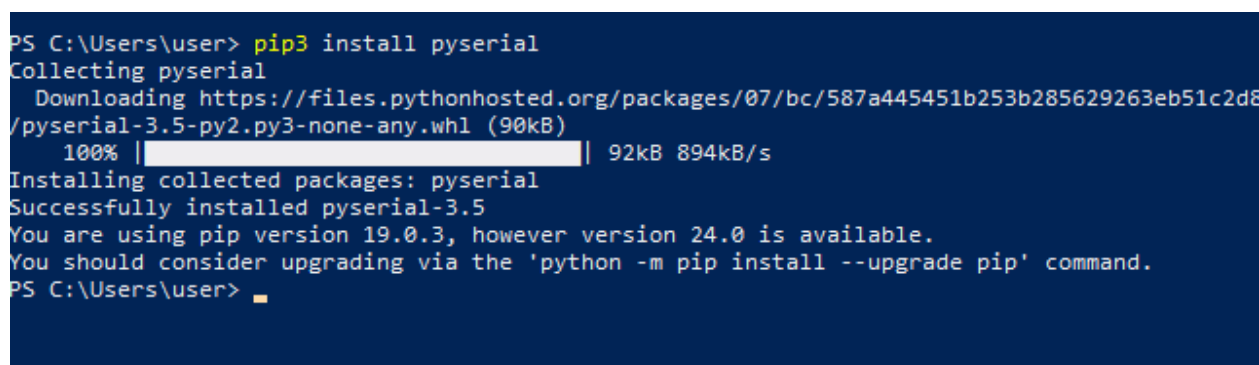
### Реализация программы с последовательным портом средствами Python

#### Цель работы:

Изучить приведённую программу и получить навыки использования последовательных портов

#### Ход работы:

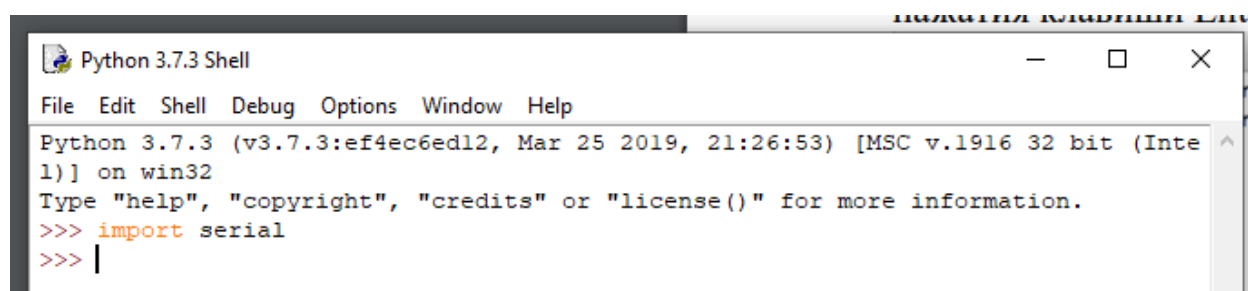
Сначала установим PySerial так как я уверен, что не устанавливал его ранее



```
PS C:\Users\user> pip3 install pyserial
Collecting pyserial
  Downloading https://files.pythonhosted.org/packages/07/bc/587a445451b253b285629263eb51c2d8
/pyserial-3.5-py2.py3-none-any.whl (90kB)
    100% |#####| 92kB 894kB/s
Installing collected packages: pyserial
Successfully installed pyserial-3.5
You are using pip version 19.0.3, however version 24.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\Users\user>
```

Рис 3.1.

Теперь введём `import serial` в idle

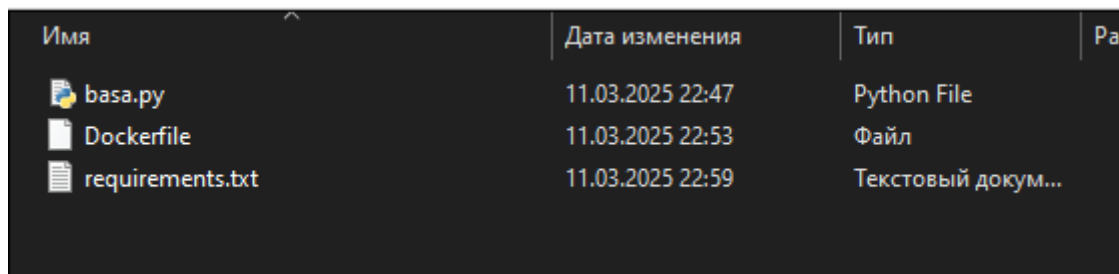


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import serial
>>> |
```

Рис 3.2.

Так приведённый в методичке код для python отформатируем его добавив комментарии и расставим отступы. Код всё равно выдал ошибку из-за отсутствия портов. Предполагаю, что их можно создать при помощи специальных инструментов.

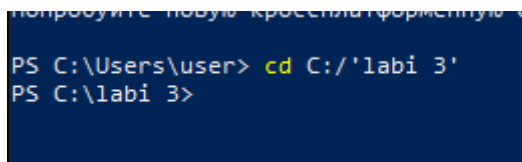
Перейдём к созданию Docker с нашей программой. Сохраним нашу программу с расширением .py и создадим докерфайл без расширения и requirements.txt



Имя	Дата изменения	Тип	Pa
basa.py	11.03.2025 22:47	Python File	
Dockerfile	11.03.2025 22:53	Файл	
requirements.txt	11.03.2025 22:59	Текстовый докум...	

Рис .3.3.

Далее перейдём в Power Shell и зайдём в папку проекта



```
PS C:\Users\user> cd C:\'labi 3'  
PS C:\labi 3>
```

Рис 3.4.

Далее по аналогии с практической работой 2 создаём Docker Container, однако он не запустится из-за неработающей программы.

**Вывод:**

В ходе работы мы изучили работу последовательных портов инструменты их симуляции и поместили программу в Docker Container

## Отчёт по практической работе №2

### Основы работы с технологиями контейнеризации и ботами Telegram

#### Цель работы:

Создание сервера с постоянно работающим Telegramm ботом

#### Ход работы:

Для начала откроем PowerShell и подключимся к серверу шлюзу и через него к рабочему серверу

```
PS C:\Users\user> ssh student@193.124.118.93
The authenticity of host '193.124.118.93 (193.124.118.93)' can't be established.
ED25519 key fingerprint is SHA256:9YfF0RJ043svji7MPRcbpG4Jp3k2f7tVmDEavUEQwYQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '193.124.118.93' (ED25519) to the list of known hosts.
student@193.124.118.93's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.2.0-1015-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Mon Mar 10 13:26:01 2025 from 85.249.163.217
student@ruvds-x7i06:~$ ssh student@10.8.0.5
student@10.8.0.5's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

12 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
*** System restart required ***
Last login: Mon Mar 10 16:26:10 2025 from 10.8.0.1
```

Рис 2.1

Создадим директорию по номеру зачётной книжки и перейдём в неё.

Установим Python необходимой версии и создадим окружение

```
student@user-IPMSB-H61:~$ mkdir 220803148 && cd 220803148
student@user-IPMSB-H61:~/220803148$ python3.10
Python 3.10.16 (main, Dec 4 2024, 08:53:37) [GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
student@user-IPMSB-H61:~/220803148$ python3.10 -m venv env
student@user-IPMSB-H61:~/220803148$ source env/bin/activate
(env) student@user-IPMSB-H61:~/220803148$ pip install telepot==12.7
```

Рис 2.2.

Создадим bot.py и запишем в него код из методички параллельно создадим самого бота в телеграмм и получим токен, после чего запустим его.

```
(env) student@user-IPMSB-H61:~/220803148$ nano bot.py
(env) student@user-IPMSB-H61:~/220803148$ nano bot.py
(env) student@user-IPMSB-H61:~/220803148$ python bot.py
File "/home/student/220803148/bot.py", line 9
    bot.sendMessage(chat_id, 'Oks')
    ^
IndentationError: expected an indented block after 'if' statement on line 8
(env) student@user-IPMSB-H61:~/220803148$ nano bot.py
(env) student@user-IPMSB-H61:~/220803148$ python bot.py
Traceback (most recent call last):
  File "/home/student/220803148/bot.py", line 8, in <module>
    if command == '/command1':
NameError: name 'command' is not defined
(env) student@user-IPMSB-H61:~/220803148$ nano bot.py
(env) student@user-IPMSB-H61:~/220803148$ (env) student@user-IPMSB-H61:~/220803148$ python bot.py
I am listening ...
Got command: /start
From : 985955237
Got command: /command1
From : 985955237
Got command: /command2
From : 985955237
```

Рис 2.3.

Зайдём в телеграмм найдём бота и протестируем его.



Рис 2.4.

После того что мы увидим что бот работает создадим docker

```
(env) student@user-IPMSB-H61:~/220803148$ deactivate
deactivate: command not found
(env) student@user-IPMSB-H61:~/220803148$ deactivate
student@user-IPMSB-H61:~/220803148$ nano requirements.txt
student@user-IPMSB-H61:~/220803148$ nano Dockerfile
student@user-IPMSB-H61:~/220803148$ docker build -t 220803148
ERROR: docker: 'docker buildx build' requires 1 argument

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Run 'docker buildx build --help' for more information
[+] Building 11.1s (13/13) FINISHED                                docker:default
-> [internal] load build definition from Dockerfile                0.7s
-> => transferring dockerfile: 205B                                0.0s
-> [internal] load metadata for docker.io/library/python:3.10     2.3s
-> [internal] load metadata for docker.io/library/python:3.10-slim 2.2s
-> [internal] load -dockerignore                                   0.7s
-> => transferring context: 2B                                       0.0s
-> [builder 1/3] FROM docker.io/library/python:3.10@sha256:c79cd7b345b44821-0dee8cd8d8e314450aac30ea0f7000f7a7152 0.0s
-> [internal] load build context                                   0.7s
-> => transferring context: 505B                                       0.0s
-> [stage-1 1/4] FROM docker.io/library/python:3.10-slim@sha256:f000fc3f447306d9be2ae53dc7ab447fe9b33113af309125 0.0s
-> CACHED [stage-1 2/4] WORKDIR /code                             0.0s
-> CACHED [builder 2/3] COPY requirements.txt .                    0.0s
-> CACHED [builder 3/3] RUN pip install --user -r requirements.txt 0.0s
-> CACHED [stage-1 3/4] COPY --from=builder /root/.local /root/.local 0.0s
-> [stage-1 4/4] COPY ./bot.py .                                    1.7s
-> exporting to image                                              1.0s
-> => exporting layers                                              1.2s
-> => writing image sha256:f0b7f0daec3f003c5c77a4f0071a10048100f0b1c1125f014a00f11104ac16a0 0.1s
-> => pushing to docker.io/library/220803148                       0.1s
```

Рис 2.5.

Запустим docker образ и получаем container ID

```
student@user-IPMSB-H61:~/220803148$ docker run -d --restart=always 220803148
5feab6795db1749ce619abb30793513244116f79fd5daa7ac0c43d9dd839acf3
student@user-IPMSB-H61:~/220803148$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED
5feab6795db1   220803148     "python -u ./bot.py"    About a minute ago
```

Рис 2.6.

Далее сохраним архив docker образа на ПК. Здесь удобно использовать команду `rwd` которая выведет нам путь в текущую папку.

```
student@ruvds-x7106:~$ ssh student@10.8.0.5
student@10.8.0.5's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

12 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
*** System restart required ***
Last login: Mon Mar 10 21:43:55 2025 from 10.8.0.1
student@user-IPMSB-H61:~$ cd 220803148
student@user-IPMSB-H61:~/220803148$ ls
bot.py Dockerfile docker_image_220803148.tar env requirements.txt
student@user-IPMSB-H61:~/220803148$ pwd
/home/student/220803148
student@user-IPMSB-H61:~/220803148$ exit
logout
Connection to 10.8.0.5 closed.
student@ruvds-x7106:~$ scp student@10.8.0.5:/home/student/220803148/docker_image_220803148.tar .
student@10.8.0.5's password:
docker_image_220803148.tar                                100% 138MB  5.8MB/s   00:23
student@ruvds-x7106:~$ pwd
/home/student
student@ruvds-x7106:~$ exit
logout
Connection to 193.124.118.93 closed.
PS C:\Users\user> scp student@193.124.118.93:/home/student/docker_image_220803148.tar .
student@193.124.118.93's password:
docker_image_220803148.tar                                100% 138MB  1.2MB/s   02:00
PS C:\Users\user> ls
```

Рис 2.7.

Откроем архив для просмотра скачанных файлов

Имя	Размер	Сжат	Тип	Изменён	CRC32
..			Папка с файлами		
blobs	144 782 655	144 782 655	Папка с файлами	10.03.2025 21:24	
index.json	366	366	JSON File	01.01.1970 3:00	
manifest.json	2 296	2 296	JSON File	01.01.1970 3:00	
oci-layout	31	31	Файл	01.01.1970 3:00	
repositories	92	92	Файл	01.01.1970 3:00	

Рис 2.8.

**Вывод:**

В ходе выполненной работы мы создали постоянно работающего Телеграмм бота и сохранили его в docker container, также получили опыт взаимодействия с серверами через Power Shell.



# Отчёт по практической работе №1

## Основы Git и GitHub

### Цель работы:

Выполнение практической работы направлено на изучение: 1. наиболее распространенных практик в области контроля версий программного обеспечения, его использования в командной разработке ПО и DevOps; 2. концепции Git, основанной на понятиях репозитория и ветвления версий ПО; 3. порядка использования GitHub и его базовых операций.

### Ход работы:

Для начала установим Git Bash из интернет

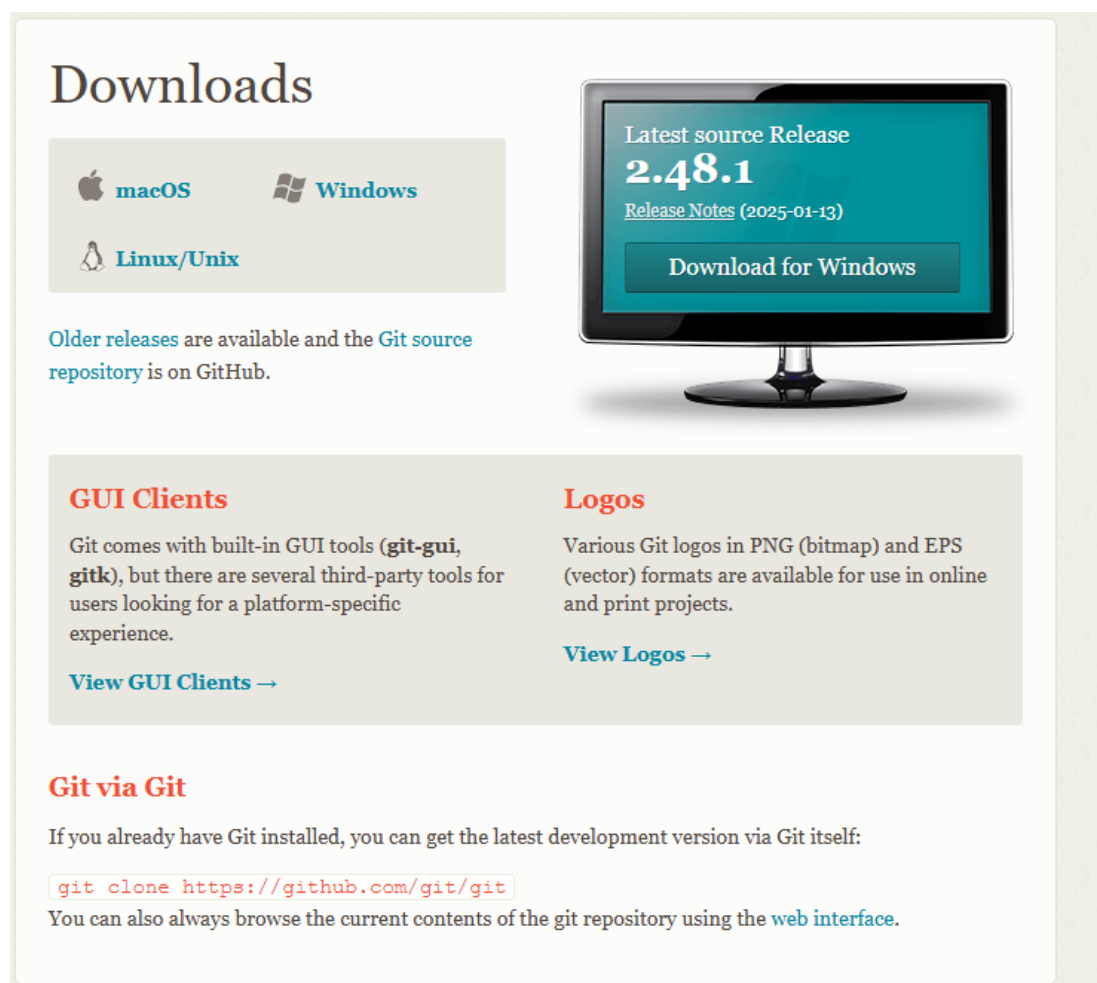


Рис 1.1.

Далее зайдём в папку, заранее созданную для материалов по дисциплине

```
user@DESKTOP-7K13101 MINGW64 ~  
$ cd D:  
  
user@DESKTOP-7K13101 MINGW64 /d/  
$ cd labi  
  
user@DESKTOP-7K13101 MINGW64 /d/labi/  
$ cd myrepo
```

Рис 1.2.

Создадим папку myrepo и сразу перейдём к созданию веток создадим согласно методичке ветку newbranch и переключимся на неё

```
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (master)  
$ git branch newbranch  
  
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (master)  
$ git branch  
* master  
  newbranch  
  
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (master)  
$ git checkout newbranch  
Switched to branch 'newbranch'  
  
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)  
$ git branch  
  master  
* newbranch
```

Рис 1.3.

Далее создадим новый файл в созданной ветке и сохраним изменения

```
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)  
$ touch newbranchfile  
  
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)  
$ git add newbranchfile  
  
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)  
$ git commit -m "Добавлен пустой файл newbranchfile"  
On branch newbranch  
nothing to commit, working tree clean
```

Рис 1.4.

Согласно заданию отменим последнее действие

```
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)
$ git revert HEAD --no-edit
[newbranch 00f150c] Revert "added newbranchfile"
Date: Mon Mar 10 19:13:10 2025 +0300
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 newbranchfile
```

Рис 1.5.

Создадим другой файл

```
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)
$ touch newgoodfile

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)
$ git add newgoodfile

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)
$ git commit -m "Добавлен новый файл newgoodfile"
[newbranch 76e64a5] Добавлен новый файл newgoodfile
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 newgoodfile
```

Рис 1.6.

Далее используем merge заранее переключив ветку на master

```
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)
$ git merge new-branch-name
merge: new-branch-name - not something we can merge

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)
$ git merge newbranch
Already up to date.

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (newbranch)
$ git checkout master
Switched to branch 'master'

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (master)
$ git merge newbranch
Updating fc34406..76e64a5
Fast-forward
 newbranchfile => newgoodfile | 0
1 file changed, 0 insertions(+), 0 deletions(-)
rename newbranchfile => newgoodfile (100%)
```

Рис 1.7.

На этом мы закончили первое задание методички. Для 2ого задания регистрируем второй аккаунт GitHub и выполняем fork репозитория из основного аккаунта

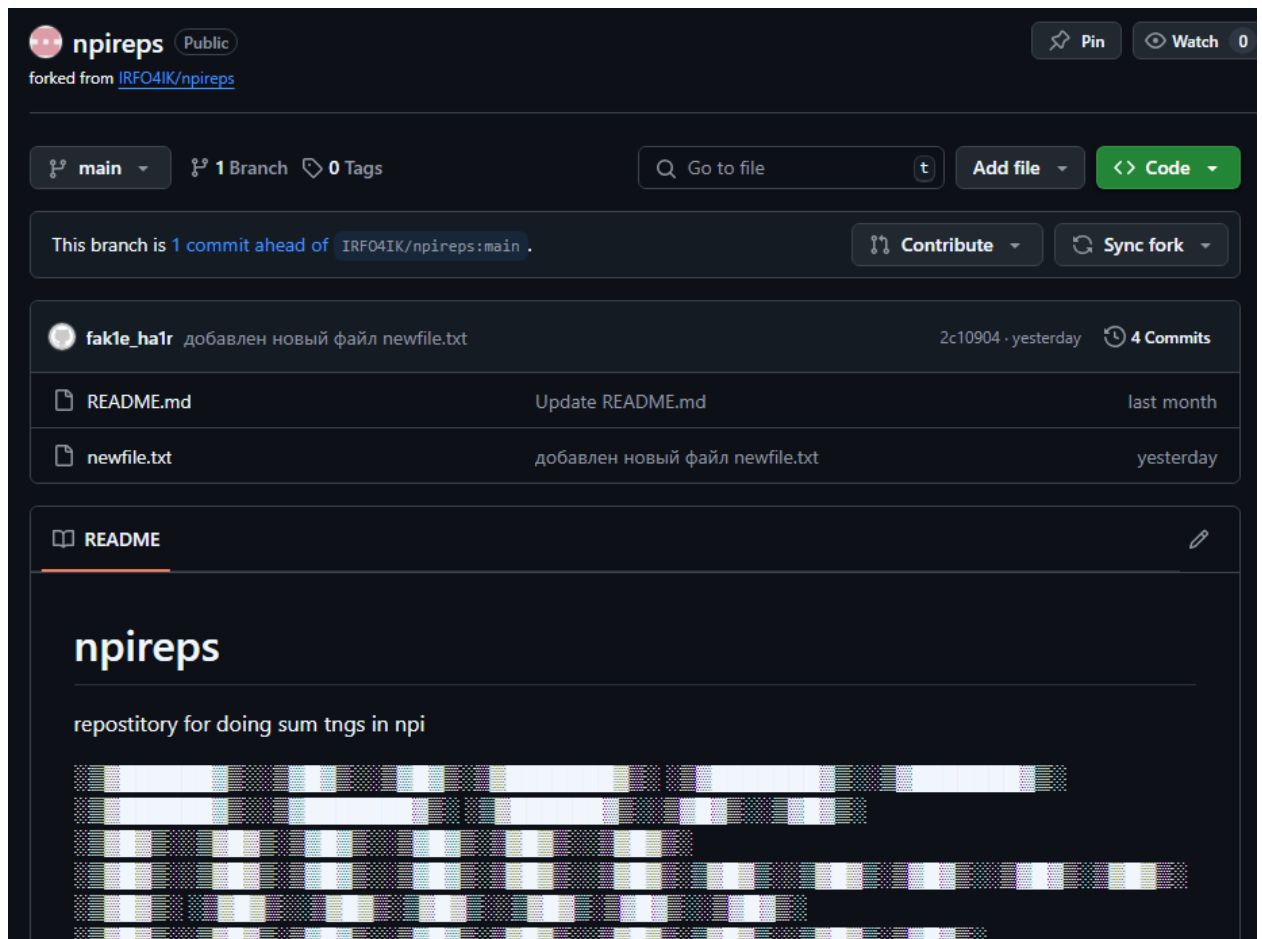


Рис 1.8.

Снова возвращаемся в Git Bush и создаём локальную копию fork проекта

```
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (master)
$ git clone https://github.com/IRFO4IK2/npireps.git
Cloning into 'npireps'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo (master)
$ cd npireps

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo/npireps (main)
$ touch newfile.txt

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo/npireps (main)
$ git add newfile.txt

user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo/npireps (main)
$ git commit -m "добавлен новый файл newfile.txt"
[main 2c10904] добавлен новый файл newfile.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 newfile.txt
```

Рис 1.9.

Далее создадим и сохраним файл по аналогии из первого задания

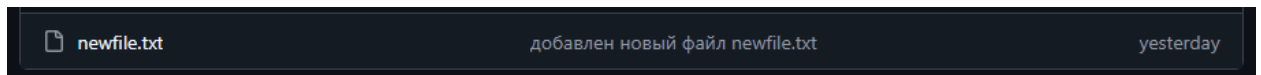


Рис 1.10.

Далее синхронизируем изменения на GitHub

```
user@DESKTOP-7K13101 MINGW64 /d/labi/myrepo/npireps (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/IRF04IK2/npireps.git
 330d996..2c10904  main -> main
```

Рис 1.11.

Далее создадим pull request

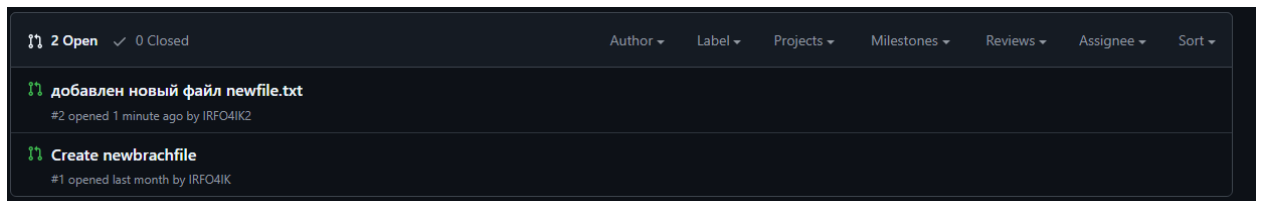


Рис1.12

## Вывод:

В ходе работы мы изучили основные команды Git Bush и смогли сохранить изменения в GitHub, что удобно для кооперации с другими разработчиками.