

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины
«Искусственный интеллект и машинное обучение»
Вариант 9

Выполнил:
Кравчук Мирослав Витальевич
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи», очная
форма обучения

(подпись)

Проверил:
Доцент департамента цифровых,
робототехнических систем и
электроники Воронкин Р.А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2025 г.

Тема: Работа с Jupyter Notebook, JupyterLab и Google Colab

Цель: исследовать базовые возможности интерактивных оболочек Jupyter Notebook, JupyterLab и Google Colab для языка программирования Python.

Ссылка на GitHub: <https://github.com/miron2314/DLlab-1.git>

Порядок выполнения работы:

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный язык программирования.

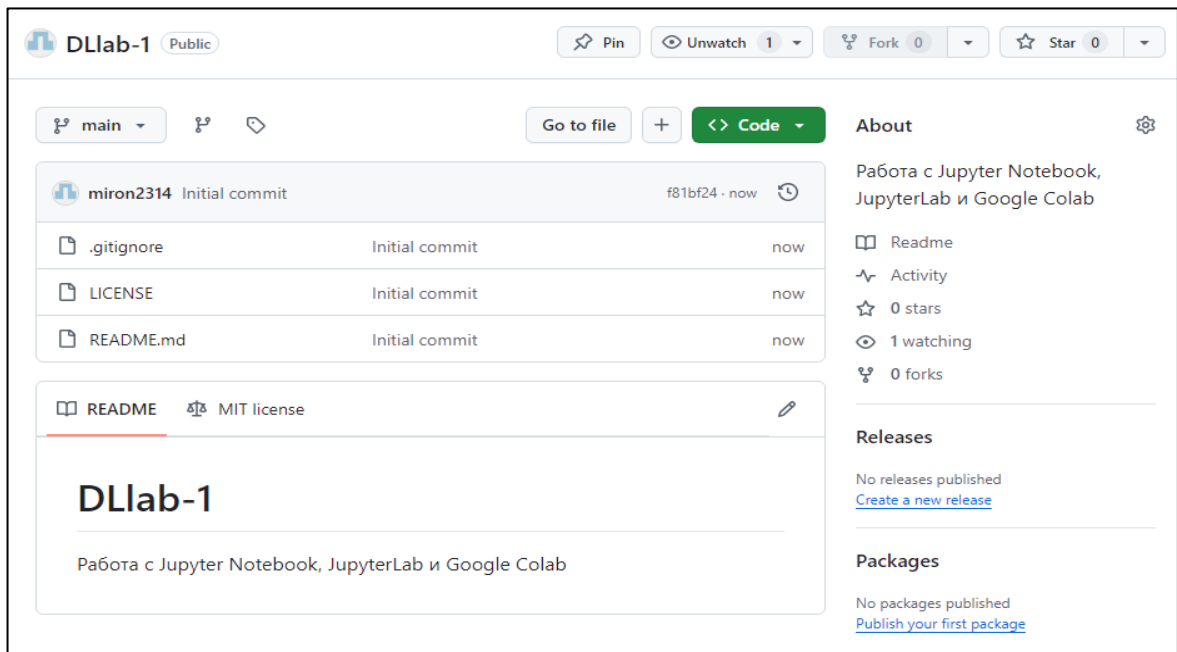


Рисунок 1. Репозиторий

3. Выполнил клонирование репозитория.

```
PS C:\Users\Student\source\repos> cd C:\Users\Student
PS C:\Users\Student> git clone https://github.com/miron2314/DLlab-1.git
Cloning into 'DLlab-1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
PS C:\Users\Student>
```

Рисунок 2. Клонирование

4.Проработал примеры работы с JupyterNotebook.

```
In [2]: 3+2
```

```
Out[2]: 5
```

```
In [3]: a = 5  
b = 7  
print(a + b)
```

```
12
```

```
In [5]: n = 7  
for i in range(n):  
    print(i*10)
```

```
0  
10  
20  
30  
40  
50  
60
```

```
In [6]: i = 0  
while True:  
    i+=1  
    if i>5:  
        break  
    print("Test while")
```

```
Test while  
Test while  
Test while  
Test while  
Test while
```

```
In [7]: from matplotlib import pylab as plt  
%matplotlib inline
```

Рисунок 3. Проработка примеров

```
In [7]: from matplotlib import pylab as plt  
%matplotlib inline
```

```
In [8]: x =[i for i in range(50)]  
y =[i**2 for i in range(50)]  
plt.plot(x,y)
```

```
Out[8]: [<matplotlib.lines.Line2D at 0x11b359c76d8>]
```

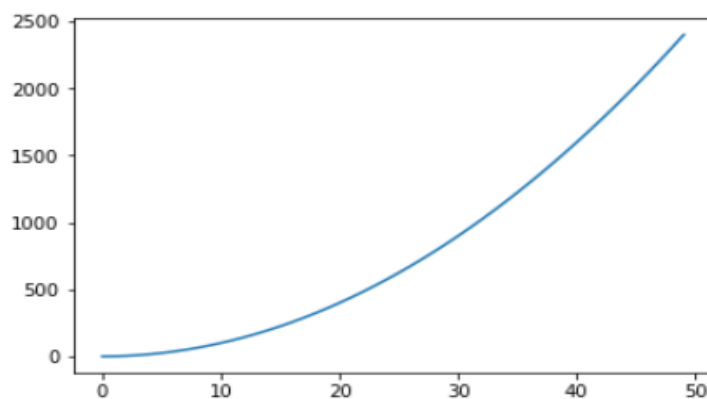


Рисунок 4. Проработка примеров

5.Проработан пример работы с JupyterNotebook

Листинг кода:

```
# Заголовок первого уровня
## Заголовок второго уровня
**Полужирный текст**, *курсив*, `код в строке`
<br>Список:
- Пункт 1
- Пункт 2
- Пункт 3
<br>Формула: $y = mx + b$
```

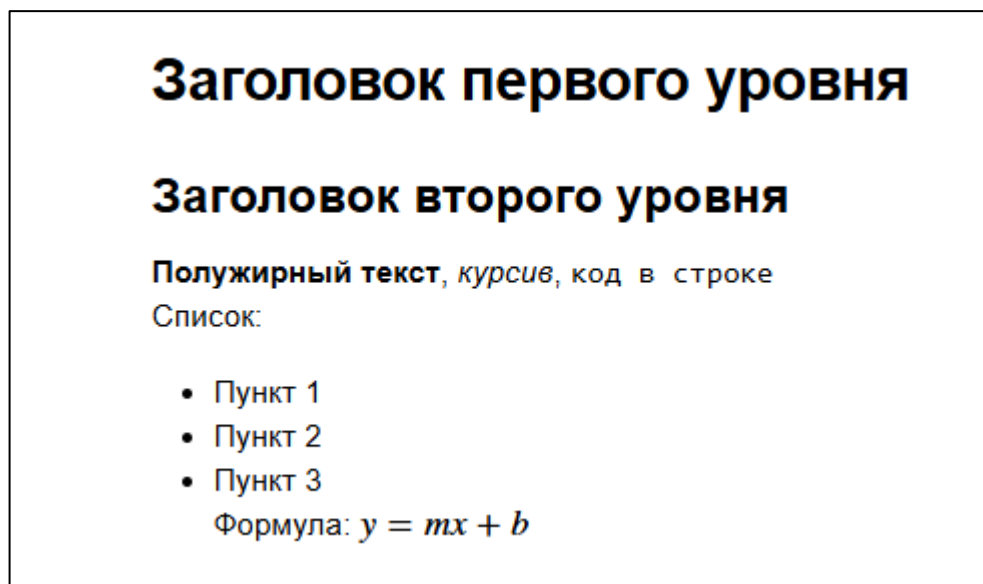


Рисунок 5. Проработка примера

6. Проработан пример с построением графика.

Листинг кода:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
```

```

y = np.sin(x)
plt.plot(x, y)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("График синусоиды")
plt.show()

```

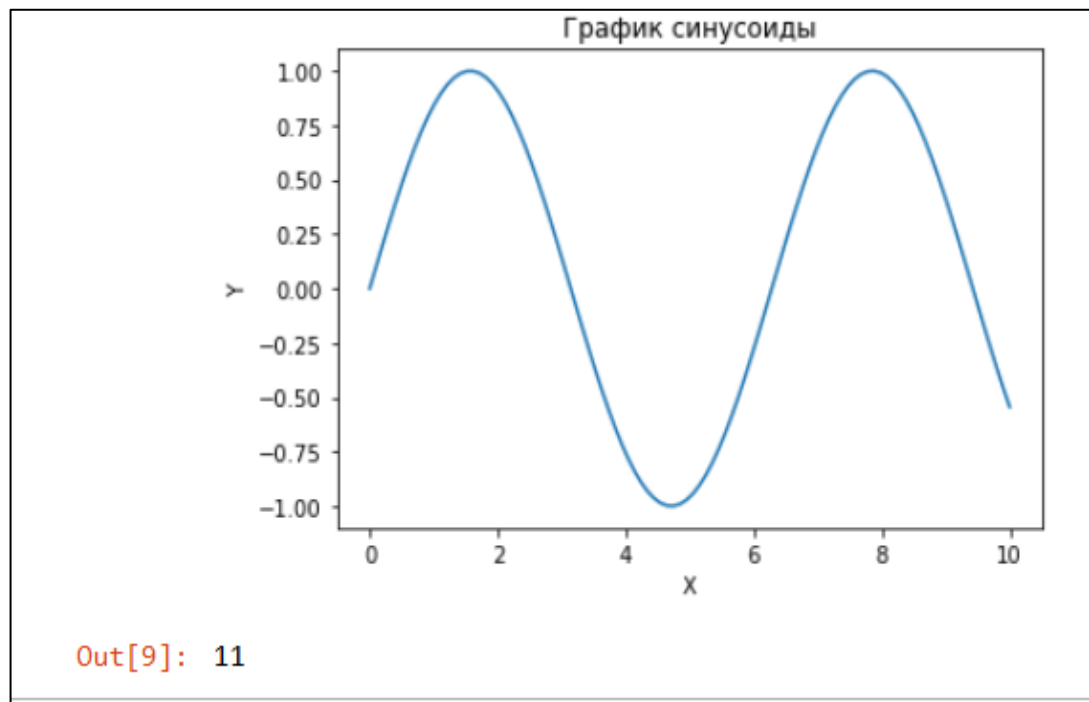


Рисунок 6. Проработка примера с графиком

7. Выполнил практическое задание.

Листинг кода:

```

# Практическое задание №1
** жирный ** и * курсивный * текст
< br > Список:
- Пункт 1
- Пункт 2
- Пункт 3
< ol > Список:
< li > Пункт 1
< li > Пункт 2
< li > Пункт 3
< / ol >
< br >  $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ 

```

```
<img  
src = "Dekstop\1lab.jpg" >
```

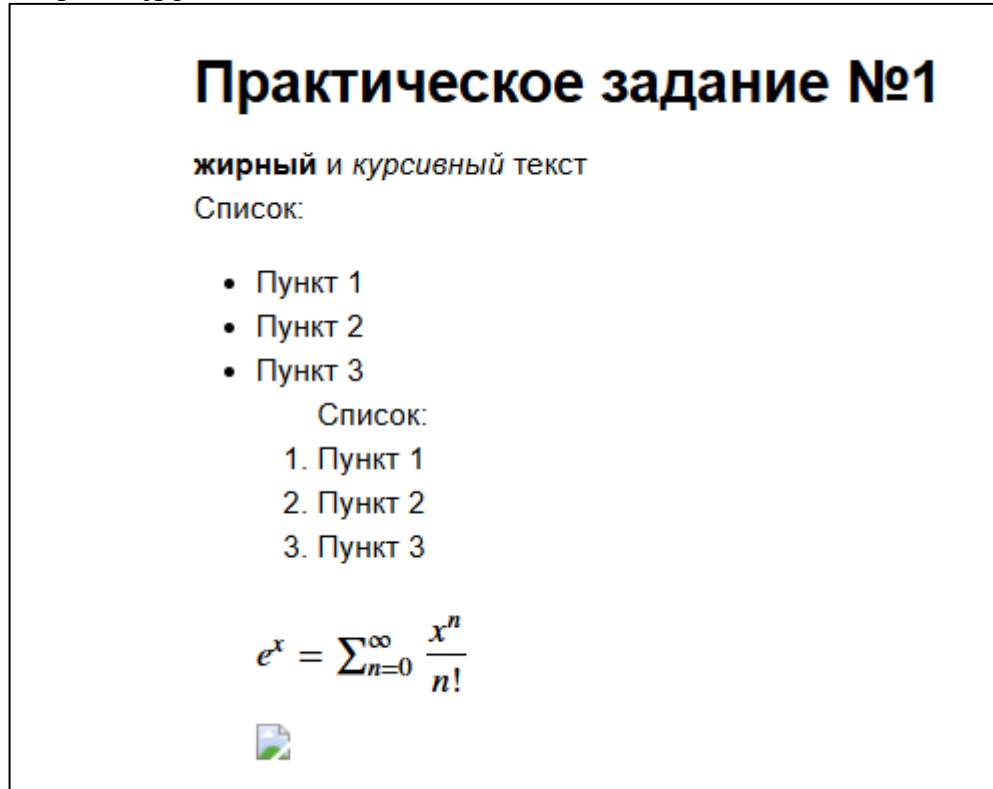


Рисунок 7. Проработка примера

8. Выполнено практическое задание с Markdown-ячейкой в Google Colab.

Листинг кода:

```
# Заголовок первого уровня  
## Заголовок второго уровня  
**Полужирный текст**, *курсив*, `код в строке`  
<br>Список:  
- Пункт 1  
- Пункт 2  
- Пункт 3  
<br>Формула: $y = mx + b$
```

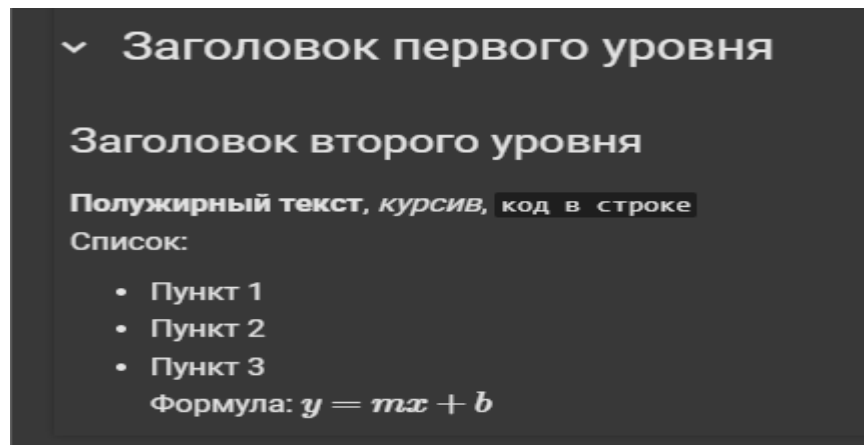


Рисунок 8. Markdown-ячейка

9. Выполнено практическое задание с использованием ячейки Python-кода.

Листинг кода:

```
import os
with open("example.txt", "w") as f:
    f.write("Первая строка\n")
    f.write("Вторая строка\n")
with open("example.txt", "r") as f:
    content = f.read()
    print("Содержимое файла:\n", content)
print("Файл существует:", os.path.exists("example.txt"))
os.remove("example.txt")
print("Файл удален.")
```

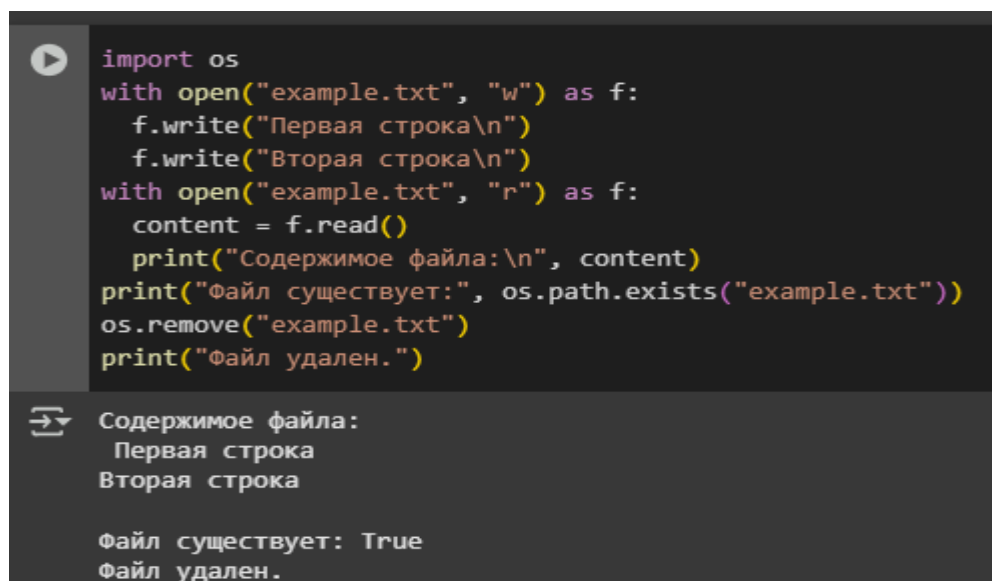


Рисунок 9. Python-код

10. Выполнил индивидуальное задание.

Вариант 9

Ряды Фурье для периодической функции:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(nx) + b_n \sin(nx).$$

Листинг кода:

```
import numpy as np
from scipy.integrate import quad
import matplotlib.pyplot as plt
def f(x):
    return np.where((x >= -np.pi / 2) & (x <= np.pi / 2), 1, -1)
def compute_fourier_coefficients(f, T, N):
    def integrand_a0(x):
        return f(x)
    result_a0, _ = quad(integrand_a0, -T/2, T/2)
    a0 = result_a0 * (1/T)
    coefficients = []
    for n in range(1, N+1):
        def integrand_an(x):
            return f(x)*np.cos(n*2*np.pi*x/T)

        def integrand_bn(x):
            return f(x)*np.sin(n*2*np.pi*x/T)

        result_an, _ = quad(integrand_an, -T/2, T/2)
        result_bn, _ = quad(integrand_bn, -T/2, T/2)

        an = result_an * (2/T)
        bn = result_bn * (2/T)

        coefficients.append((an, bn))

    return a0, coefficients
def approximate_function(a0, coefficients, x_values, T):
    approximated_values = np.zeros_like(x_values)
    for i, x in enumerate(x_values):
        value = a0
        for j, (an, bn) in enumerate(coefficients):
            n = j + 1
            value += an * np.cos(n*2*np.pi*x/T) + bn * np.sin(n*2*np.pi*x/T)
        approximated_values[i] = value
```



```

    return approximated_values
T = 2*np.pi
N = 10
x_values = np.linspace(-np.pi, np.pi, 1000)
a0, coeffs = compute_fourier_coefficients(f, T, N)
approximated_values = approximate_function(a0, coeffs, x_values, T)
plt.figure(figsize=(8, 6))
plt.plot(x_values, f(x_values), label='Исходная функция')
plt.plot(x_values, approximated_values, '--', label=f'Приближённая функция
({N} гармоник)')
plt.legend()
plt.title('Разложение функции в ряд Фурье')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()

```

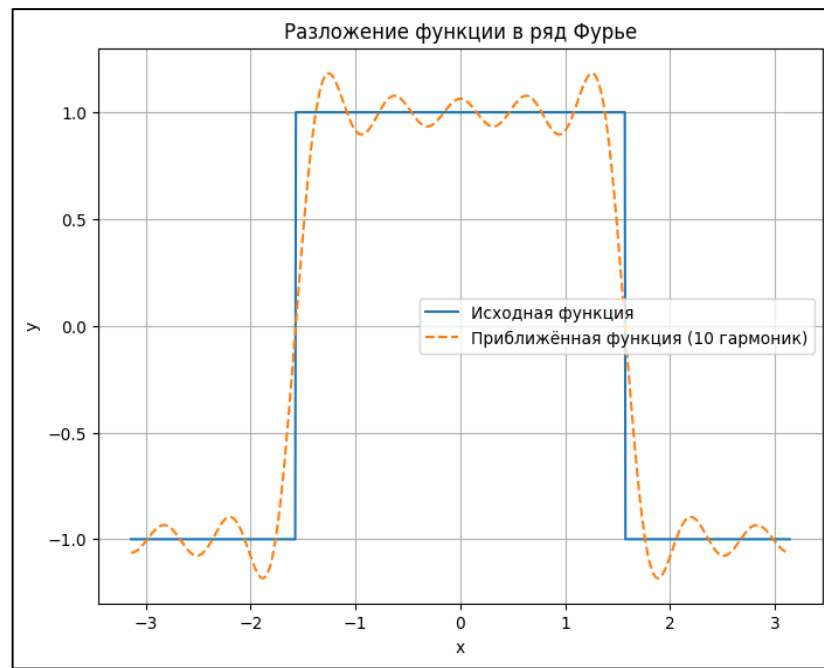


Рисунок 10. Результат работы программы

11.Выполнено задание 2:

Задание 2. Работа с файлами

Цель задания: изучить загрузку, создание и сохранение файлов в Google Colab.

Листинг кода:

```

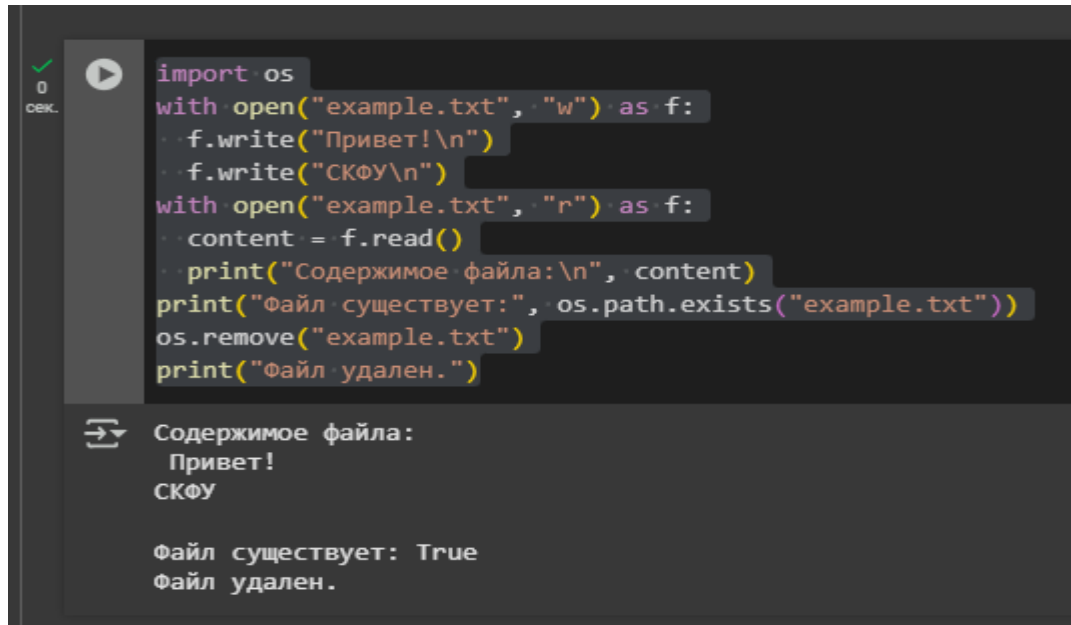
import os
with open("example.txt", "w") as f:

```

```

f.write("Привет!\n")
f.write("СКФУ\n")
with open("example.txt", "r") as f:
    content = f.read()
    print("Содержимое файла:\n", content)
print("Файл существует:", os.path.exists("example.txt"))
os.remove("example.txt")
print("Файл удален.")

```



The screenshot shows a Jupyter Notebook cell with the following Python code:

```

import os
with open("example.txt", "w") as f:
    f.write("Привет!\n")
    f.write("СКФУ\n")
with open("example.txt", "r") as f:
    content = f.read()
    print("Содержимое файла:\n", content)
print("Файл существует:", os.path.exists("example.txt"))
os.remove("example.txt")
print("Файл удален.")

```

Below the code, the output is displayed:

```

Содержимое файла:
Привет!
СКФУ

Файл существует: True
Файл удален.

```

Рисунок 11. Задание 2

12. Выполнено задание 3:

Задание 3. Магические команды Jupyter

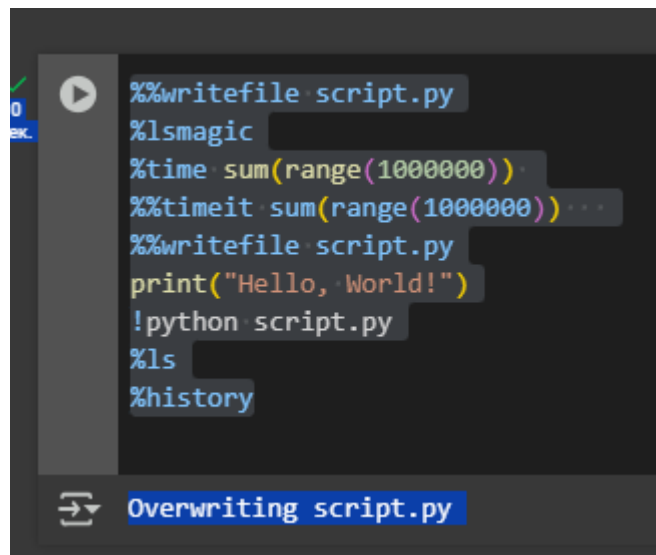
Цель задания: изучить магические команды для удобной работы в Google Colab.

Листинг кода:

```

%%writefile script.py
%lsmagic
%time sum(range(1000000))
%%timeit sum(range(1000000))
%%writefile script.py
print("Hello, World!")
!python script.py
%ls
%history

```



```
%%writefile script.py
%lsmagic
%time sum(range(1000000))
%%timeit sum(range(1000000)) ...
%%writefile script.py
print("Hello, World!")
!python script.py
%ls
%history
```

Overwriting script.py

Рисунок 12. Задание 3

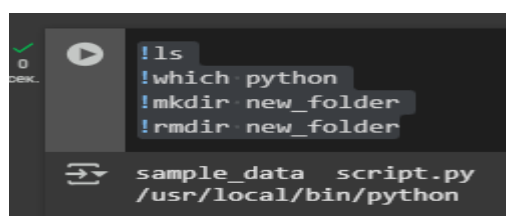
13. Выполнено задание 4:

Задание 4. Взаимодействие с оболочкой системы

Цель задания: изучить выполнение команд терминала прямо из Google Colab

Листинг кода:

```
!ls
!which python
!mkdir new_folder
!rmdir new_folder
```



```
!ls
!which python
!mkdir new_folder
!rmdir new_folder
```

sample_data script.py
/usr/local/bin/python

Рисунок 13. Задание 4

14. Подключил Google Drive к Colab с помощью команды:

Листинг кода:

```
from google.colab import drive
drive.mount('/content/drive')
```

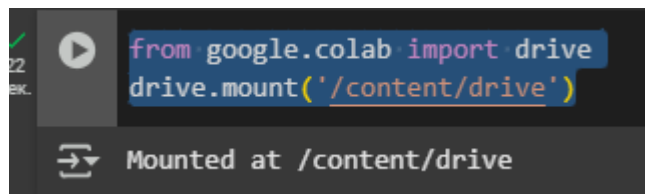


Рисунок 14. Подключение Google диска

15. Проверил, что диск успешно подключился, используя `ls /content/drive/MyDrive`.

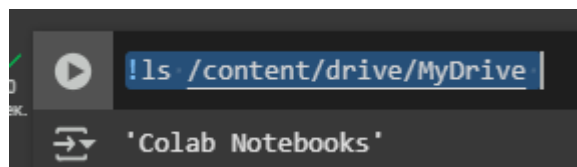


Рисунок 15. Проверка подключения

16. Выполнено задание 5:

Задание 5. Работа с Google Drive в Google Colab

Цель задания: научиться подключать Google Drive, загружать и сохранять файлы, работать с обычными текстовыми файлами

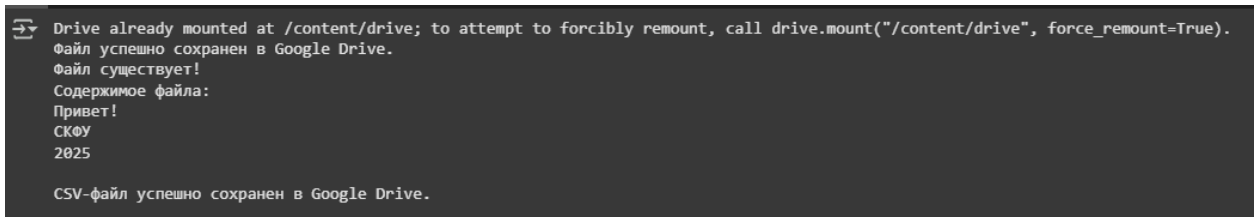
Листинг кода:

```
from google.colab import drive
drive.mount('/content/drive')
file_path = '/content/drive/MyDrive/text.txt'
with open(file_path, 'w') as file:
    file.write('Привет!\n')
    file.write('СКФУ\n')
    file.write('2025\n')
print('Файл успешно сохранен в Google Drive.')
import os
if os.path.exists(file_path):
    print('Файл существует!')
else:
    print('Файл не найден!')
with open(file_path, 'r') as file:
    content = file.read()
print('Содержимое файла:')
print(content)
students = [
    ['Белов Вадим', 20, 'Группа 1'],
    ['Бакулин Вадим', 21, 'Группа 2'],
    ['Кравчук Мирослав', 22, 'Группа 3']
```

```

]
csv_path = '/content/drive/MyDrive/students.csv'
with open(csv_path, 'w') as csv_file:
    for student in students:
        line = ','.join([str(item) for item in student]) + '\n'
        csv_file.write(line)
print('CSV-файл успешно сохранен в Google Drive.')

```



```

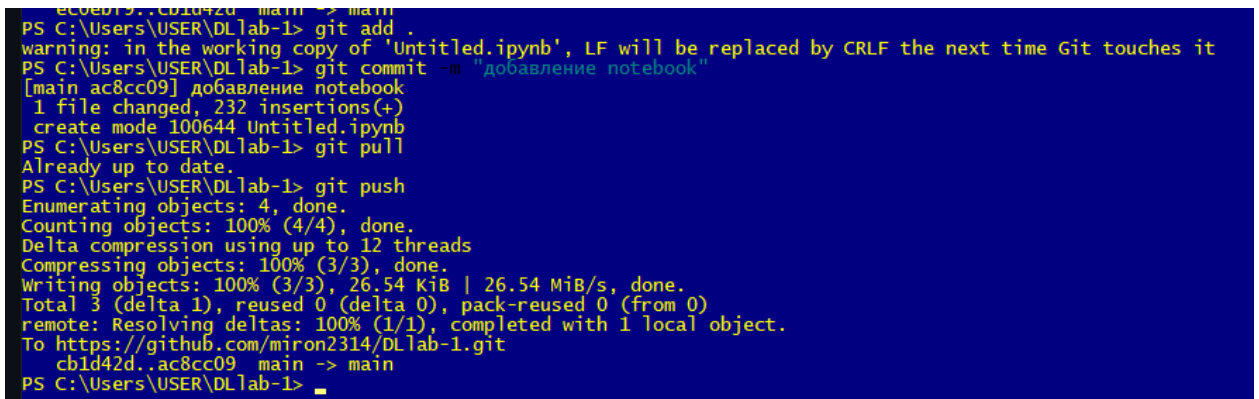
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Файл успешно сохранен в Google Drive.
Файл существует!
Содержимое файла:
Привет!
СКОУ
2025

CSV-файл успешно сохранен в Google Drive.

```

Рисунок 16. Задание 5

17. Зафиксированы изменения на репозитории и отправлены на сервер GitHub.



```

PS C:\Users\USER\DLlab-1> git add .
warning: in the working copy of 'Untitled.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\USER\DLlab-1> git commit -m "добавление notebook"
[main ac8cc09] добавление notebook
1 file changed, 232 insertions(+)
create mode 100644 Untitled.ipynb
PS C:\Users\USER\DLlab-1> git pull
Already up to date.
PS C:\Users\USER\DLlab-1> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 26.54 KiB | 26.54 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/miron2314/DLlab-1.git
cb1d42d..ac8cc09 main -> main
PS C:\Users\USER\DLlab-1>

```

Рисунок 17. Отправка на сервер GitHub

Ответы на контрольные вопросы:

1. Какие основные отличия JupyterLab от Jupyter Notebook?

JupyterLab — это мощная интерактивная среда для разработки, анализа данных и документирования исследований. Она объединяет функциональность Jupyter Notebook, текстового редактора, терминала и файлового менеджера в одном интерфейсе, что делает ее удобным инструментом для научной и образовательной деятельности.

В Jupyter notebook вы можете разрабатывать, документировать и выполнять приложения на языке Python, он состоит из двух компонентов: веб-приложение, запускаемое в браузере, и ноутбуки – файлы, в которых можно работать с исходным кодом программы, запускать его, вводить и выводить данные и т. п.

JupyterLab: это полноценная интегрированная среда (IDE) с вкладками, панелями, файловым менеджером, терминалом и редактором кода. Можно открывать и редактировать несколько файлов одновременно

2. Как создать новую рабочую среду (ноутбук) в JupyterLab?

1. В меню выберите File → New → Notebook .
2. Выберите доступное ядро.
3. Откроется новая тетрадь, состоящая из ячеек.

3. Какие типы ячеек поддерживаются в JupyterLab и как их переключать?

Код (Code) – для написания и выполнения программного кода.

Текст (Markdown) – используется для оформления пояснений, форматированного текста и математических формул на основе LaTeX.

Вывод (Raw) – предназначен для хранения необработанного текста, например, для экспорта в другие форматы.

Для того, чтобы изменить тип ячейки на Markdown нужно нажать **M**.
Чтобы переключить на тип ячейки код- **Y**.

4. Как выполнить код в ячейке и какие горячие клавиши для этого используются?

Запустить ячейку- Shift + Enter

Добавить новую ячейку ниже- B

Удалить текущую ячейку-D D

Изменить тип ячейки на Markdown- M

Изменить тип ячейки на код-Y

Переключение между режимами (редактирование/командный)- Enter/

Esc

5. Как запустить терминал или текстовый редактор внутри JupyterLab?

JupyterLab позволяет открывать терминал (File → New → Terminal) и выполнять команды оболочки.

6. Какие инструменты JupyterLab позволяют работать с файлами и структурами каталогов?

В Jupyter Notebook можно работать только с .ipynb -файлами.

В JupyterLab можно работать с разными типами файлов: .ipynb , .py , .csv , .md и даже .json , .yaml и .txt.

7. Как можно управлять ядрами (kernels) в JupyterLab?

1. **Просмотр установленных ядер.** Их можно посмотреть на странице Launcher или через терминал, запустив команду `jupyter kernelspec list`.

2. **Управление запущенными ядрами.** Для этого нужно использовать вкладку «Запущенные терминалы и ядра» (Running Terminals and Kernels). На ней можно закрыть или выключить открытые вкладки, запущенные ядра и терминалы по отдельности, наведя курсор на правую сторону вкладки и нажав на появившуюся кнопку X.

3. **Использование кнопки меню Kernel.** Она предлагает набор опций для управления ядрами: перезапустить, выключить и изменить ядра.

8. Каковы основные возможности системы вкладок и окон в интерфейсе JupyterLab?

Основные возможности системы вкладок и окон в интерфейсе JupyterLab:

Основная рабочая область. Позволяет группировать документы (блокноты, текстовые файлы и пр.) и другие инструменты (терминалы, консоли и т. д.) в виде панелей с вкладками, размер и расположение которых можно изменить перетаскиванием.

Вкладка Tabs в боковой панели. Показывает список открытых документов и инструментов в рабочей области с возможностью переключения.

Режим работы с отдельным документом. Позволяет сфокусироваться на отдельном документе и инструменте без того, чтобы закрывать все остальные вкладки в рабочей области. Его можно запустить из панели View («Single-Document Mode») или воспользоваться сочетанием горячих клавиш (по умолчанию Ctrl+Shift+Enter).

Настройка рабочего пространства. В JupyterLab есть возможность разделить окна по горизонтали и вертикали.

Кроме того, для навигации и запуска инструментов в JupyterLab можно использовать горячие клавиши, которые можно настроить

9. Какие магические команды можно использовать в JupyterLab для измерения времени выполнения кода? Приведите примеры.

`%timeit my_function()` Измеряет время выполнения команды.

`%timeit.` Запускает команду несколько раз и вычисляет среднее время выполнения. Пример использования:

```
%timeit sum(range(100))
```

`%%time.` Даёт информацию о единичном запуске кода в ячейке.

Например,

```
import time
```

```
start_time = time.time()
```

```
# измеряемый код
```

```
sum(range(100))
```

```
end_time = time.time()
```

```
elapsed_time = end_time - start_time
```

```
print(f'Время запуска: {elapsed_time:.2f} секунд')
```

10. Какие магические команды позволяют запускать код на других языках программирования в JupyterLab?

Некоторые магические команды, которые позволяют запускать код на других языках программирования в JupyterLab:

`%%python2`, `%%python3`, `%%R`, `%%bash`. Обозначают начало ячейки с кодом на определённом языке программирования (например, Python, R, Bash).

`%%latex`. Позволяют получать отрисовку ячеек с кодом в LaTeX.

`%%html`, `%%javascript` (или `%%js`), `%%markdown`, `%%ruby`, `%%sh`.

Аналогично служат команды для других языков программирования, например, Ruby, Pearl, JavaScript.

11. Какие основные отличия Google Colab от JupyterLab?

Google Colab (или Google Colaboratory) – это облачная среда для работы с Jupyter Notebook, предоставляемая Google. Она позволяет запускать код на удаленных серверах, что особенно полезно для задач машинного обучения и анализа данных. Google Colab предоставляет доступ к GPU и TPU бесплатно (с ограничениями), а также интегрируется с Google Диском.

12. Как создать новый ноутбук в Google Colab?

1. Перейдите в Файл → Новый ноутбук.
2. Откроется рабочая область с первой ячейкой.

13. Какие типы ячеек доступны в Google Colab, и как их переключать?

Код (Code) – для написания и выполнения Python-кода.

Текст (Markdown) – используется для оформления документации, пояснений и формул (LaTeX).

14. Как выполнить код в ячейке Google Colab и какие горячие клавиши для этого используются?

Запустить ячейку- Shift + Enter

Добавить новую ячейку ниже- Ctrl+ M B

Удалить текущую ячейку- Ctrl + M D

Изменить тип ячейки на Markdown- Ctrl + M M

Изменить тип ячейки на код- Ctrl + M Y

15. Какие способы загрузки и сохранения файлов поддерживает Google Colab?

Google Colab поддерживает загрузку и сохранение файлов через локальный компьютер, Google Диск, а также с помощью URL-ссылок и API.

16. Как можно подключить Google Drive к Google Colab и работать с файлами?

Google Colab позволяет работать с файлами на Google Диске и загружать файлы в локальное окружение.

Подключение Google Диска:

```
from google.colab import drive  
drive.mount('/content/drive')
```

После выполнения появится ссылка, по которой нужно авторизоваться.

!ls- Просмотр списка файлов в текущей директории

!pwd- Вывод текущей директории

!rm filename- Удаление файла

!mkdir new_folder- Создание папки

17. Какие команды используются для загрузки файлов в Google Colab из локального компьютера?

В Google Colab для загрузки файлов из локального компьютера можно использовать следующие команды:

Использование модуля files из библиотеки google.colab

Uploaded =files.upload() После выполнения этой команды появится кнопка для выбора файлов на локальном компьютере. Загруженные файлы будут доступны в переменной uploaded.

2.Использование files.download Чтобы скачать файл из Colab на локальный компьютер, можно использовать данную команду.

18. Как посмотреть список файлов, хранящихся в среде Google Colab?

Чтобы посмотреть список файлов, хранящихся в среде Google Colab, используется команда! ls.

19. Какие магические команды можно использовать в Google Colab для измерения времени выполнения кода? Приведите примеры.

1.%time: измеряет время одной строки (%time sum (range (1000))

2. `%timeit`: выполняет строку несколько раз до точности (`%timeit sum(range(10000))`).

3. `%%time`: измеряет время всего блока кода.

Пример:

```
%%time
```

```
total = sum ( range(1000));
```

4. `%timeit`: выполняет блок кода несколько раз.

Пример:

```
%%timeit
```

```
total = sum ( range(10000));
```

20. Как можно изменить аппаратные ресурсы в Google Colab (например, переключиться на GPU)?

В Google Colab выберите Среда выполнения> Изменить среду выполнения, затем выберите GPU и нажмите Сохранить.

Вывод: в ходе лабораторной работы были приобретены навыки работы с Jupyter Notebook, JupyterLab и Google Colab.