

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 5

Выполнил:
Беков Шамиль Расулович
2 курс, группа ИВТ-б-о-23-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Проверил:
Доцент департамента цифровых,
робототехнических систем и электроники
института перспективной инженерии
Воронкин Роман Александрович

(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.

Порядок выполнения работы:

Работа в Jupyter Notebook.

Запустили оболочку Jupyter-Notebook в браузере.

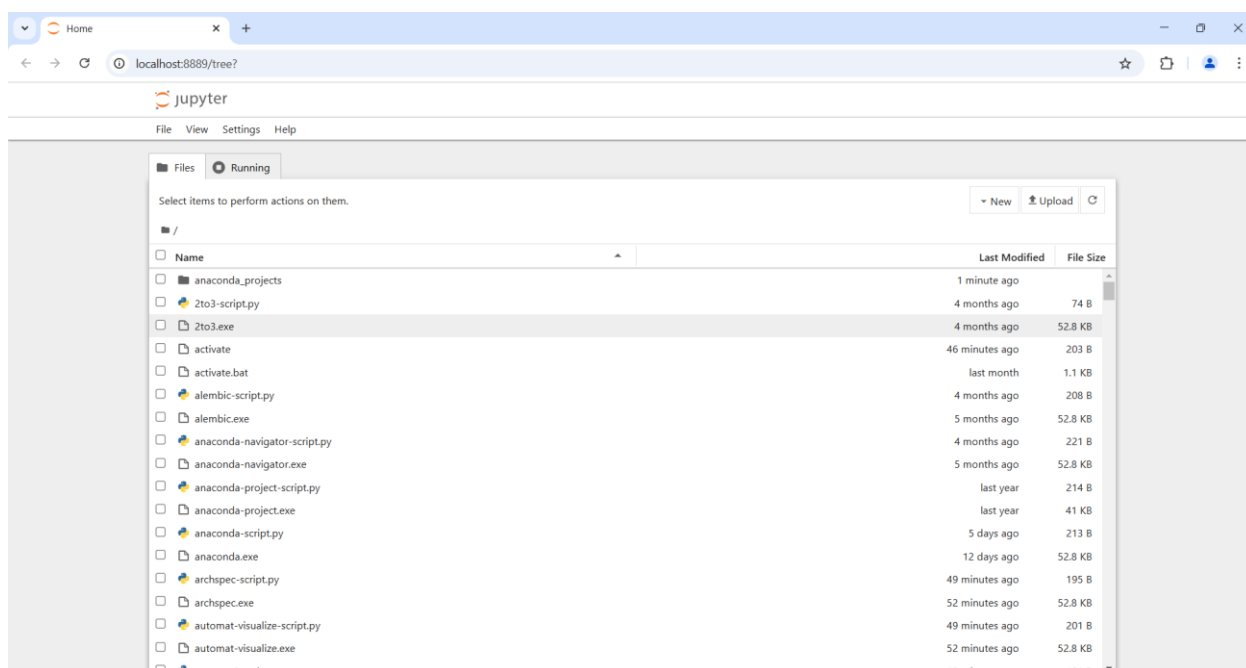


Рисунок 1. Оболочка Jupyter-Notebook в браузере

Создали ноутбук и запустили его.

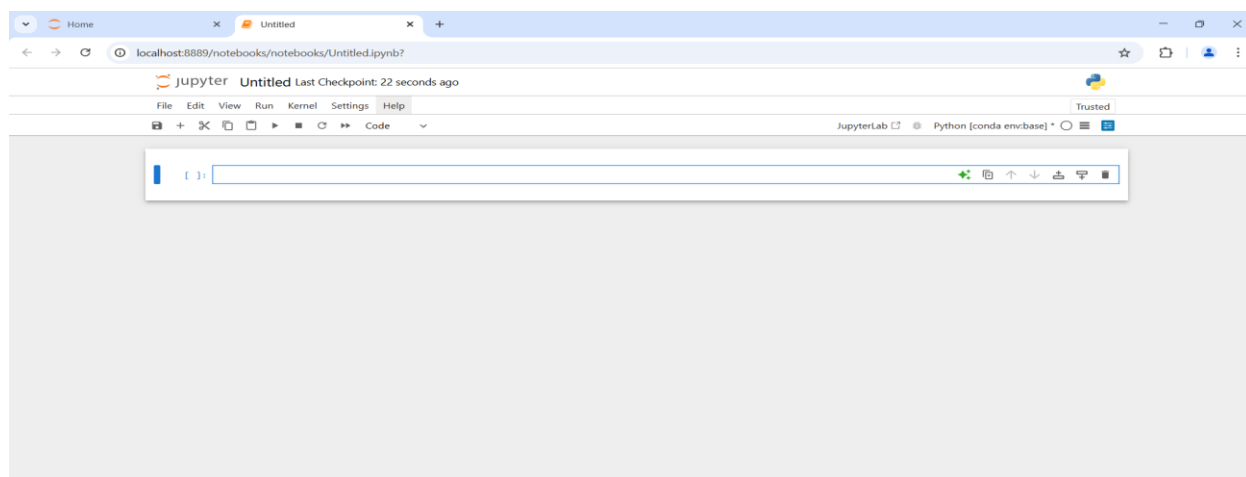


Рисунок 2. Запущенный ноутбук

Напишем небольшую программу и выполним ее, существует несколько способов запуска ячеек:

Ctrl+Enter – выполнить содержимое ячейки.

Shift+Enter - выполнить содержимое ячейки и перейти на ячейку ниже.

Alt+Enter – выполнить содержимое ячейки и вставить новую ячейку ниже.



```
[3]: 2+3
[3]: 5

[5]: a=5
     b=7
     print(a+b)
12

[1]: n=7
     for i in range(n):
         print(i*10)
0
10
20
30
40
50
60

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

[3]: print('test')
test
```

Рисунок 3. Запуск ячеек с помощью Ctrl+Enter; Shift+Enter; Alt+Enter

Выведем изображение (график) в ноутбук, для этого выполним команду `%matplotlib inline`, по умолчанию графики Matplotlib открываются в отдельном окне или вкладке веб-браузера, с помощью опции `inline` график отображается непосредственно в ячейке ноутбука.

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

```
[41]: import matplotlib.pyplot as plt

      %matplotlib inline
      x=[i for i in range(50)]
      y=[i**2 for i in range(50)]

      plt.plot(x,y)
      plt.show()
```

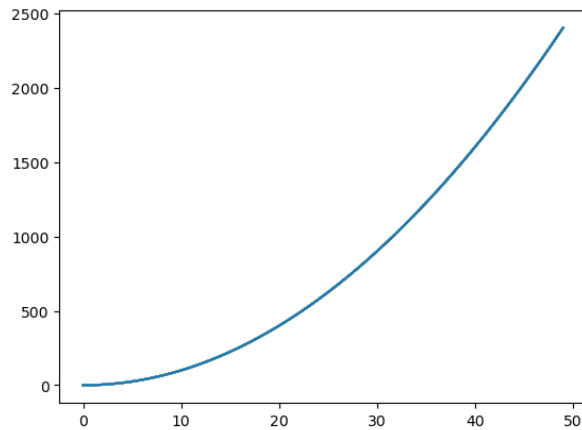


Рисунок 4. Вывод графика Matplotlib

Выполнили магические команды, т.е. дополнительные команды, выполняемые в рамках оболочки, которые облегчают процесс разработки и расширяют наши возможности.

```
[43]: %lsmagic
```

```
[43]: root
      line
      cell
      js "DisplayMagics"
      javascript "DisplayMagics"
      latex "DisplayMagics"
      svg "DisplayMagics"
      html "DisplayMagics"
      markdown "DisplayMagics"
      prun "ExecutionMagics"
      debug "ExecutionMagics"
      timeit "ExecutionMagics"
      time "ExecutionMagics"
      capture "ExecutionMagics"
      code_wrap "ExecutionMagics"
      sx "OSMagics"
      system "OSMagics"
      ! "OSMagics"
      writefile "OSMagics"
      script "ScriptMagics"
      sh "Other"
      bash "Other"
      perl "Other"
      ruby "Other"
      python "Other"
      python2 "Other"
      python3 "Other"
      pypy "Other"
      cmd "Other"
```

Рисунок 5. Использование команды %lsmagic

```
[45]: %env TEST=5
```

```
env: TEST=5
```

Рисунок 6. Работа с переменным окружением с помощью команды `%env`

```
[91]: %run test.py
```

```
Hello  
Hello  
Hello
```

Рисунок 7. Запуск кода из другого файла с помощью команды `%run`

```
[95]: %%time  
import time  
for i in range(50):  
    time.sleep(0.1)
```

```
CPU times: total: 31.2 ms  
Wall time: 5.03 s
```

Рисунок 8. Получение информации о времени работы кода в рамках одной ячейки с помощью `%%time`

```
[93]: %timeit x=[(i**10) for i in range(10)]
```

```
3.19 µs ± 282 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

Рисунок 9. Получение информации о среднем значении трех наиболее быстрых прогонах с помощью `%timeit`

Работа в Jupyter Lab.

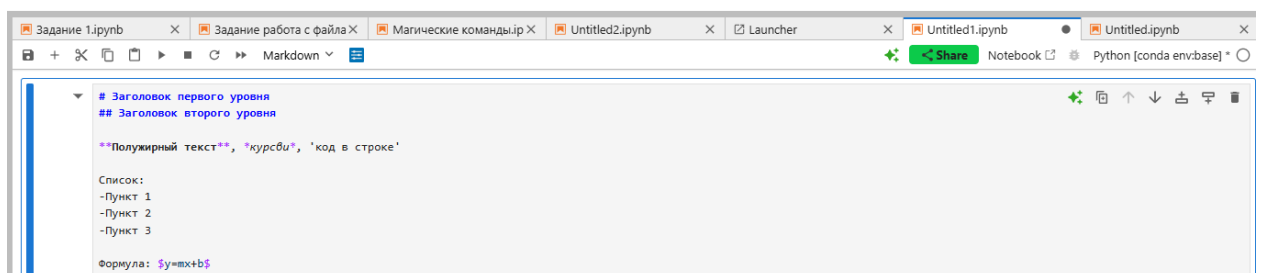


Рисунок 10. Синтаксис для форматирования текста Markdown

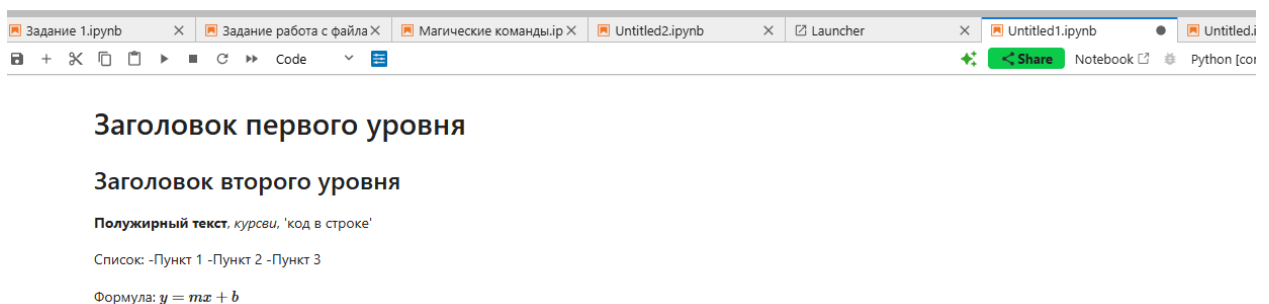


Рисунок 11. Отформатированный текст Markdown

```
[17]: %timeit sum(range(1000))
```

42.7 μ s \pm 1.74 μ s per loop (mean \pm std. dev. of 7 runs, 10,000 loops each)

Рисунок 12. Измерение времени выполнения команды

```
[19]: %%time
total=0
for i in range(10**6):
    total+=i
```

CPU times: total: 359 ms
Wall time: 348 ms

Рисунок 13. Измерили время выполнения всей ячейки

Wall time: 348 ms

```
[5]: import pandas as pd

df=pd.read_csv("test.csv")
df.head()
```

[5]: **Bekov Shamil**

Рисунок 14. Работа с файлами в Jupyter Lab

```
[3]: 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()
```

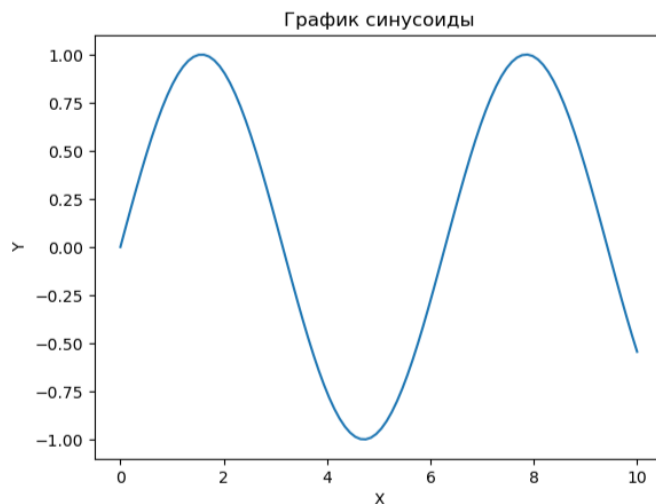


Рисунок 15. Изображение графика с помощью matplotlib в Jupyter Lab

```
PS C:\Users\User\anaconda3> python --version
```

Python

```
PS C:\Users\User\anaconda3> █
```

Google Colab:

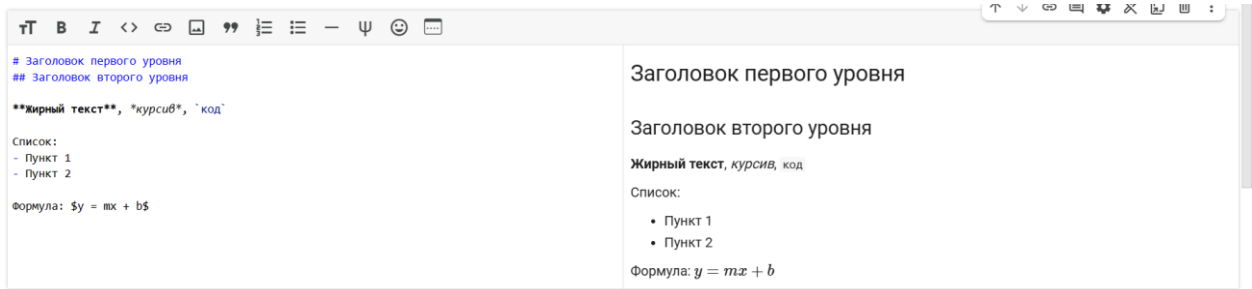


Рисунок 16. Markdown-форматирование в Google Colab.

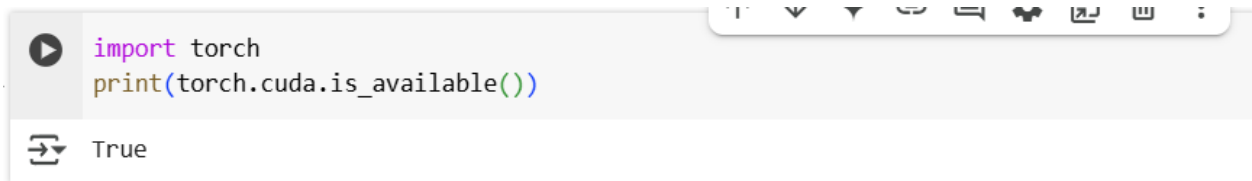


Рисунок 17. Проверка наличия GPU

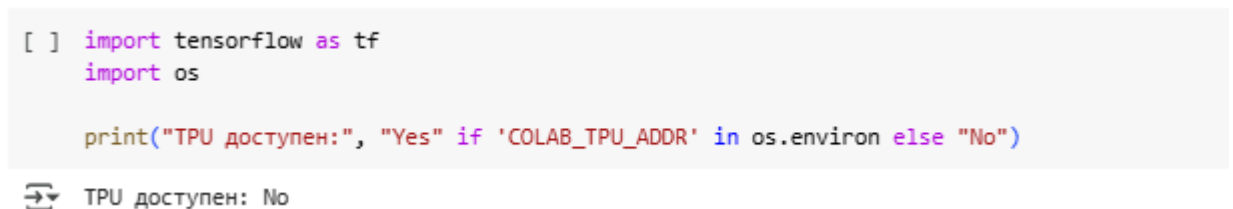


Рисунок 18. Проверка наличия TPU

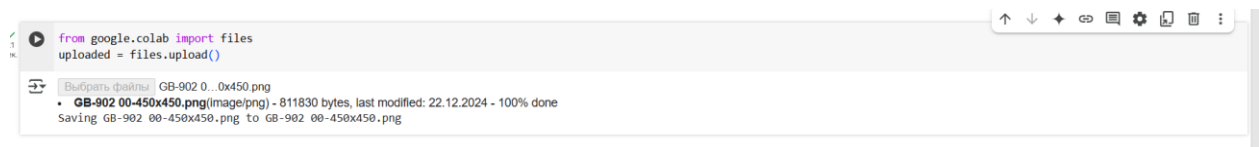


Рисунок 19. Загрузка файлов в локальное окружение

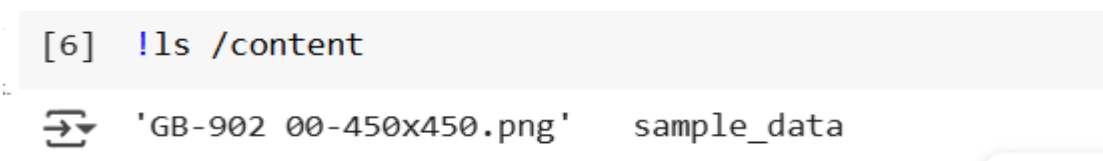


Рисунок 20. Просмотр списка файлов в директории

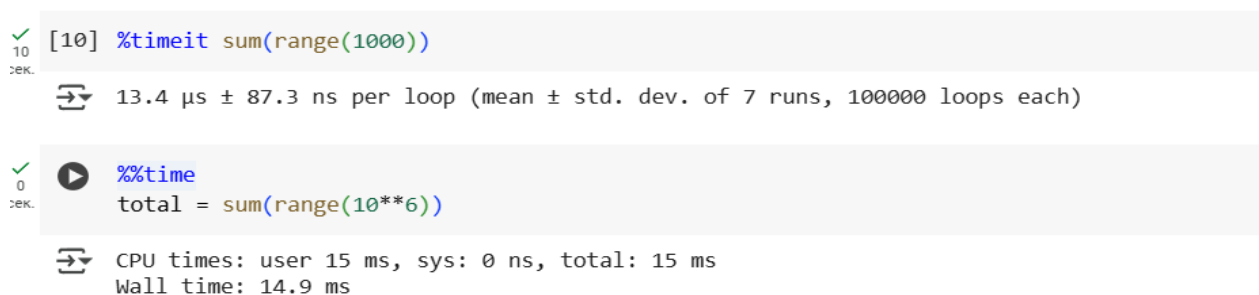


Рисунок 21. Измеряем время выполнения кода и время выполнения всей ячейки

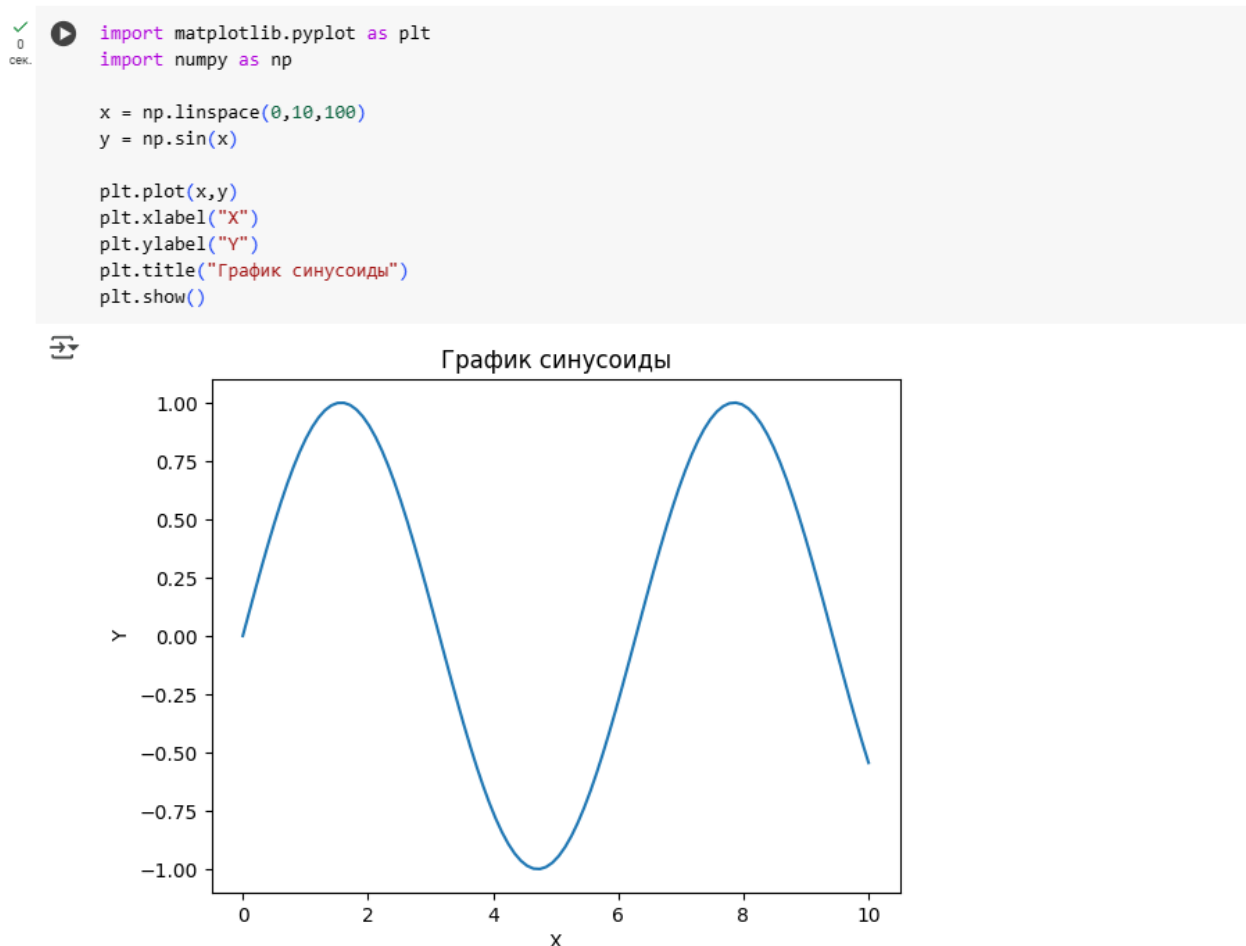


Рисунок 22. Построили график с помощью динамической визуализации

The figure shows a terminal window with the command `!pip install numpy pandas matplotlib` and its output. The output lists the versions of the installed packages and their dependencies:

```
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

Рисунок 23. Установка стандартных библиотек

Задание 1 (Работа с ячейками Markdown):

The figure shows a Jupyter Notebook cell with the following content:

```
# Практическое задание #1

**Жирный текст**, *курсивный текст*

Нумерованный список:
1. Первый
2. Второй
3. Третий

Маркированный список:
- Первый
- Второй
- Третий

## Интеграл от экспоненциальной функции:

$$\int e^x dx = e^x + C$$


![Белая птица](https://sm.author.today/content/2024/04/05/5a3867e2c06e405b94835cc1549e4eal.jpg)
```

Рисунок 24. Часть задания 1 с Markwond-ячейкой.

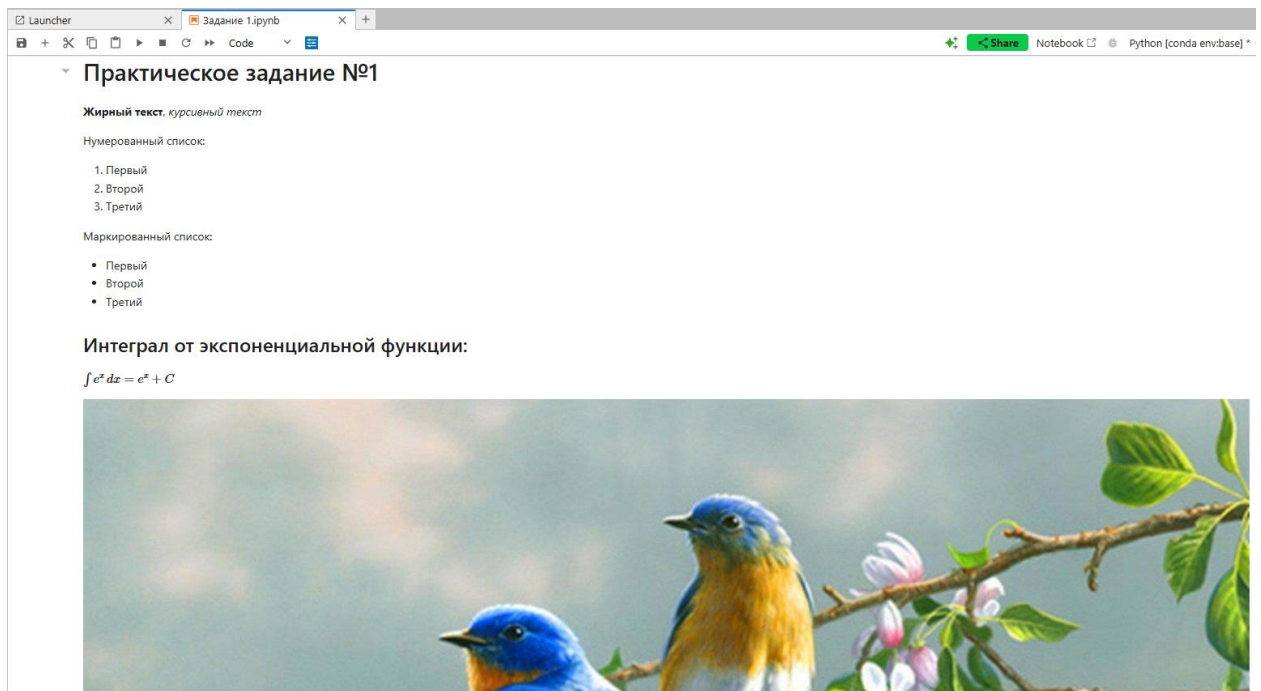


Рисунок 25. Результат работы Markdown-ячейки

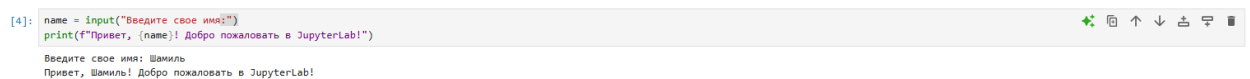


Рисунок 26. Часть задания 1 с ячейкой python-кода.

Задание 2 (Работа с файлами):

1. Создайте и сохраните текстовый файл с помощью `open()`.
2. Запишите в него несколько строк текста.
3. Закройте файл и затем откройте его снова, считав содержимое и выведя на экран.
4. Проверьте, существует ли файл, используя `os.path.exists()`.
5. Удалите файл с помощью модуля `os`.

Рисунок 27. Условия к заданию 2

```
import os

with open("example.txt", "w") as f:
    f.write("Беков\n")
    f.write("Шамиль\n")
```

Рисунок 28. Создание файла и запись в него строк текста.

```
with open("example.txt", "r") as f:
    content=f.read()
    print("Содержимое файла:\n", content)
```

Рисунок 29. Открытие файла и считывание содержимого.

```
print("Файл существует:", os.path.exists("example.txt"))
```

Рисунок 30. Проверка существования файла.

```
os.remove("example.txt")  
print("Файл удален.")
```

Рисунок 31. Удаление файла.

```
[2]: import os  
  
with open("example.txt", "w") as f:  
    f.write("Беков\n")  
    f.write("Шамиль\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.")
```

Содержимое файла:

Беков

Шамиль

Файл существует: True

Файл удален.

[]:

Рисунок 32. Результат работы.

Задание 3 (Магические команды Jupyter):

1. Выведите список всех доступных магических команд (`%lsmagic`).
2. Используйте `%time` и `%%timeit` для измерения времени выполнения кода.
3. Создайте Python-скрипт в Jupyter (`%%writefile script.py`) и выполните его через `!python script.py` .
4. Выведите список файлов в текущей директории с помощью `%ls` .
5. Используйте `%history` для просмотра истории команд.

Рисунок 33. Условия к заданию 3

```
[3]: %lsmagic
```

```
[3]: root  
    cell  
    line
```

Find...



Рисунок 34. Использование команды %lsmagic

```
[7]: %time sum(range(12345678))

CPU times: total: 672 ms
Wall time: 684 ms
[7]: 76207876467003
```

Рисунок 35. Работа команды %time

```
[55]: %ls

'*. y r6ea*069f C:\Users\User\anaconda3\Scripts
'faE0n0 *.fa b*.: B81B-8858

'afa|E-*f Y feE C:\Users\User\anaconda3\Scripts

05.03.2025 09:13 <DIR> .
18.02.2025 23:05 <DIR> ..
24.09.2024 19:58 1a676 .anaconda_powershell_prompt-post-link.bat
24.09.2024 19:58 1a676 .anaconda_prompt-post-link.bat
30.09.2024 18:21 2a013 .anaconda-navigator-post-link.bat
05.03.2025 08:53 <DIR> .ipynb_checkpoints
05.11.2024 18:58 246 .nb_conda_kernels-post-link.bat
05.11.2024 18:58 247 .nb_conda_kernels-pre-unlink.bat
24.09.2024 20:02 2a013 .notebook-post-link.bat
28.08.2023 11:21 1a242 .qt-post-link.bat
24.09.2024 20:01 2a013 .spyder-post-link.bat
05.03.2025 08:42 <DIR> .virtual_documents
18.02.2025 22:59 <DIR> __pycache__
04.10.2024 16:22 54a032 2to3.exe
04.10.2024 16:22 74 2to3-script.py
18.02.2025 23:04 203 activate
07.01.2025 22:32 1a111 activate.bat
20.09.2024 16:53 54a032 alembic.exe
21.10.2024 20:44 208 alembic-script.py
06.02.2025 19:06 54a032 anaconda.exe
18.02.2025 23:49 <DIR> anaconda_projects
20.09.2024 16:53 54a032 anaconda-navigator.exe
30.09.2024 18:30 221 anaconda-navigator-script.py
.. ..
```

Рисунок 36. Работа команды %ls

```
[57]: %history

%%writefile test_script.py
for i in range(3):
    print(f"Итерация {i}")

!python test_script.py

%time sum(range(1000000))

import json
import getpass
import hashlib

def import_pandas_safely():
    try:
        return __import__('pandas')
    except ImportError:
        return False

__pandas = import_pandas_safely()

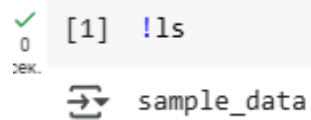
def is_data_frame(v: str):
    obj = eval(v)
    if isinstance(obj, pandas.core.frame.DataFrame) or isinstance(obj, pandas.core.series.Series):
```

Рисунок 37. Работа команды %history

Задание 4. (Взаимодействие с оболочкой системы).

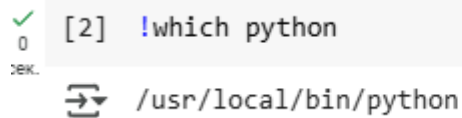
1. Выведите список файлов в текущей директории с помощью `!ls`.
2. Проверьте, какой Python используется (`!which python`).
3. Создайте папку `test_folder` (`!mkdir test_folder`) и убедитесь, что она появилась.
4. Переместите файл в новую папку и удалите его.
5. Очистите вывод в ячейке (`!clear`).

Рисунок 38. Условия к заданию 4



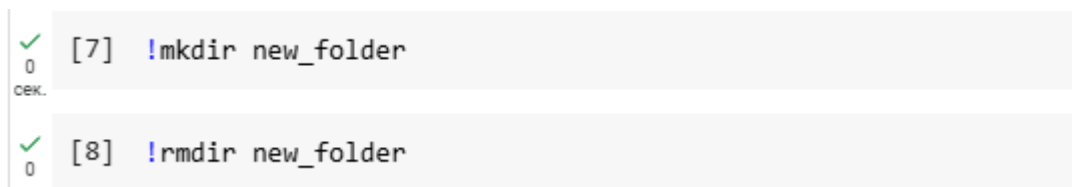
A screenshot of a Google Colab terminal window. On the left, there is a green checkmark, a '0' in a box, and the text 'DEK.'. The terminal shows the command `[1] !ls` being executed, and the output `sample_data` is displayed below it.

Рисунок 39. Работа `!ls` в Google Colab



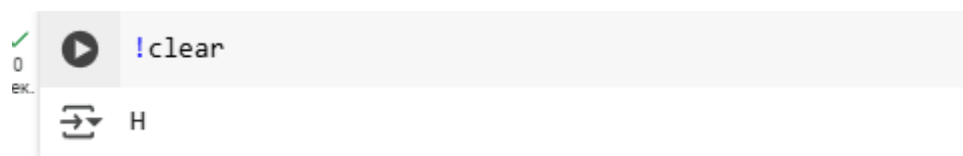
A screenshot of a Google Colab terminal window. On the left, there is a green checkmark, a '0' in a box, and the text 'DEK.'. The terminal shows the command `[2] !which python` being executed, and the output `/usr/local/bin/python` is displayed below it.

Рисунок 40. Работа `!which python` в Google Colab



A screenshot of two Google Colab terminal windows. The top window shows the command `[7] !mkdir new_folder` being executed, with a green checkmark, a '0' in a box, and 'DEK.' on the left. The bottom window shows the command `[8] !rmdir new_folder` being executed, with a green checkmark, a '0' in a box, and 'DEK.' on the left.

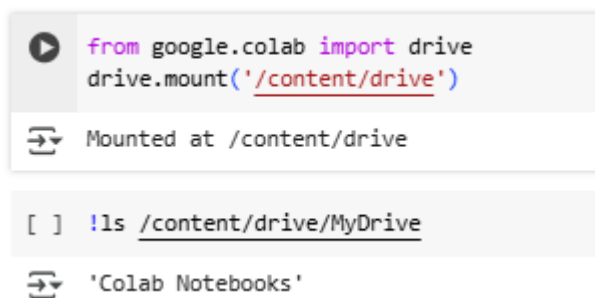
Рисунок 41. Создание и удаление файла в Google Colab



A screenshot of a Google Colab terminal window. On the left, there is a green checkmark, a '0' in a box, and the text 'DEK.'. The terminal shows the command `!clear` being executed, and the output `Н` is displayed below it.

Рисунок 42. Использование команды `!clear`

Задание 5 (Работа с Google Drive в Google Colab):



A screenshot of two Google Colab terminal windows. The top window shows the code `from google.colab import drive` and `drive.mount('/content/drive')` being executed, with a play button icon on the left. The output `Mounted at /content/drive` is displayed below it. The bottom window shows the command `[] !ls /content/drive/MyDrive` being executed, with a green checkmark, a '0' in a box, and 'DEK.' on the left. The output `'Colab Notebooks'` is displayed below it.

Рисунок 43. Подключение Google Drive к Colab, проверка успешного подключения диска

```
[ ] file_path = "/content/drive/MyDrive/shamil.txt"

with open(file_path, "w") as f:
    f.write("Первая строка текста\n")
    f.write("Вторая строка текста.")

print("Файл успешно сохранен в Google Drive.")
```

⇒ Файл успешно сохранен в Google Drive.

Рисунок 44. Открытие файла в текстовом файле и запись в него нескольких
текст файла

```
[ ] with open(file_path, "r") as f:
    data = f.read()
    print(data)
```

⇒ Первая строка текста
Вторая строка текста.

Рисунок 45. Открытие файла и считывание его содержимого

```
[ ] students = [
    ["ФИО", "Возраст", "группа"],
    ["Беков Ш.Р", 19, "ИВТ-6-о-23-1"],
    ["Петров П.П.", 22, "ИВТ-102"],
    ["Сидорова А.А.", 21, "ИВТ-103"]
]
csv_path = "/content/drive/MyDrive/students.csv"
```

```
[ ] with open(csv_path, "w") as f:
    for student in students:
        f.write(",".join(map(str, student)) + "\n")

print("Файл students.csv успешно сохранен в Google Drive.")
```

⇒ Файл students.csv успешно сохранен в Google Drive.

Рисунок 46. Создание CSV-файла и запись в него списка

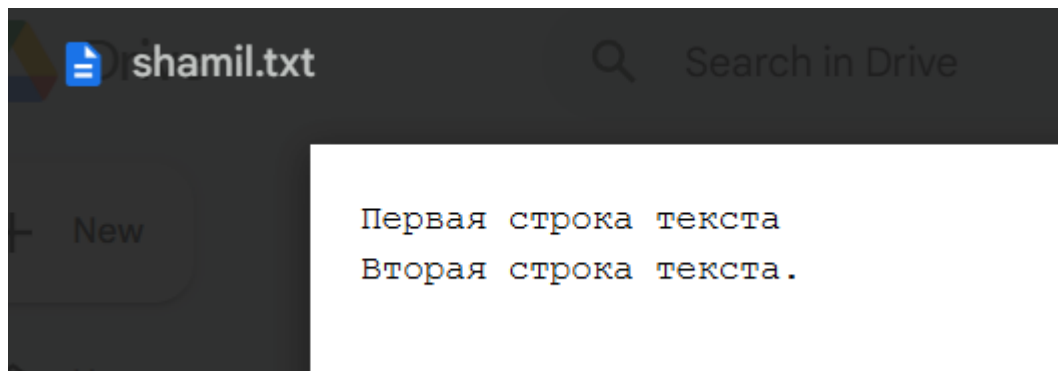


Рисунок 47. Результат работы кода на рисунке 45

	A	B	C
1	ФИО	Возраст	группа
2	Беков Ш.Р	19	ИВТ-6-а-23-1
3	Петров П.П.	22	ИВТ-102
4	Сидорова А.А.	21	ИВТ-103

Рисунок 48. Результат работа кода на рисунке 46

Контрольные вопросы:

1. Какие основные отличия JupyterLab от Jupyter Notebook?

JupyterLab — это более мощная и модульная версия Jupyter Notebook с поддержкой вкладок, окон, текстового редактора, терминала и других инструментов. В Jupyter Notebook интерфейс более простой, с одной колонкой, содержащей ноутбук.

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

В JupyterLab можно создать новый ноутбук через «File» → «New» → «Notebook» или нажать на значок «+» и выбрать «Notebook» в Launcher.

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

Поддерживаются ячейки кода, Markdown и Raw. Переключение — через меню «Cell» → «Cell Type» или горячие клавиши (например, Esc + M для Markdown, Esc + Y для кода).

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

Код выполняется нажатием Shift + Enter или кнопки «Run» в панели инструментов. Ctrl + Enter выполняет без перехода к следующей ячейке, Alt + Enter выполняет и вставляет новую ячейку.

5. Как запустить терминал или текстовый редактор внутри JupyterLab?

Терминал и текстовый редактор запускаются через Launcher (значок «+») или «File» → «New» → «Terminal»/«Text File».

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

В JupyterLab есть файловый браузер в левой панели, который позволяет управлять файлами и каталогами. Можно загружать, удалять, переименовывать файлы.

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

Ядра управляются через «Kernel» → «Restart Kernel», «Interrupt Kernel», «Shut Down Kernel» и через панель «Running Terminals and Kernels».

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

JupyterLab поддерживает систему вкладок и окон, позволяя работать с несколькими ноутбуками, терминалами и текстовыми файлами одновременно, перетаскивать и организовывать их.

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

%time измеряет время выполнения одной строки, %%time измеряет время выполнения всей ячейки, %timeit и %%timeit выполняют код несколько раз и показывают среднее время выполнения.

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

`%magic %lsmagic` показывает список доступных магических команд. `%script` позволяет запускать код на других языках, например, `%%bash`, `%%perl`, `%%ruby`, `%%python3`.

11. Какие основные отличия Google Colab от JupyterLab?

Google Colab — облачный сервис, не требует локальной установки. Поддерживает GPU, TPU, интеграцию с Google Drive. JupyterLab работает локально, требует установки и настройки.

12. Как создать новый ноутбук в Google Colab?

В Google Colab новый ноутбук создается через «Файл» → «Создать новый блокнот»

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

Доступны ячейки кода и текстовые (Markdown). Переключение через меню или с помощью `Ctrl + M + M` (Markdown), `Ctrl + M + Y` (код).

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

Код выполняется `Shift + Enter`, `Ctrl + Enter` выполняет без перехода, `Alt + Enter` выполняет и добавляет новую ячейку.

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

Файлы можно загружать с компьютера, работать с Google Drive, скачивать файлы командой `!wget`, сохранять результаты в Google Drive.

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

Подключить Google Drive можно через `from google.colab import drive; drive.mount('/content/drive')`.

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

Файлы загружаются командой `from google.colab import files; files.upload()`.

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

Список файлов можно посмотреть командой `!ls` или в панели файлового менеджера в левой части интерфейса.

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

`%time` измеряет время выполнения одной строки, `%%time` измеряет время выполнения всей ячейки, `%timeit` и `%%timeit` выполняют код несколько раз и показывают среднее время выполнения.

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

Изменить аппаратные ресурсы можно через «Среда выполнения» → «Сменить среду выполнения» и выбрать GPU или TPU.

Вывод: в ходе практической работы мы исследовали базовые возможности интерактивных оболочек Jupyter Notebook, JupyterLab и Google Colab для языка программирования Python.