

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

«Построение 3D графиков. Работа с matplotlib Toolkit»

Отчет по лабораторной работе № 3.5

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2023г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

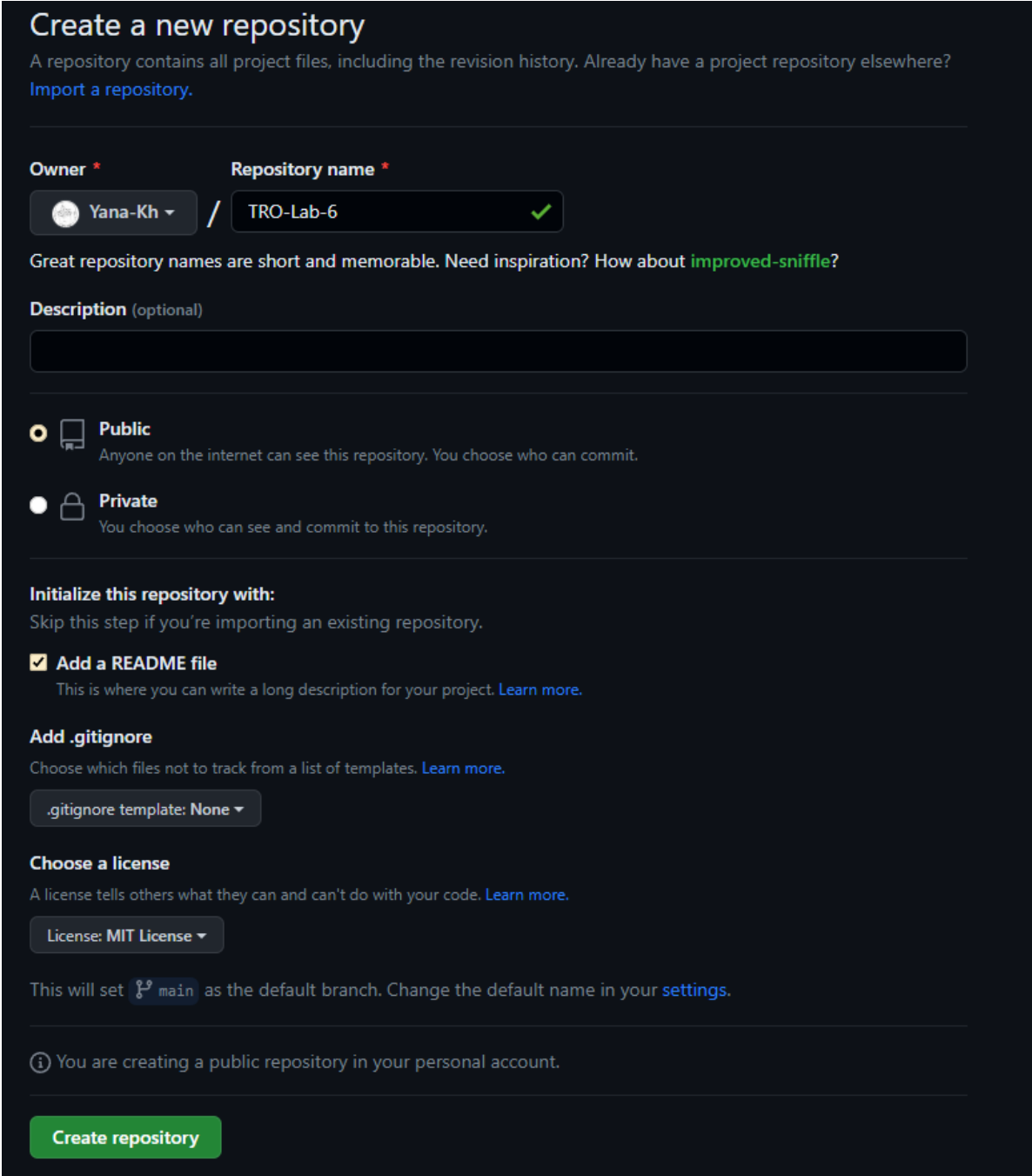
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель работы: исследовать базовые возможности визуализации данных в трехмерном пространстве средствами библиотеки matplotlib языка программирования Python.

Ход работы:


1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 Yana-Kh / TRO-Lab-6 ✓

Great repository names are short and memorable. Need inspiration? How about [improved-sniffle?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.


☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)


Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/TRO-Lab-6.git
Cloning into 'TRO-Lab-6'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование репозитория

4. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-6>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/ynakh/OneDrive/Рабочий стол/Git/TRO-Lab-6/.git/hooks]

C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-6>
```

Рисунок 3 – Организация репозитория в соответствии с моделью git-flow

5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-6>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\TRO-Lab-6>
```

Рисунок 4 – Дополнение файла .gitignore

6. Проработать примеры лабораторной работы.

Лабораторная работа 3.6 Построение 3D графиков. Работа с mplot3d Toolkit

Линейный график

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

```
x = np.linspace(-np.pi, np.pi, 50)
y = x
z = np.cos(x)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y, z, label='parametric curve')
```

```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x299c00bfd0>]
```

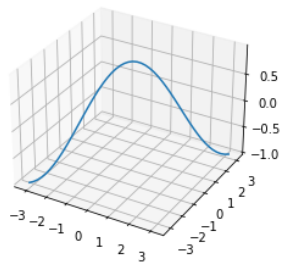


Рисунок 5 – Проработка примеров лабораторной работы

Точечный график

```
np.random.seed(123)
```

```
x = np.random.randint(-5, 5, 40)
y = np.random.randint(0, 10, 40)
z = np.random.randint(-5, 5, 40)
s = np.random.randint(10, 100, 40)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, s=s)
```

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x299c47c5ee0>
```

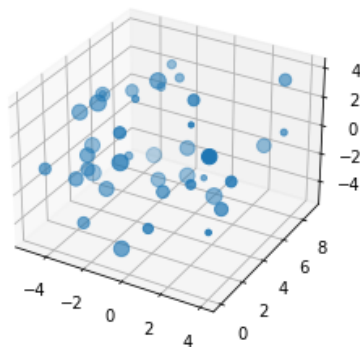


Рисунок 6 – Проработка примеров лабораторной работы

Каркасная поверхность

```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(x, y, z)
```

<mpl_toolkits.mplot3d.art3d.Line3DCollection at 0x299c48d7fa0>

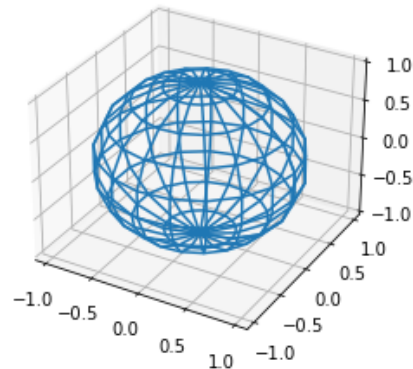


Рисунок 7 – Проработка примеров лабораторной работы

Поверхность

```
: u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z, cmap='inferno')
```

: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x299c4977430>

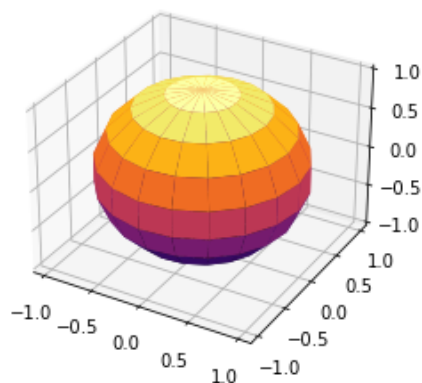


Рисунок 8 – Проработка примеров лабораторной работы

7. Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения трехмерного графика, условие которой предварительно необходимо согласовать с преподавателем.

Задача:

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения трехмерного графика, условие которой предварительно необходимо согласовать с преподавателем.

Задача по математике

Условие

Построить график поверхности заданной функцией

$$f(x, y) = \sin(x^2 + y^2)$$

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def f(x, y):
    return np.sin(x ** 2 + y ** 2)

x = np.linspace(-1, 1, 20)
y = np.linspace(-1, 1, 20)

X, Y = np.meshgrid(x, y)
Z = f(X, Y)

fig = plt.figure()
fig.subplots_adjust(0.1, 0, 1, 20)

ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='inferno')
ax.set_title('График функции');
ax.set_xlabel("X", fontsize=15, labelpad=10)
ax.set_ylabel("Y", fontsize=15, labelpad=10)
ax.set_zlabel("Z", fontsize=15, labelpad=10)
```

```
Text(0.5, 0, 'Z')
```

График функции

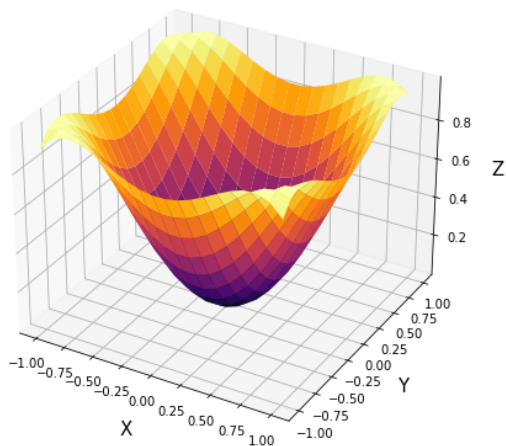


Рисунок 9 – Результат работы

Задача по физике

Условие ¶

Определить давление, при котором 1 м^3 газа, имеющий температуру от 50 до 70 С (с шагом в 2 градуса), содержит $2.4 \cdot 10^{26}$ молекул

Дано: $V = 1\text{ м}^3$, $t = 50 - 70\text{ С}$ $N = 2.4 \cdot 10^{26}$ $p = ?$

Решение задачи:

Запишем формулу связи давления идеального газа с концентрацией молекул и абсолютной температурой:

$$p = nkT$$

В этой формуле n – концентрация молекул, которую легко определить по формуле:

$$n = \frac{N}{V}$$

Также в условии температура дана в Цельсиях. Чтобы перевести температуру в Кельвины, то есть в абсолютную шкалу, нужно воспользоваться формулой:

$$T = t + 273$$

С учетом всего записанного, первая формула станет такой:

$$p = \frac{N}{V} k(t + 273)$$

Здесь k – это постоянная Больцмана, равна $1.38 \cdot 10^{-23}$ Дж/К.

Посчитаем ответ:

```
|: # Объявление переменных
V = 1
t = 50
N = 2.4 * 10 ** (-26)
k = 1.38 * 10 ** (-23)

|: def p(t, V):
    return (N/V * k * (t + 273))

x = np.linspace(50, 70, 10)
y = np.linspace(1, 3, 20)
X, Y = np.meshgrid(x, y)

p = p(X, Y)

fig = plt.figure()
fig.subplots_adjust(0.1,0.1,20)
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(Y, X, p, cmap='inferno')
ax.set_title("Зависимость давления от температуры и объема");
ax.set_xlabel("Объем", fontsize=10, labelpad=10)
ax.set_ylabel("Температура", fontsize=10, labelpad=10)
ax.set_zlabel("Плотность", fontsize=10, labelpad=10)
```

Рисунок 10 – Результат работы

Зависимость давления от температуры и объема

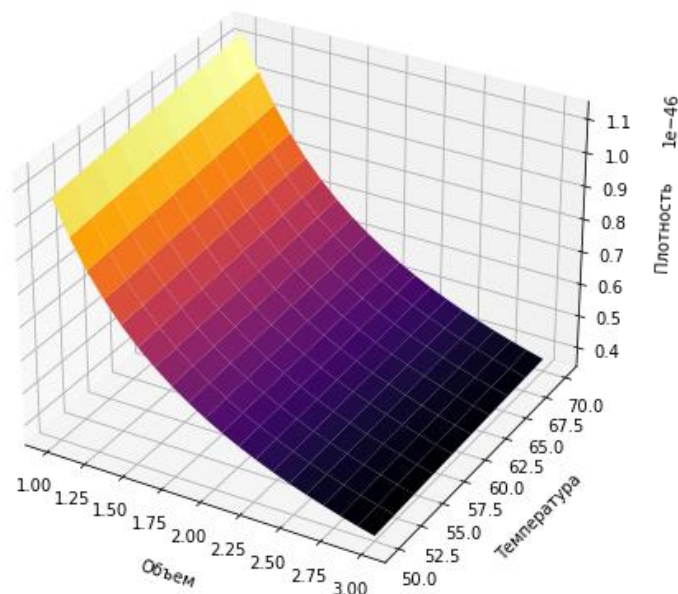


Рисунок 11 – Результат работы

8. Зафиксируйте сделанные изменения в репозитории.

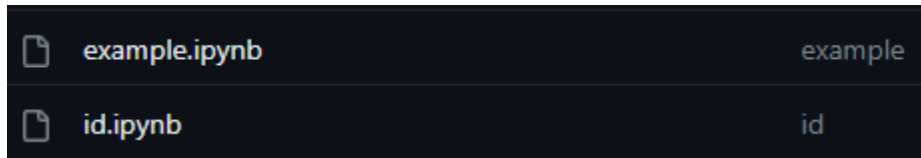


Рисунок 19 – Фиксирование изменений в репозитории

Вопросы для защиты работы

1. Как выполнить построение линейного 3D-графика с помощью matplotlib?

Matplotlib позволяет строить 3D графики. Для этого импортируем необходимые модули для работы с 3D:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Для построения линейного графика используется функция plot().

```
Axes3D.plot(self, xs, ys, *args, zdir='z', **kwargs)
```

где:

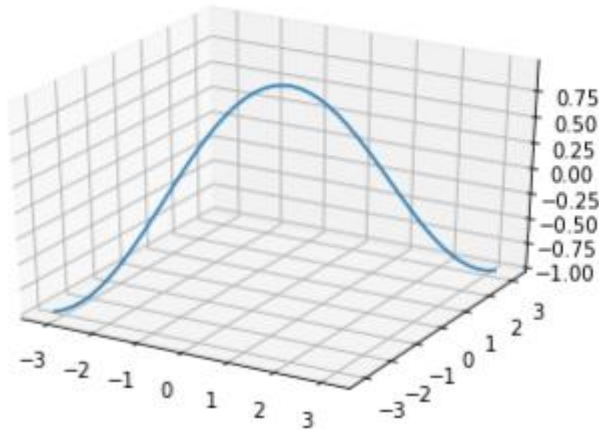
- xs: 1D-массив - x координаты.
- ys: 1D-массив - y координаты.
- zs: скалярное значение или 1D-массив - z координаты. Если передан скаляр, то он будет присвоен всем точкам графика.
- zdir: {'x', 'y', 'z'} - определяет ось, которая будет принята за z направление, значение по умолчанию: 'z'.
- **kwargs - дополнительные аргументы, аналогичные тем, что используются в функции plot() для построения двумерных графиков.

Пример:

```
x = np.linspace(-np.pi, np.pi, 50)
y = x
z = np.cos(x)
```



```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y, z, label='parametric curve')
```



2. Как выполнить построение точечного 3D-графика с помощью matplotlib?

Для построения точечного графика используется функция `scatter()`.

`Axes3D.scatter(self, xs, ys, zs=0, zdir='z', s=20, c=None, depthshade=True, *args, **kwargs)`

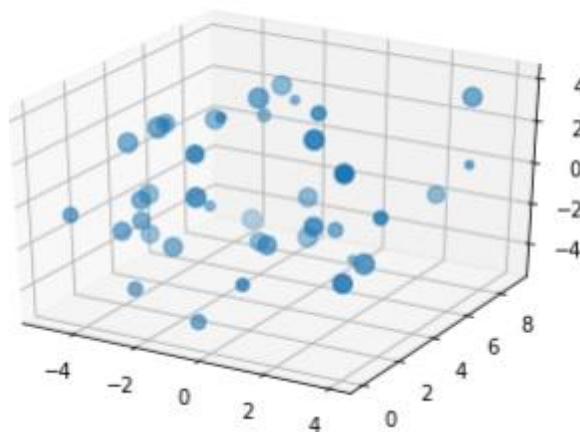
- `xs, ys`: массив - координаты точек по осям `x` и `y`.
- `zs`: `float` или массив, `optional` - координаты точек по оси `z`. Если передан скаляр, то он будет присвоен всем точкам графика. Значение по умолчанию: `0`.
- `zdir`: `{'x', 'y', 'z', '-x', '-y', '-z'}`, `optional` - определяет ось, которая будет принята за `z` направление, значение по умолчанию: `'z'`
- `s`: скаляр или массив, `optional` - размер маркера. Значение по умолчанию: `20`.
- `c`: `color`, массив, массив значений цвета, `optional` - цвет маркера.
- `depthshade`: `bool`, `optional` - затенение маркеров для придания эффекта глубины.
- `**kwargs` - дополнительные аргументы, аналогичные тем, что используются в функции `scatter()` для построения двумерных графиков.

Пример:

```
np.random.seed(123)
x = np.random.randint(-5, 5, 40)
y = np.random.randint(0, 10, 40)
z = np.random.randint(-5, 5, 40)
s = np.random.randint(10, 100, 20)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x, y, z, s=s)
```



3. Как выполнить построение каркасной поверхности с помощью matplotlib?

Для построения каркасной поверхности используется функция `plot_wireframe()`

- `plot_wireframe(self, X, Y, Z, *args, **kwargs)`
- X, Y, Z: 2D-массивы - данные для построения поверхности.
- `rcount, ccount: int` - максимальное количество элементов каркаса, которое будет использовано в каждом из направлений. Значение по умолчанию: 50.

– `rstride, cstride: int` - параметры определяют величину шага, с которым будут браться элементы строки / столбца из переданных массивов. Параметры `rstride, cstride` и `rcount, ccount` являются взаимоисключающими.

– `**kwargs` - дополнительные аргументы, определяемые `Line3DCollection`

Пример:

```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
```

```
x = np.cos(u)*np.sin(v)
```

```
y = np.sin(u)*np.sin(v)
```

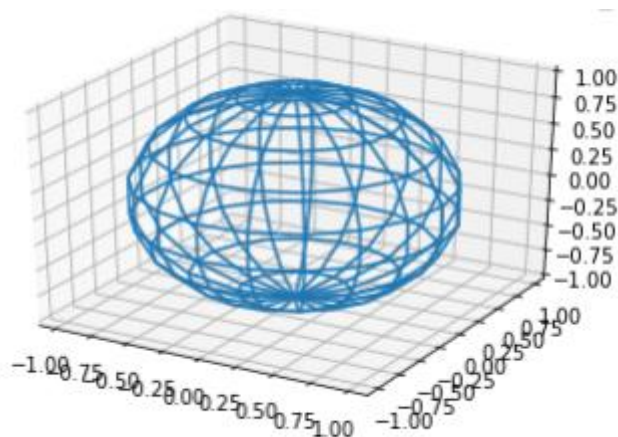
```
z = np.cos(v)
```

```
fig = plt.figure()
```

```
ax = fig.add_subplot(111, projection='3d')
```

```
ax.plot_wireframe(x, y, z)
```

```
ax.legend()
```



4. Как выполнить построение трехмерной поверхности с помощью `matplotlib`?

Для построения поверхности используйте функцию `plot_surface()`.

```
plot_surface(self, X, Y, Z, *args, norm=None, vmin=None, vmax=None, lightsource=None, **kwargs)
```

– `X, Y, Z` : 2D-массивы - данные для построения поверхности.

- rcount, ccount : int - см. rcount, ccount в “Каркасная поверхность”.
- rstride, cstride : int - см. rstride, cstride в “Каркасная поверхность”.
- color: color - цвет для элементов поверхности.
- cmap: Colormap - Colormap для элементов поверхности.
- facecolors: массив элементов color - индивидуальный цвет для каждого элемента поверхности.
- norm: Normalize - нормализация для colormap.
- vmin, vmax: float - границы нормализации.
- shade: bool - использование тени для facecolors. Значение по умолчанию: True.
- lightsource: LightSource - объект класса LightSource – определяет источник света, используется, только если shade = True.
- **kwargs - дополнительные аргументы, определяемые Poly3DCollection