

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное образовательное  
учреждение высшего образования**  
**«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**  
**Кафедра инфокоммуникаций**  
**Институт цифрового развития**

**ОТЧЁТ**

по лабораторной работе №3.6

Дисциплина: «Программирование на Python»

Тема: «Построение 3D графиков. Работа с mplot3d Toolkit»

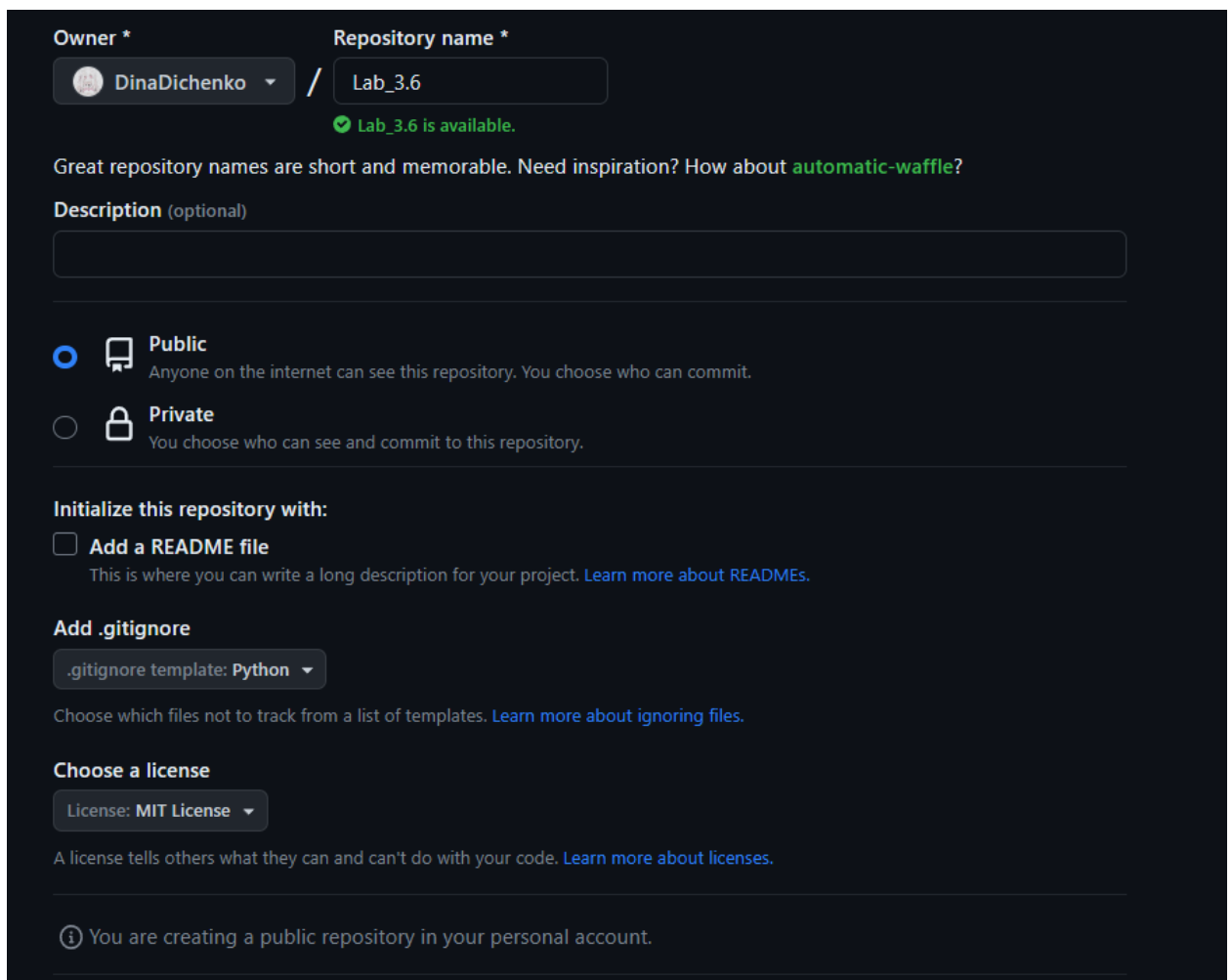
Выполнила: студентка 2 курса,  
группы ИВТ-б-о-21-1  
Диченко Дина Алексеевна

Ставрополь 2023

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

### Практическая часть:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add `.gitignore`).



Owner \* / Repository name \*

DinaDichenko / Lab\_3.6

✔ Lab\_3.6 is available.

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

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:

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

**Add .gitignore**

.gitignore template: **Python**

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

**Choose a license**

License: **MIT License**

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

*i* You are creating a public repository in your personal account.

Рисунок 1. Создание репозитория

2. Выполните клонирование созданного репозитория на рабочий компьютер.

```
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ>git clone https://github.com/DinaDichenko/Lab_3.6.git
Cloning into 'Lab_3.6'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2. Клонирование репозитория

3. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ\Lab_3.6>git flow init
Which branch should be used for bringing forth production releases?
- develop
- main
Branch name for production releases: [main] main
Which branch should be used for integration of the "next release"?
- develop
Branch name for "next release" development: [develop] develop
How to name your supporting branch prefixes?
Feature branches? [feature/] f
Bugfix branches? [bugfix/] b
Release branches? [release/] r
Hotfix branches? [hotfix/] h
Support branches? [support/] s
Version tag prefix? [] v
Hooks and filters directory? [C:/Users/super/OneDrive/Рабочий стол/DI/ВУЗЬ/Lab_3.6/.git/hooks] hf
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ\Lab_3.6>
```

Рисунок 3. Организация репозитория в соответствии с git-flow

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

```
.venv
env/
venv/
ENV/
env.bak/
venv.bak/
.idea

.ipynb_checkpoints
*/.ipynb_checkpoints/*

# IPython
profile_default/
ipython_config.py

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site
```

Рисунок 4. Изменение файла .gitignore

5. Проработайте примеры лабораторной работы в отдельном ноутбуке.

```
In [3]: import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        import numpy as np
```

### Линейный график

```
In [5]: 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')

        plt.show()
```

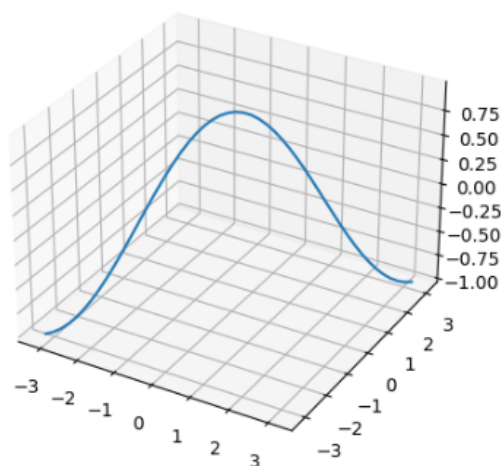


Рисунок 5. Проработка примеров

6. 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.

### Построить график, заданный системой уравнений

$$\begin{cases} x = \cos(u) * u * (1 + \cos(\frac{v}{2})); \\ y = u/2 * \sin(v); \\ z = (\sin(u) * u) * (1 + \cos(\frac{v}{2})) \end{cases}$$

$$0 \leq u \leq 2\pi, 0 \leq v \leq 36\pi$$

```
In [10]: import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

```
In [12]: u, v = np.mgrid[0:2*np.pi:20j, 0:36*np.pi:10j]
x = np.cos(u) * u * (1 + np.cos(v / 2))
y = u / 2 * np.sin(v)
z = (np.sin(u) * u) * (1 + np.cos(v / 2))

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

plt.show()
```

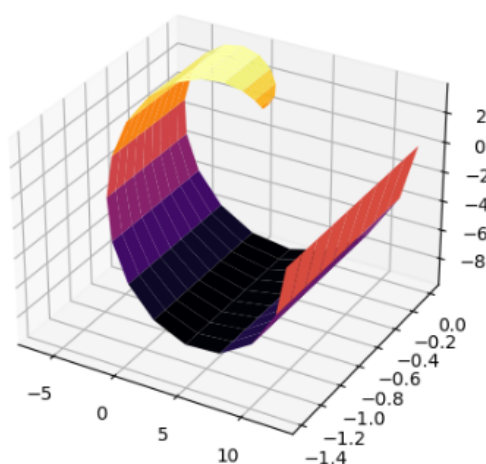


Рисунок 6. Индивидуальное задание

### Контрольные вопросы:

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

Для построения линейного графика используется функция `plot()`.

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

- `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.

## 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. Возможные значения:
  - Строковое значение цвета для всех маркеров.
  - Массив строковых значений цвета.
  - Массив чисел, которые могут быть отображены в цвета через функции `map` и `norm`.
  - 2D массив, элементами которого являются `RGB` или `RGBA`.
- `depthshade`: bool, optional - затенение маркеров для придания эффекта глубины.
- `**kwargs` - дополнительные аргументы, аналогичные тем, что используются в функции `scatter()` для построения двумерных графиков.

## 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.

## 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 (<https://devpractice.ru/matplotlib-lesson-5-1-mplot3d-toolkit/#p3>)".
- rstride, cstride : int - см. *rstride*, *cstride* в "Каркасная поверхность (<https://devpractice.ru/matplotlib-lesson-5-1-mplot3d-toolkit/#p3>)".
- 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*([https://matplotlib.org/api/\\_as\\_gen/mpl\\_toolkits.mplot3d.art3d.Poly3DCollection.html#mpl\\_toolkits.mplot3d.art3d.Poly3DCollection](https://matplotlib.org/api/_as_gen/mpl_toolkits.mplot3d.art3d.Poly3DCollection.html#mpl_toolkits.mplot3d.art3d.Poly3DCollection)).

**Вывод:** в результате выполнения работы были исследованы базовые возможности визуализации данных в трехмерном пространстве средствами библиотеки `matplotlib` языка программирования Python.