**1. Построить 3D график по следующим формулам:**

x=u cos(u) (cos(v)+1)

y=u sin(u) (cos(v)+1)

z=u sin(v)

u Î[0;3π], v Î[-π ; π]

Графиков должно быть два: каркасный и сплошной – разных цветов.

**Программа:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib import cm

# Создаем сетку параметров

u = np.linspace(0, 3\*np.pi, 100)

v = np.linspace(-np.pi, np.pi, 100)

u, v = np.meshgrid(u, v)

# Вычисляем координаты точек поверхности

x = u \* np.cos(u) \* (np.cos(v) + 1)

y = u \* np.sin(u) \* (np.cos(v) + 1)

z = u \* np.sin(v)

# Создаем фигуру с двумя подграфиками

fig = plt.figure(figsize=(12, 6))

# Первый подграфик - каркасный (wireframe)

ax1 = fig.add\_subplot(121, projection='3d')

ax1.plot\_wireframe(x, y, z, color='blue', rstride=5, cstride=5)

ax1.set\_title('Каркасный график')

ax1.set\_xlabel('X')

ax1.set\_ylabel('Y')

ax1.set\_zlabel('Z')

# Второй подграфик - сплошной (surface)

ax2 = fig.add\_subplot(122, projection='3d')

surf = ax2.plot\_surface(x, y, z, cmap=cm.coolwarm,

linewidth=0, antialiased=False)

ax2.set\_title('Сплошной график')

ax2.set\_xlabel('X')

ax2.set\_ylabel('Y')

ax2.set\_zlabel('Z')

# Добавляем цветовую шкалу для сплошного графика

fig.colorbar(surf, ax=ax2, shrink=0.5, aspect=5)

plt.tight\_layout()

plt.show()

**2. Используя возможности библиотеки Matplotlib, постройте график полярной розы в полярной системе координат. Формула розы и ее вид представлен ниже.**

**Программа:**

import numpy as np

import matplotlib.pyplot as plt

t = np.linspace(0, 8\*np.pi, 2000)

fig = plt.figure(figsize=(9,9))

cx = plt.polar(4, 1, 3)

plt.polar(t, np.sin(1.72\*t), 'r-')

plt.grid(True)

plt.show()

**3. Постройте график спирали по формуле: x=t sin(t), y=t cos(t), t Î[0; 5 π]. Оптимальный вариант – использовать возможности библиотеки Matplotlib.**

**Программа:**

import numpy as np

import matplotlib.pyplot as plt

# Создаем массив значений параметра t

t = np.linspace(0, 5\*np.pi, 500)

# Вычисляем координаты x и y

x = t \* np.sin(t)

y = t \* np.cos(t)

# Создаем фигуру и оси

plt.figure(figsize=(8, 8))

ax = plt.subplot(1, 1, 1)

# Рисуем спираль

ax.plot(x, y, 'b-', linewidth=2) # Синяя сплошная линия

# Настраиваем график

ax.set\_title('Спираль: x = t·sin(t), y = t·cos(t)\n t ∈ [0, 5π]')

ax.set\_xlabel('x')

ax.set\_ylabel('y')

ax.grid(True, linestyle='--', alpha=0.7)

ax.axhline(0, color='black', linewidth=0.5)

ax.axvline(0, color='black', linewidth=0.5)

# Устанавливаем равные масштабы по осям

ax.set\_aspect('equal')

plt.show()

**4. Разработайте форму, которая обеспечивает ввод информации о студентах (ФИО, № группы, 3 оценки за сессию) в базу данных. Одна из функций формы – вывод содержимого базы в текстовое окно.**

**Программа:**

import tkinter as tk

from tkinter import messagebox

import sqlite3

class StudentDatabaseApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("База данных студентов")

# Создаем базу данных и таблицу

self.conn = sqlite3.connect('students.db')

self.cursor = self.conn.cursor()

self.cursor.execute('''CREATE TABLE IF NOT EXISTS students

(id INTEGER PRIMARY KEY AUTOINCREMENT,

full\_name TEXT,

group\_number TEXT,

grade1 INTEGER,

grade2 INTEGER,

grade3 INTEGER)''')

self.conn.commit()

# Создаем элементы интерфейса

self.create\_widgets()

def create\_widgets(self):

# Поля для ввода данных

tk.Label(self.root, text="ФИО:").grid(row=0, column=0, padx=5, pady=5, sticky='e')

self.full\_name\_entry = tk.Entry(self.root, width=30)

self.full\_name\_entry.grid(row=0, column=1, padx=5, pady=5)

tk.Label(self.root, text="№ группы:").grid(row=1, column=0, padx=5, pady=5, sticky='e')

self.group\_entry = tk.Entry(self.root, width=30)

self.group\_entry.grid(row=1, column=1, padx=5, pady=5)

tk.Label(self.root, text="Оценка 1:").grid(row=2, column=0, padx=5, pady=5, sticky='e')

self.grade1\_entry = tk.Entry(self.root, width=30)

self.grade1\_entry.grid(row=2, column=1, padx=5, pady=5)

tk.Label(self.root, text="Оценка 2:").grid(row=3, column=0, padx=5, pady=5, sticky='e')

self.grade2\_entry = tk.Entry(self.root, width=30)

self.grade2\_entry.grid(row=3, column=1, padx=5, pady=5)

tk.Label(self.root, text="Оценка 3:").grid(row=4, column=0, padx=5, pady=5, sticky='e')

self.grade3\_entry = tk.Entry(self.root, width=30)

self.grade3\_entry.grid(row=4, column=1, padx=5, pady=5)

# Кнопки

tk.Button(self.root, text="Добавить студента", command=self.add\_student).grid(row=5, column=0, columnspan=2, pady=10)

tk.Button(self.root, text="Показать всех студентов", command=self.show\_students).grid(row=6, column=0, columnspan=2, pady=5)

# Текстовое окно для вывода

self.output\_text = tk.Text(self.root, width=50, height=15)

self.output\_text.grid(row=7, column=0, columnspan=2, padx=5, pady=5)

def add\_student(self):

# Получаем данные из полей ввода

full\_name = self.full\_name\_entry.get()

group\_number = self.group\_entry.get()

try:

grade1 = int(self.grade1\_entry.get())

grade2 = int(self.grade2\_entry.get())

grade3 = int(self.grade3\_entry.get())

except ValueError:

messagebox.showerror("Ошибка", "Оценки должны быть целыми числами")

return

# Проверяем, что все поля заполнены

if not all([full\_name, group\_number]):

messagebox.showerror("Ошибка", "Все поля должны быть заполнены")

return

# Проверяем, что оценки в диапазоне 1-5

if not all(1 <= grade <= 5 for grade in [grade1, grade2, grade3]):

messagebox.showerror("Ошибка", "Оценки должны быть от 1 до 5")

return

# Добавляем студента в базу данных

self.cursor.execute("INSERT INTO students (full\_name, group\_number, grade1, grade2, grade3) VALUES (?, ?, ?, ?, ?)",

(full\_name, group\_number, grade1, grade2, grade3))

self.conn.commit()

# Очищаем поля ввода

self.full\_name\_entry.delete(0, tk.END)

self.group\_entry.delete(0, tk.END)

self.grade1\_entry.delete(0, tk.END)

self.grade2\_entry.delete(0, tk.END)

self.grade3\_entry.delete(0, tk.END)

messagebox.showinfo("Успех", "Студент успешно добавлен")

def show\_students(self):

# Очищаем текстовое окно

self.output\_text.delete(1.0, tk.END)

# Получаем всех студентов из базы данных

self.cursor.execute("SELECT \* FROM students")

students = self.cursor.fetchall()

if not students:

self.output\_text.insert(tk.END, "В базе данных нет студентов")

return

# Выводим данные в текстовое окно

for student in students:

self.output\_text.insert(tk.END,

f"ID: {student[0]}\n"

f"ФИО: {student[1]}\n"

f"Группа: {student[2]}\n"

f"Оценки: {student[3]}, {student[4]}, {student[5]}\n"

f"Средний балл: {sum(student[3:6])/3:.2f}\n"

f"{'-'\*30}\n")

def \_\_del\_\_(self):

# Закрываем соединение с базой данных при закрытии приложения

self.conn.close()

# Создаем и запускаем приложение

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = StudentDatabaseApp(root)

root.mainloop()

**5. Для построения графика функции y=2x2+4 требуются массивы х на отрезке от -5 до 5 с шагом 0.1 и массив y, определяемый по формуле. Составьте скрипт, которые такие массивы создаст (можно, например, с использованием библиотеки NumPy). Определите минимальное значение y в соответствующем массиве.**

**Программа:**

import numpy as np

import matplotlib.pyplot as plt

# Создаём массив x от -5 до 5 с шагом 0.1

x = np.arange(-5, 5 + 0.1, 0.1) # 5 + 0.1 чтобы включить правую границу

# Вычисляем массив y по формуле y = 2x^2 + 4

y = 2 \* x\*\*2 + 4

# Находим минимальное значение y

min\_y = np.min(y)

plt.plot(x, y, 'b-', linewidth=2)

plt.title("График функции y = 2x² + 4")

plt.xlabel("x")

plt.ylabel("y")

plt.grid(True)

plt.show()