**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. 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.**

**import numpy as np**

**x = np.arange(-5,5,0.1)**

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

**print(x,'\n',y)**

**print('min - ',min(y))**

**5. Сгенерируйте массив 5х5 случайных чисел от 1 до 20 и все числа от 3 до 8 замените на 0.**

**import numpy as np**

**a = np.random.randint(1,21,size=(5,5))**

**print(a)**

**for i in range(5):**

**for j in range(5):**

**if 0<=a[i,j]<=8:**

**a[i,j]=0**

**print(a)**

**6. 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()**

**7. Составьте блок-схему и напишите программу тестовой проверки знаний. Программа должна вывести 10 примеров (например, вычисления значений таблицы умножения), задающихся случайным образом и выставить оценку за 10 правильных ответов – «отлично», за 9 и 8 – «хорошо», за 7 и 6 «удовлетворительно», за 6 и менее – «плохо».**

**from random import \***

**ошибки = 0**

**for i in range(10):**

**a = randint(2,10)**

**b = randint(2,10)**

**r = int(input(str(a)+' x '+str(b)+' = '))**

**if r!=a\*b:**

**print('Неверно. Ответ = ',b)**

**ошибки+=1**

**print('Число ошибок: ',ошибки)**

**if ошибки==0: print('отлично')**

**elif 8<=ошибки<=9: print('хорошо')**

**elif 6<=ошибки<=7: print('удовлетворительно')**

**elif ошибки<6: print('плохо')**

**8. Сгенерировать массив 5х5 случайных целых чисел от 1 до 10. Посчитайте, сколько раз каждое из этих чисел повторяется в массиве.**

**import numpy as np**

**a=np.random.randint(1,11,size=(5,5))**

**print(a)**

**частота= {n:0 for n in range(1,11)}**

**print(частота)**

**for i in range(5):**

**for j in range(5):**

**частота[a[i,j]]+=1**

**print(частота)**

**9. Напишите скрипт, содержащий функцию dict\_l(lst), которая принимает аргумент в виде списка и возвращает словарь, в котором каждый элемент списка является и ключом, и значением. Предполагается, что элементы списка будут соответствовать правилам задания ключей в словарях. А скрипт должен проверять, как работает эта функция.**

**def dict\_l(lst):**

**d={}**

**for key in lst:**

**d[key]=key**

**return d**

**print(dict\_l([1,2,3,4,5]))**

**10. Дан следующий набор данных: nm = ['ФИО', ' Возраст', 'Должность', 'Зарплата', 'Стаж'] dt= ['Соловьев Н.С.', '65', 'Директор','150000', '12', 'Брежнев Л.И.', 'Зам. директора', ''43,'120000', '10', 'Коробова С.И.', '34', 'Главный бухгалтер', '100000', '7', 'Старков Н.В.', '35', 'Завхоз', '70000','10', 'Баранова С.А.', '33', 'Зав. складом', '90000', '6']. Составьте скрипт, который содержит две функции. Первая (пусть будет inp) обеспечивает ввод приведенных выше данных в бинарный файл с помощью библиотеки pickle, а вторая функция обеспечивает чтение этих данных с помощью этой же библиотеки и вывода таблицы в консоль в виде таблицы с помощью библиотеки prettytable.**

**import pickle**

**from prettytable import \***

**nm = ['ФИО', 'Возраст', 'Должность', 'Зарплата', 'Стаж']**

**dt= ['Соловьев Н.С.', '65', 'Директор', '150000', '12',**

**'Брежнев Л.И.', '45', 'Зам. директора','120000', '10',**

**'Коробова С.И.', '34', 'Главный бухгалтер', '100000', '7',**

**'Старков Н.В.', '35', 'Завхоз', '70000','10',**

**'Баранова С.А.', '33', 'Зав. складом', '90000', '6']**

**all = [nm,dt]**

**def inp(nm, dt):**

**f = open('all.bin', 'wb')**

**pickle.dump(all, f)**

**f.close()**

**def out():**

**tab = PrettyTable()**

**f = open('all.bin', 'rb')**

**all = pickle.load(f)**

**tab.field\_names= all[0]**

**rows1 = all[1]**

**row=[]**

**rows=[]**

**for i in range(len(rows1)):**

**if i%5==0 and i>0:**

**rows.append(row)**

**row=[]**

**row.append(rows1[i])**

**tab.add\_rows(rows)**

**print(tab)**

**inp(nm, dt)**

**out()**

**11. Поменяйте в словаре d = {1: '5', 2: '6', 3: '7', 4: '8'} ключи и значения местами.**

**Выведите итоговый словарь на экран.**

**a = {1: '5', 2: '6', 3: '7', 4: '8'}**

**b = {i: k for k, i in a.items()}**

**print(b)**

**12. Составьте блок-схему и напишите программу, которая вводит строку символов и заменяет все разделители на символ «\*», причем если между словами несколько разделителей, то должен стоять ОДИН символ «\*». Например, «мама,,,, мыла....раму???» меняется на «мама\*мыла\*раму».**

**s = input('Введите строку: ')**

**r='.,:;!?/\)([]'**

**for i in r:**

**s = s.replace(i,' ')**

**a = s.split(' ')**

**i=0**

**while i<len(a):**

**if a[i]=='': del a[i]**

**else: i+=1**

**b='\*'.join(a)**

**print(b)**

**13. Создайте текстовый файл, содержащий фамилии студентов и их экзаменационные оценки по 5 предметам. Напишите скрипт, печатающий фамилий студентов, сдавших экзамены только на "5".**

**with open('grades.txt', encoding='UTF-8') as f:**

**for line in f:**

**s=line.split(' ')**

**s[-1]=s[-1][0] #убрали \n**

**n=0**

**for i in range(1,len(s)):**

**if s[i]=='5': n+=1**

**if n==5: print(s[0])**

**14. Возьмите любой текстовый файл (чтобы не мучиться с кодировкой, проще сделать его в редакторе Python) с произвольным текстом. Составьте программу, с помощью которой сосчитайте количество каждой из букв алфавита, используемых в тексте (т. е. частоту использования букв).**

**# -\*- coding: utf-8 -\*-**

**буквы = 'йцукенгшщзхъэждлорпавыфячсмитьбю'**

**частоты = {буква: 0 for буква in буквы}**

**## преобразовали строку в словарь**

**file='book1.txt'**

**data=open(file).read()**

**## положили текст из файла в строку data**

**data=data.lower()**

**## привели данные к нижнему регистру**

**##print (data[:200])**

**for c in data:**

**if c in частоты:**

**частоты[c] +=1**

**## если символ есть в словаре, то мы увеличиваем его частоту на 1**

**##print(частоты)**

**частоты = list(частоты.items())**

**## преобразование из словаря в список**

**## частоты.sort(reverse=True, key=f) ## Сортировка по убыванию!**

**частоты.sort(key=lambda x: x[0]) ## Сортировка по убыванию но по буквам!**

**## lambda позволяет создать коротенькие функции**

**for буква, количество in частоты:**

**print('%s - %7d' % (буква, количество))**

**15.Составьте скрип, который рисует черно-белую шахматную доску 8х8.**

**from tkinter import \***

**окно = Tk() # Вывод окна**

**окно.title(" Начало ") # заголовок окна**

**окно.geometry('1200x1024') # размер окна**

**холст = Canvas(окно, bg='white', width=1000, height=1000) # задать холст**

**для рисования**

**холст.pack() # вывести холст**

**x=10; y=10; a=100; color="black"**

**for r in range(8):**

**for i in range(8):**

**холст.create\_rectangle(x,y,x+a,y+a,outline="gray",width=6,fill=color)**

**x+=a**

**if color=="black": color="white"**

**else: color="black"**

**x=10; y+=a;**

**if color=="black": color="white"**

**else: color="black"**

**16. Даны названия должностей всех сотрудников фирмы. Разумеется, в этом списке некоторые должности могут повторяться. Нужно составить штатное расписание, т.е. сделать из имеющегося списка два: в первом – название всех должностей (каждая из должностей должна встречаться в кортеже только один раз), во втором – количество занятых ставок каждой из этих должностей. Список должностей должен быть отсортирован по алфавиту.**

**штат=['директор','бухгалтер','программист','охранник','программист',**

**'менеджер','менеджер','программист','уборщица','уборщица','охранник']**

**штат.sort()**

**i,s=0,[]**

**while i<len(штат):**

**n,j=1,i+1**

**while j<len(штат):**

**if штат[i]==штат[j]:**

**n+=1**

**del штат[j]**

**else: j+=1**

**s.append(n)**

**i+=1**

**print(штат,s)**

**17.Напишите программу печати слов предложения в обратном порядке.**

**s = input('Введите строку: ')**

**# удаление разделителей**

**r='.,:;!?/\)([]'**

**for i in r:**

**s = s.replace(i,' ')**

**# переворачивание**

**a = s.split(' ')**

**i=0**

**while i<len(a):**

**if a[i]=='': del a[i]**

**else: i+=1**

**a.reverse()**

**b=' '.join(a)**

**print(b)**

**18. Напишите программу, которая проверяет, соблюдается ли равенство открывающих и закрывающих скобок в арифметическом выражении языка Python. Выведите на печать исходный текст и количество открывающих и закрывающих скобок.**

**s=input('Задайте формулу: ')**

**i,n=0,0**

**while i<len(s) and n>=0:**

**if s[i]=='(': n+=1**

**elif s[i]==')': n-=1**

**i+=1**

**if n<0: print('Неверный порядок скобок')**

**elif n>0: print('Несовпадает количество скобок')**

**else: print('Все хорошо со скобками')**

**19. 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. JS**

**<!DOCTYPE html>**

**<html lang="ru">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>База данных студентов</title>**

**<style>**

**body {**

**font-family: Arial, sans-serif;**

**margin: 20px;**

**}**

**.form-group {**

**margin-bottom: 10px;**

**}**

**label {**

**display: inline-block;**

**width: 120px;**

**}**

**input {**

**padding: 5px;**

**width: 200px;**

**}**

**button {**

**padding: 8px 15px;**

**background-color: #4CAF50;**

**color: white;**

**border: none;**

**cursor: pointer;**

**margin-right: 10px;**

**}**

**button:hover {**

**background-color: #45a049;**

**}**

**textarea {**

**width: 100%;**

**height: 200px;**

**margin-top: 20px;**

**padding: 10px;**

**}**

**</style>**

**</head>**

**<body>**

**<h1>База данных студентов</h1>**

**<div class="form-group">**

**<label for="fullName">ФИО:</label>**

**<input type="text" id="fullName" required>**

**</div>**

**<div class="form-group">**

**<label for="groupNumber">№ группы:</label>**

**<input type="text" id="groupNumber" required>**

**</div>**

**<div class="form-group">**

**<label for="grade1">Оценка 1:</label>**

**<input type="number" id="grade1" min="1" max="5" required>**

**</div>**

**<div class="form-group">**

**<label for="grade2">Оценка 2:</label>**

**<input type="number" id="grade2" min="1" max="5" required>**

**</div>**

**<div class="form-group">**

**<label for="grade3">Оценка 3:</label>**

**<input type="number" id="grade3" min="1" max="5" required>**

**</div>**

**<button onclick="addStudent()">Добавить студента</button>**

**<button onclick="showStudents()">Показать всех</button>**

**<button onclick="clearDatabase()">Очистить базу</button>**

**<textarea id="output" readonly></textarea>**

**<script>**

**// Инициализация базы данных (если её нет в localStorage)**

**if (!localStorage.getItem('students')) {**

**localStorage.setItem('students', JSON.stringify([]));**

**}**

**// Функция добавления студента**

**function addStudent() {**

**const fullName = document.getElementById('fullName').value;**

**const groupNumber = document.getElementById('groupNumber').value;**

**const grade1 = document.getElementById('grade1').value;**

**const grade2 = document.getElementById('grade2').value;**

**const grade3 = document.getElementById('grade3').value;**

**if (!fullName || !groupNumber || !grade1 || !grade2 || !grade3) {**

**alert("Заполните все поля!");**

**return;**

**}**

**const student = {**

**fullName,**

**groupNumber,**

**grades: [grade1, grade2, grade3]**

**};**

**const students = JSON.parse(localStorage.getItem('students'));**

**students.push(student);**

**localStorage.setItem('students', JSON.stringify(students));**

**alert("Студент добавлен!");**

**document.getElementById('fullName').value = "";**

**document.getElementById('groupNumber').value = "";**

**document.getElementById('grade1').value = "";**

**document.getElementById('grade2').value = "";**

**document.getElementById('grade3').value = "";**

**}**

**// Функция вывода всех студентов**

**function showStudents() {**

**const students = JSON.parse(localStorage.getItem('students'));**

**let output = "";**

**if (students.length === 0) {**

**output = "База данных пуста.";**

**} else {**

**students.forEach((student, index) => {**

**output += `Студент ${index + 1}:\n`;**

**output += `ФИО: ${student.fullName}\n`;**

**output += `Группа: ${student.groupNumber}\n`;**

**output += `Оценки: ${student.grades.join(', ')}\n\n`;**

**});**

**}**

**document.getElementById('output').value = output;**

**}**

**// Функция очистки базы данных**

**function clearDatabase() {**

**if (confirm("Вы уверены, что хотите очистить базу данных?")) {**

**localStorage.setItem('students', JSON.stringify([]));**

**document.getElementById('output').value = "База данных очищена.";**

**}**

**}**

**</script>**

**</body>**

**</html>**