Додаток А

Програмна частина проєкту

Код надсилання листів мовою програмування Python:

```
@"import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.application import MIMEApplication
import os

smtp_server = 'smtp.gmail.com'
smtp_port = 587

smtp_username = 'email_sender'

smtp_password = 'password_smtp'


sender_email = 'email_sender'


#email from db
receiver_email = 'email_receiver'

# Создание сообщения
message = MIMEMultipart()


message['From'] = sender_email
message['To'] = receiver_email

#subject from tb
message['Subject'] = 'email_Subject'
msg_alternative = MIMEMultipart('related')
msg_alternative.attach(MIMEText(body1, ""html""))
message.attach(msg_alternative)

#message.attach(MIMEText(body1, 'html'))
file_paths = [files_PATH]

# Прикрепление файлов к сообщению
for file_path in file_paths:
    with open(file_path, 'rb') as attachment:
        attached_file = MIMEApplication(attachment.read(),
_subtype=os.path.splitext(file_path)[1][1:])
    attached_file.add_header('Content-Disposition', 'attachment',
filename=os.path.basename(file_path))
    message.attach(attached_file)

server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls()  # Начало защищенного соединения
# Аутентификация на SMTP сервере
server.login(smtp_username, smtp_password)

server.sendmail(sender_email, receiver_email, message.as_string())

server.quit()
                              print('Лист було надіслано!')";
```

Програмний код завантаження файлів на Google Drive

```
                        string gdriveapi_py = @"import os
import time
import tkinter as tk
from tkinter import messagebox, filedialog
import pyperclip
from google.oauth2 import service_account
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload
import socket  # Для обработки socket.timeout
from http.client import HTTPException

# Функция для выбора файла с ключом сервисного аккаунта
def select_service_account_key():
    root = tk.Tk()
    root.withdraw()  # Скрыть основное окно
    file_path = filedialog.askopenfilename(
        title='Select service account key file',
        filetypes=(('JSON files', '*.json'), ('All files', '*.*'))
    )
    return file_path

def authenticate():
    SCOPES = ['https://www.googleapis.com/auth/drive.file']
    key_file = select_service_account_key()
    if not key_file:
        print(""No service account key file selected"")
        return None
    creds = service_account.Credentials.from_service_account_file(key_file,
scopes=SCOPES)
    service = build('drive', 'v3', credentials=creds)
    return service

# Функция для создания папки
def create_folder(service, folder_name):
    file_metadata = {
        'name': folder_name,
        'mimeType': 'application/vnd.google-apps.folder',
    }
    folder = service.files().create(body=file_metadata, fields='id').execute()

    return folder.get('id')

# Функция для обновления разрешений
def update_permissions(service, folder_id):
    permission = {
        'type': 'anyone',
        'role': 'reader',
    }
    service.permissions().create(fileId=folder_id, body=permission).execute()

# Функция для загрузки файлов
def upload_files(service, folder_id, file_paths):
    file_ids = []
    for file_path in file_paths:
        file_metadata = {
            'name': os.path.basename(file_path),
            'parents': [folder_id]
        }
        media = MediaFileUpload(file_path)
```

```
        request = service.files().create(body=file_metadata, media_body=media,
fields='id')
        # Установка тайм-аута
        request.http.timeout = 30
        try:
            file = request.execute()
            file_ids.append(file.get('id'))
        except (socket.timeout, ConnectionResetError, HTTPException) as e:
            show_error_message(str(e))
    return file_ids

def get_folder_link(folder_id):
    return ""https://drive.google.com/drive/folders/{}"".format(folder_id)

def copy_link_to_clipboard(link):
    pyperclip.copy(link)

def show_error_message(error_message):
    root = tk.Tk()
    root.withdraw()
    messagebox.showerror(""Помилка"", 'Сталася Помилка: {}\nПожалуйста, попробуйте
позже.'.format(error_message))
    root.destroy()

if __name__ == '__main__':
    service = authenticate()

    if service:
        folder_id = create_folder(service, 'Документи до листа')


        update_permissions(service, folder_id)
        file_paths = [files_PATH]


        file_ids = upload_files(service, folder_id, file_paths)
                if file_ids:
            print('File IDs:', file_ids)

        # Получаем ссылку на папку
        folder_link = get_folder_link(folder_id)
        print('Folder Link:', folder_link)
        root = tk.Tk()
        root.title(""Посилання на гугл-диск папку для представника"")
        link_label = tk.Label(root, text=""Посилання:"")
        link_label.pack()
        link_text = tk.Text(root, height=1, width=50)
        link_text.pack()
        link_text.insert(tk.END, folder_link)

        def copy_link():
            copy_link_to_clipboard(folder_link)
            print(""Link copied to clipboard"")

        copy_button = tk.Button(root, text=""Копіювати"", command=copy_link)
        copy_button.pack()

        root.mainloop()



      ";
```

Програмний код мовою Python для вирішення рівняння методом Гауса

```
import numpy as np

def gaussian_elimination(A, b):
    n = len(b)
    M = np.hstack((A, b.reshape(-1, 1))).astype(float)

    print("Початкова розширена матриця:")
    print(M)
    print()

    for i in range(n):
        max_row = np.argmax(np.abs(M[i:, i])) + i
        M[[i, max_row]] = M[[max_row, i]]

        print(f"Доповнена матриця після заміни рядків {i}:")
        print(M)
        print()

        for j in range(i+1, n):
            factor = M[j, i] / M[i, i]
            M[j, i:] -= factor * M[i, i:]

            print(f"Доповнена матриця після видалення рядка {j}  {i}:")
            print(M)
            print()

    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (M[i, -1] - np.dot(M[i, i+1:n], x[i+1:])) / M[i, i]

        print(f"Вектор рішення після кроку зворотної заміни {n-i}:")
        print(x)
        print()

    return x

A = np.array([
    [2, -1, 0, 0, 0, 0, 0, 0, 0],
    [-1, 2, -1, 0, 0, 0, 0, 0, 0],
    [0, -1, 2, -1, 0, 0, 0, 0, 0],
    [0, 0, -1, 2, -1, 0, 0, 0, 0],
    [0, 0, 0, -1, 2, -1, 0, 0, 0],
    [0, 0, 0, 0, -1, 2, -1, 0, 0],
    [0, 0, 0, 0, 0, -1, 2, -1, 0],
    [0, 0, 0, 0, 0, 0, -1, 2, -1],
    [0, 0, 0, 0, 0, 0, 0, -1, 2]
])
```

```
C_u = np.array([
    [1, 0],
    [0, 1],
    [1, 0],
    [0, 1],
    [1, 0],
    [0, 1],
    [1, 0],
    [0, 1],
    [1, 0]
])
C_v = np.array([
    [1, 0, 1, 0, 1, 0, 1, 0, 1],
    [0, 1, 0, 1, 0, 1, 0, 1, 0]
])
B = np.array([
    [2, -1],
    [-1, 2]
])
f_u = np.array([
    [1],
    [0],
    [1],
    [0],
    [1],
    [0],
    [1],
    [0],
    [1]
])
f_v = np.array([
    [1],
    [0]
])

M = np.block([
    [A, C_u],
    [C_v, B]
])

f = np.vstack([f_u, f_v]).flatten()

solution = gaussian_elimination(M, f)

a = solution[:9]
b = solution[9:]

print("Вектор рішення a:")
print(a)
print()
```

```
print("Вектор рішення b:")
print(b)
print()

a, b
```

Програмний код вікна потабличного перегляду інформації в базі даних

```
  k = 0;
  int rowsCount = dgv_info.Rows.Count;
  if (cb_db_table_select.SelectedIndex == 0)//
  {
      search = "%" + textBox1.Text + "%";
      string pattern = @"^(?!\s*$).+";
      bool isNotEmpty = Regex.IsMatch(textBox1.Text, pattern);
      if ((isNotEmpty == false) || (textBox1.Text == "Фільтрація по коміркам"))
      {
          dgv_info.Columns[0].ReadOnly = true;
          dgv_info.Columns[1].ReadOnly = false;
          dgv_info.Columns[2].ReadOnly = false;
          dgv_info.Columns[3].ReadOnly = true;
          dgv_info.Columns[4].ReadOnly = true;

          Column1.Visible = true;
          Column2.Visible = true;
          Column3.Visible = true;
          Column4.Visible = true;
          Column5.Visible = true;
          Column6.Visible = false;
          Column7.Visible = false;
          Column8.Visible = false;
          Column9.Visible = false;
          Column10.Visible = false;
          Column1.HeaderText = "№ закладу";
          Column2.HeaderText = " Назва";
          Column3.HeaderText = "Тип";
          Column4.HeaderText = "Редаговано";
          Column5.HeaderText = "Дата";
          dgv_info.Rows.Clear();
          institutions.Clear();
          db.Execute<Institutions>("db.db", @"select * from institutions", ref
institutions);

          foreach (var item in institutions)
          {
              dgv_info.Rows.Add(item.I_id, item.I_name, item.I_type, item.U_id,
item.I_date);
          }

      }
      else
      {
          dgv_info.Columns[0].ReadOnly = true;
          dgv_info.Columns[1].ReadOnly = false;
          dgv_info.Columns[2].ReadOnly = false;
          dgv_info.Columns[3].ReadOnly = true;
          dgv_info.Columns[4].ReadOnly = true;
```

```
            Column1.Visible = true;
            Column2.Visible = true;
            Column3.Visible = true;
            Column4.Visible = true;
            Column5.Visible = true;
            Column6.Visible = false;
            Column7.Visible = false;
            Column8.Visible = false;
            Column9.Visible = false;
            Column10.Visible = false;
            Column1.HeaderText = "№ закладу";
            Column2.HeaderText = " Назва";
            Column3.HeaderText = "Тип";
            Column4.HeaderText = "Редаговано";
            Column5.HeaderText = "Дата";
            dgv_info.Rows.Clear();
            institutions.Clear();
            db.Execute<Institutions>("db.db", @"select * from institutions where i_id
like '" + search + "' or i_name like '" + search + "' or i_type like '" + search +
"' or u_id like '" + search + "' or i_date like '" + search + "' ", ref
institutions);

            foreach (var item in institutions)
            {
                dgv_info.Rows.Add(item.I_id, item.I_name, item.I_type, item.U_id,
item.I_date);

            }
        }
    }
    else if (cb_db_table_select.SelectedIndex == 1)//
    {
        search = "%" + textBox1.Text + "%";
        string pattern = @"^(?!\s*$).+";
        bool isNotEmpty = Regex.IsMatch(textBox1.Text, pattern);
        if ((isNotEmpty == false) || (textBox1.Text == "Фільтрація по коміркам"))
        {
            dgv_info.Rows.Clear();
            inst_addr.Clear();
            db.Execute<Inst_address>("db.db", @"select * from inst_address", ref
inst_addr);
            dgv_info.Columns[0].ReadOnly = true;
            dgv_info.Columns[1].ReadOnly = true;
            dgv_info.Columns[2].ReadOnly = false;
            dgv_info.Columns[3].ReadOnly = false;
            dgv_info.Columns[4].ReadOnly = false;
            dgv_info.Columns[5].ReadOnly = false;
            dgv_info.Columns[6].ReadOnly = false;
            dgv_info.Columns[7].ReadOnly = false;
            dgv_info.Columns[8].ReadOnly = true;
            dgv_info.Columns[9].ReadOnly = true;
            Column1.Visible = true;
            Column2.Visible = true;
            Column3.Visible = true;
            Column4.Visible = true;
            Column5.Visible = true;
            Column6.Visible = true;
            Column7.Visible = true;
            Column8.Visible = true;
            Column9.Visible = true;
            Column10.Visible = true;
            Column1.HeaderText = "№ адреси";
            Column2.HeaderText = "№ закладу";
            Column3.HeaderText = "Вулиця";
            Column4.HeaderText = "Район";
```

```
          Column5.HeaderText = "Тип міста";
          Column6.HeaderText = "Місто";
          Column7.HeaderText = "Область";
          Column8.HeaderText = "Поштовий індекс";
          Column9.HeaderText = "Редаговано";
          Column10.HeaderText = "Дата";
          foreach (var item in inst_addr)
          {

              dgv_info.Rows.Add(item.I_a_id, item.I_id, item.I_a_st,
item.I_a_district, item.I_a_city_type, item.I_a_city, item.I_a_region,
item.I_a_post_index, item.U_id, item.I_a_date);
          }
      }
      else
      {
          dgv_info.Rows.Clear();
          dgv_info.Columns[0].ReadOnly = true;
          dgv_info.Columns[1].ReadOnly = true;
          dgv_info.Columns[2].ReadOnly = false;
          dgv_info.Columns[3].ReadOnly = false;
          dgv_info.Columns[4].ReadOnly = false;
          dgv_info.Columns[5].ReadOnly = false;
          dgv_info.Columns[6].ReadOnly = false;
          dgv_info.Columns[7].ReadOnly = false;
          dgv_info.Columns[8].ReadOnly = true;
          dgv_info.Columns[9].ReadOnly = true;
          Column1.Visible = true;
          Column2.Visible = true;
          Column3.Visible = true;
          Column4.Visible = true;
          Column5.Visible = true;
          Column6.Visible = true;
          Column7.Visible = true;
          Column8.Visible = true;
          Column9.Visible = true;
          Column10.Visible = true;
          Column1.HeaderText = "№ адреси";
          Column2.HeaderText = "№ закладу";
          Column3.HeaderText = "Вулиця";
          Column4.HeaderText = "Район";
          Column5.HeaderText = "Тип міста";
          Column6.HeaderText = "Місто";
          Column7.HeaderText = "Область";
          Column8.HeaderText = "Поштовий індекс";
          Column9.HeaderText = "Редаговано";
          Column10.HeaderText = "Дата";
          inst_addr.Clear();
          db.Execute<Inst_address>("db.db", @"select * from inst_address where
i_a_id like '" + search + "' or i_id like '" + search + "' or i_a_st like '" +
search + "' or i_a_district like '" + search + "' or i_a_city_type like '" + search
+ "' or i_a_city like '" + search + "' or i_a_region like '" + search + "' or
i_a_post_index like '" + search + "' or u_id like '" + search + "' or i_a_date like
'" + search + "'", ref inst_addr);

          foreach (var item in inst_addr)
          {

              dgv_info.Rows.Add(item.I_a_id, item.I_id, item.I_a_st,
item.I_a_district, item.I_a_city_type, item.I_a_city, item.I_a_region,
item.I_a_post_index, item.U_id, item.I_a_date);
          }
      }
  }
  else if (cb_db_table_select.SelectedIndex == 2)//
```

```
  {
      search = "%" + textBox1.Text + "%";
      string pattern = @"^(?!\s*$).+";
      bool isNotEmpty = Regex.IsMatch(textBox1.Text, pattern);
      if ((isNotEmpty == false) || (textBox1.Text == "Фільтрація по коміркам"))
      {
          dgv_info.Rows.Clear();
          inst_em.Clear();
          db.Execute<Inst_email>("db.db", @"select * from inst_email", ref inst_em);
dgv_info.Columns[0].ReadOnly = true;
          dgv_info.Columns[0].ReadOnly = true;
          dgv_info.Columns[1].ReadOnly = true;
          dgv_info.Columns[2].ReadOnly = false;
          dgv_info.Columns[3].ReadOnly = true;
          dgv_info.Columns[4].ReadOnly = true;
          Column1.Visible = true;
          Column2.Visible = true;
          Column3.Visible = true;
          Column4.Visible = true;
          Column5.Visible = true;
          Column6.Visible = false;
          Column7.Visible = false;
          Column8.Visible = false;
          Column9.Visible = false;
          Column10.Visible = false;
          Column1.HeaderText = "№ ел. пошти";
          Column2.HeaderText = "№ закладу";
          Column3.HeaderText = "Електронна пошта";
          Column4.HeaderText = "Редаговано";
          Column5.HeaderText = "Дата";          foreach (var item in inst_em)
          {              dgv_info.Rows.Add(item.I_e_id, item.I_id, item.I_e_email,
item.U_id, item.I_e_date);
          }      }      else      {          dgv_info.Columns[0].ReadOnly = true;
          dgv_info.Columns[1].ReadOnly = true;
          dgv_info.Columns[2].ReadOnly = false;
          dgv_info.Columns[3].ReadOnly = true;
          dgv_info.Columns[4].ReadOnly = true;
          Column1.Visible = true;
          Column2.Visible = true;
          Column3.Visible = true;
          Column4.Visible = true;
          Column5.Visible = true;
          Column6.Visible = false;
          Column7.Visible = false;
          Column8.Visible = false;
          Column9.Visible = false;
          Column10.Visible = false;
          Column1.HeaderText = "№ ел. пошти";
          Column2.HeaderText = "№ закладу";
          Column3.HeaderText = "Електронна пошта";
          Column4.HeaderText = "Редаговано";
          Column5.HeaderText = "Дата";
          dgv_info.Rows.Clear();
          inst_em.Clear();
          db.Execute<Inst_email>("db.db", @"select * from inst_email where i_e_id
like '" + search + "' or i_id like '" + search + "' or i_e_email like '" + search +
"' or u_id like '" + search + "' or i_e_date like '" + search + "'", ref inst_em);
dgv_info.Columns[0].ReadOnly = true;

          foreach (var item in inst_em)
          {
              dgv_info.Rows.Add(item.I_e_id, item.I_id, item.I_e_email, item.U_id,
item.I_e_date);
          }      }
                                              }
```

Програмний код обробника авторизації в систему

```
line_u_id = @"SELECT u_id from users where  u_login=" + tbLogin.Text + "@";
int k = 0;

if ((tbLogin.Text == "Введіть логін") && (tbPass.Text == ""))
{
    MessageBox.Show("Ви не ввели ні логін ні пароль");
}
else if ((tbLogin.Text == "") && (tbPass.Text == "Введіть пароль"))
{
    MessageBox.Show("Ви не ввели ні логін ні пароль");
}
else if ((tbLogin.Text == "") && (tbPass.Text == ""))
{
    MessageBox.Show("Ви не ввели ні логін ні пароль");
}
else if ((tbLogin.Text == "Введіть логін") && (tbPass.Text == "Введіть пароль"))
{
    MessageBox.Show("Ви не ввели ні логін ні пароль");
}
else if (((tbLogin.Text == "") || (tbLogin.Text == "Введіть логін")) &&
((tbPass.Text != "") || (tbPass.Text != "Введіть пароль")))
{
    MessageBox.Show("Ви не ввели логін!");
}
else if (((tbLogin.Text != "") || (tbLogin.Text != "Введіть логін")) &&
((tbPass.Text == "") || (tbPass.Text == "Введіть пароль")))
{
    MessageBox.Show("Ви не ввели пароль!");
}
else
{
    for (int i = 0; i < users.Count; i++)
    {
        if (tbLogin.Text == users[i].U_login && tbPass.Text == users[i].U_pass)
        {
            if (users[i].U_lvl == "no")
            {

                k++;
                FormIADD IADD = new FormIADD();

                IADD.crypt_pass = crypt_pass;
                this.Hide();
                IADD.Show();
                IADD.IdUser = users[i].U_id;

                return;

            }

        }
    }
    if (k <= 0)
    {
        MessageBox.Show("Не вірний логін або пароль!");
    }
                                            }
```

Програмний код обробника реєстрації в вікні створення профілю

```
  private void btAuthor_Click(object sender, EventArgs e)
  {
      string temp = "";
      string pattern = @"^(?!\s*$).+";
      bool isNotEmpty1 = Regex.IsMatch(tbLogin.Text, pattern);
      bool isNotEmpty2 = Regex.IsMatch(tbPass.Text, pattern);
      bool isNotEmpty3 = Regex.IsMatch(textBox1.Text, pattern);
      switch (comboBox1.SelectedItem.ToString())
      {
          case "Ваша перша домашня тварина":
              temp = "Ваша перша домашня тварина";
              break;
          case "Місто де ви народились":
              temp = "Місто де ви народились";
              break;
          case "Ваше прізвисько в дитинстві":
              temp = "Ваше прізвисько в дитинстві";
              break;
          case "Місто, де зустрілись ваші Батьки":
              temp = "Місто, де зустрілись ваші Батьки";
              break;
          case "Ім'я вашого родича":
              temp = "Ім'я вашого родича";
              break;
          case "Назва Вашої першої школи":
              temp = "Ім'я вашого родича";
              break;
          default:
              MessageBox.Show("Оберіть запитання");
              break;
      }      int k = 0;
      if (((tbLogin.Text != "Введіть логін" )&&(isNotEmpty1)) &&
         ((tbPass.Text != "Введіть пароль" ) && (isNotEmpty2)) &&
         ((textBox1.Text != "Введіть відповідь ...") && (isNotEmpty3)))
      {
          for (int i = 0; i < users.Count; i++)
          {
              if (tbLogin.Text == users[i].U_login)
              {
                  k++;
              }          }          if (k <= 0)          {
              string q = @"Insert into users(u_login,
u_pass,u_secret,u_answ,u_lvl,theme)
              values('" + tbLogin.Text + @"','" + tbPass.Text + @"','" + temp +
@"','" + textBox1.Text + @"','" + "no" + @"','"+"1"+@"');";
              db.ExecuteNonQuery("db.db", q, 0);

              MessageBox.Show("Користувача '"+tbLogin.Text+"' зареєстровано." );
              string logincreated = tbLogin.Text;
              FormLoginPass flp = new FormLoginPass();
              flp.active = 1;
              flp.login1 = logincreated;
              flp.Show();
              this.Hide();                        }          else
          {
              MessageBox.Show("Такий логін вже використовується!");
          }
      }
      else
      {          MessageBox.Show("Недостатньо даних для реєстрації");      }  }
```

Програмний код класу підключення до локальної бази даних

```
using Devart.Data.SQLite;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace StudyEnglish
{
   public class ClassDataBase
   {
      #region ExecuteNonQuery
      public int ExecuteNonQuery(string connectionToDBString, string sSql, int
where)
      {
         int n = 0;
         try
         {
            using (SQLiteConnection con = new
SQLiteConnection(string.Format("Data Source=db.db;Pooling=false",
connectionToDBString)))
            {
               con.Open();
               using (SQLiteCommand sqlCommand = con.CreateCommand())
               {
                  sqlCommand.CommandText = sSql;
                  n = sqlCommand.ExecuteNonQuery();
               }
               con.Close();
            }
         }
         catch (Exception ex)
         {
            n = 0;
         }
         return n;
      }
      #endregion
      #region Execute
      protected T GetObject<T>(params object[] args)
```

```
        {
            return (T)Activator.CreateInstance(typeof(T), args);
        }

        public void Execute<T>(string connectionToDBString, string sSql, ref
List<T> listResult)
        {
            string result = "";
            try
            {

                SQLiteConnection con = new SQLiteConnection(string.Format("Data
Source=db.db;Pooling=false", connectionToDBString));
                con.Open();
                SQLiteCommand command = new SQLiteCommand(sSql, con);
                SQLiteDataReader dataReader = command.ExecuteReader();

                if (dataReader.HasRows)
                {
                    while (dataReader.Read())
                    {
                        result = "";
                        for (int i = 0; i < dataReader.FieldCount; i++)
                        {
                            try
                            {
                                result += dataReader.GetString(i) + "!";
                            }
                            catch { result += " !"; }
                        }
                        if (result.Count() > 2) result = result.Remove(result.Count() - 1);
                        if (result != "") listResult.Add(GetObject<T>(result));
                    }
                }

                con.Close();
            }
            catch (Exception ex)
            {
            }
        }
        #endregion
    }
}
```

Програмний код для впровадження методів роботи з мовою програмування Python

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Common;
using Python.Runtime;
using System.Threading;
using System.IO;

namespace DB
{
    public class PythonInterop
    {
        public static void Initialize()
        {
            string pythonDll = Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"python38.dll");

            Environment.SetEnvironmentVariable("PYTHONNET_PYDLL", pythonDll);
            PythonEngine.Initialize();
        }
        public static void RunPythonCode(string pycode, object parameter, string
parameterName)
        {

            Initialize();
            using (Py.GIL())
            {
                PythonEngine.RunSimpleString(pycode);
                using (var scope = Py.CreateScope())
                {
                    scope.Set(parameterName, parameter.ToPython());
                    scope.Exec(pycode);
                }
            }

        }

        public static object RunPythonCodeAndReturn(string pycode, object parameter,
string parameterName, string returnedVariableName)
        {
            object returnedVariable = new object();
            Initialize();
            using (Py.GIL())
            {
                using (var scope = Py.CreateScope())
                {
                    scope.Set(parameterName, parameter.ToPython());
                    scope.Exec(pycode);
                    returnedVariable = scope.Get<object>(returnedVariableName);
                }
            }
            return returnedVariable;
        }

    }
}
```

ДОДАТОК Б

Демонстраційний графічний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК 502900.009 ПЗ	Пояснювальна записка	1	72 с.
			Інші документи		
	2		Пояснювальна записка	1	Ел. варіант
			Додаток А	1	Ел. варіант
			Програмна частина системи		
			Додаток Б		
			Демонстраційний графічний матеріал	1	17 с.

					ГЮІК 502900.009 ВД			
Змін.	Арк	Номер докум.	Підп	Дата				
Розроб.		Півень Н. П.			Розробка системи автоматизації електронного листування з використанням захищених локальних баз даних	Літера	Арку	Аркуш
Переві		Ромашов Ю. В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.конт		Демська Н. П.						
Затв.		Невлюдов І.Ш.						