Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра АСУ



**Звіт**

до виконаної лабораторної роботи №7

з дисципліни “Об’єктно-орієнтоване програмування”

на тему:

**“: Розробка інтерфейсу користувача і опрацювання зображень засобами Python ”**

Виконав

студент групи КН-118

Cтельмах Віталій

Прийняв

Цимбал Ю.В.

Львів – 2021

*Лабораторна робота № 7*

*Тема роботи:* **Алгоритми розв’язання лабіринтних та ігрових задач**

***Мета роботи:*** ознайомитись з можливостями бібліотек мови Python для синтезу та опрацювання зображень, оволодіти практичними прийомами побудови графічного інтерфейсу користувача.

***Завдання*.** Дляпрограми розв’язання лабіринтної (ігрової) задачі, створеної у лабораторній роботі № 6, реалізувати віконний графічний інтерфейс користувача з візуалізацією вхідних даних та результатів.

***Послідовність роботи:***

1. Ознайомитись з теоретичним матеріалом про подання лабіринтних та ігрових задач у вигляді графових моделей.

2. Розглянути можливості пакету Gtaphviz, бібліотек PyDot та PyGraphviz для опису та візуалізації графів.

3. Розглянути можливості бібліотек PIL, Pillow та ін. для синтезу та опрацювання зображень.

4. Розглянути можливості API PyQt для створення графічного інтерфейсу користувача, роботи з векторною та растровою графікою.

5. Вибрати і встановити пакети, API та додаткові бібліотеки мови Python, необхідні для виконання завдання.

6. Спроектувати і реалізувати віконний графічний інтерфейс для програми з ЛР № 6.

Передбачити можливості:

* подання вхідних даних у вигляді графових моделей, наприклад, у форматі DOT;
* відображення та редагування графової моделі вхідних даних у текстовому вікні;
* візуалізації графової моделі вхідних даних у вигляді зображення і її збереження у файлі;
* демонстрації процесу і (або) результатів розв’язання задачі;
* відображення результатів розв’язання у вигляді окремого зображення, або зміненого початкового зображення графової моделі, збереження його у файлі;
* зручний доступ користувача до вказаних можливостей за допомогою елементів керування віконного інтерфейсу.

Програма повинна коректно працювати для різних прикладів вхідних даних (не лише для наведеного у завданні до ЛР № 6) і допускати масштабованість задачі, наприклад, збільшення розміру вхідних даних. При розробці прагнути відділити реалізацію класів, які відповідають за інтерфейс користувача від реалізації класів для знаходження розв’язку задачі, продумати і реалізувати взаємодію між цими класами та їх об’єктами.

7. Зробити висновок про ефективність та універсальність застосованих програмних засобів при розв’язанні даної задачі.

8. Оформити звіт про роботу за такою структурою:

* назва роботи;
* мета роботи;
* послідовність роботи;
* індивідуальне завдання (з ЛР № 6);
* тексти програмних модулів;
* формат файлів з вхідними даними та результатами;
* інструкція користувачу, разом з описом дій для встановлення необхідних пакетів та бібліотек;
* контрольний приклад та результати роботи програми, зокрема, у вигляді екранограм;
* висновки.

**Завдання**

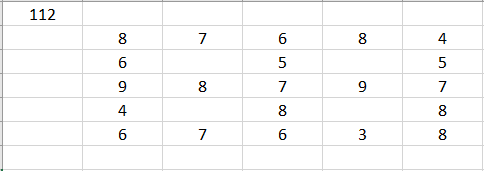
20. Від верхнього лівого квадрата прокладіть такий шлях до нижнього правого квадрата, щоб сума чисел, записаних у проміжних квадратах, а також у стартовому та фінішному, складала 112. (Шуканий маршрут може перетинати лише сторони, але не вершини проміжних квадратів і проходити через кожен проміжний квадрат лише один раз).



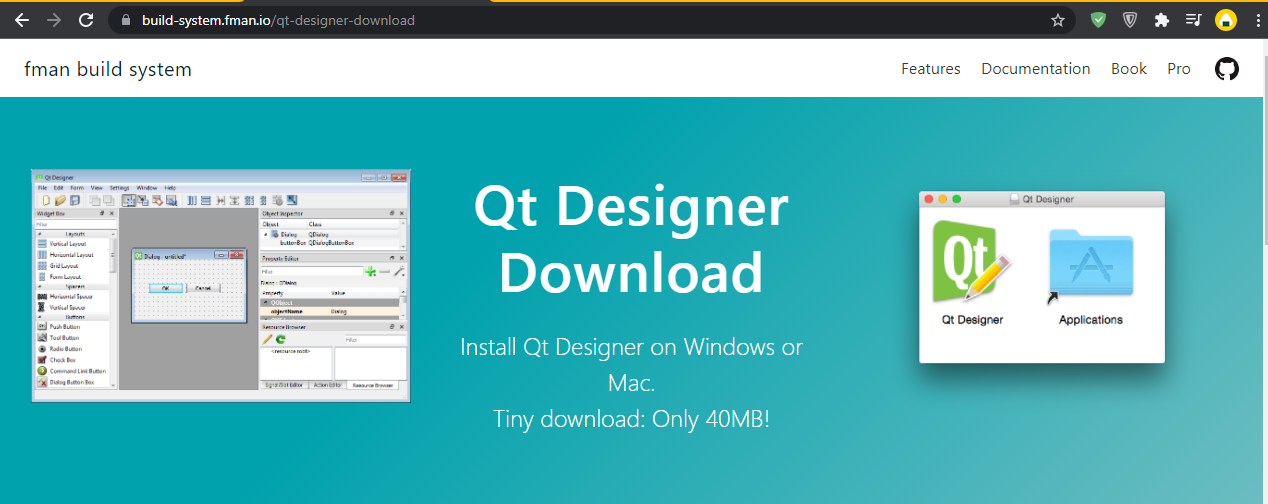
**Текст програми**

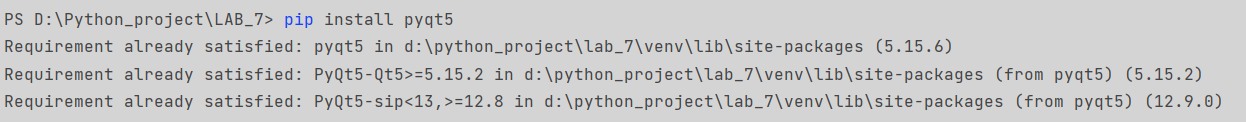
from PyQt5 import QtCore, QtGui, QtWidgets  
from PyQt5.QtWidgets import QMessageBox  
  
  
import easygui as eg  
import xlrd  
import xlsxwriter as xlwt  
import sys  
  
class AlignDelegate(QtWidgets.QStyledItemDelegate):  
 def initStyleOption(self, option, index):  
 super(AlignDelegate, self).initStyleOption(option, index)  
 option.displayAlignment = QtCore.Qt.AlignCenter  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(741, 692)  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.Load = QtWidgets.QPushButton(self.centralwidget)  
 self.Load.setGeometry(QtCore.QRect(520, 120, 181, 71))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.Load.setFont(font)  
 self.Load.setObjectName("Load")  
 self.Save = QtWidgets.QPushButton(self.centralwidget)  
 self.Save.setGeometry(QtCore.QRect(520, 210, 181, 71))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.Save.setFont(font)  
 self.Save.setObjectName("Save")  
 self.Save\_txt = QtWidgets.QPushButton(self.centralwidget)  
 self.Save\_txt.setGeometry(QtCore.QRect(520, 490, 181, 71))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.Save\_txt.setFont(font)  
 self.Save\_txt.setObjectName("Save\_txt")  
 self.Search = QtWidgets.QPushButton(self.centralwidget)  
 self.Search.setGeometry(QtCore.QRect(520, 300, 181, 71))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.Search.setFont(font)  
 self.Search.setObjectName("Search")  
 self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)  
 self.tableWidget.setGeometry(QtCore.QRect(20, 110, 451, 271))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.tableWidget.setFont(font)  
 self.tableWidget.setRowCount(0)  
 self.tableWidget.setObjectName("tableWidget")  
 self.tableWidget.setColumnCount(0)  
 self.tableWidget.horizontalHeader().setDefaultSectionSize(67)  
 self.tableWidget.verticalHeader().setSortIndicatorShown(False)  
 self.plainTextEdit = QtWidgets.QPlainTextEdit(self.centralwidget)  
 self.plainTextEdit.setGeometry(QtCore.QRect(40, 420, 421, 251))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.plainTextEdit.setFont(font)  
 self.plainTextEdit.setObjectName("plainTextEdit")  
 self.Control\_sum = QtWidgets.QLineEdit(self.centralwidget)  
 self.Control\_sum.setGeometry(QtCore.QRect(20, 40, 131, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.Control\_sum.setFont(font)  
 self.Control\_sum.setObjectName("Control\_sum")  
 self.label = QtWidgets.QLabel(self.centralwidget)  
 self.label.setGeometry(QtCore.QRect(20, 10, 211, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.label.setFont(font)  
 self.label.setObjectName("label")  
 self.label\_2 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_2.setGeometry(QtCore.QRect(150, 80, 211, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.label\_2.setFont(font)  
 self.label\_2.setObjectName("label\_2")  
 self.label\_3 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_3.setGeometry(QtCore.QRect(200, 390, 81, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.label\_3.setFont(font)  
 self.label\_3.setObjectName("label\_3")  
 self.spinBoxRow = QtWidgets.QSpinBox(self.centralwidget)  
 self.spinBoxRow.setGeometry(QtCore.QRect(230, 40, 111, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.spinBoxRow.setFont(font)  
 self.spinBoxRow.setObjectName("spinBoxRow")  
 self.label\_4 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_4.setGeometry(QtCore.QRect(230, 10, 211, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.label\_4.setFont(font)  
 self.label\_4.setObjectName("label\_4")  
 self.label\_5 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_5.setGeometry(QtCore.QRect(420, 10, 211, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.label\_5.setFont(font)  
 self.label\_5.setObjectName("label\_5")  
 self.spinBoxColumn = QtWidgets.QSpinBox(self.centralwidget)  
 self.spinBoxColumn.setGeometry(QtCore.QRect(420, 40, 111, 31))  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.spinBoxColumn.setFont(font)  
 self.spinBoxColumn.setObjectName("spinBoxColumn")  
 MainWindow.setCentralWidget(self.centralwidget)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 self.add\_functions()  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "Lab\_7"))  
 self.Load.setText(\_translate("MainWindow", "Load Excel"))  
 self.Save.setText(\_translate("MainWindow", "Save Excel"))  
 self.Save\_txt.setText(\_translate("MainWindow", "Save\_txt"))  
 self.Search.setText(\_translate("MainWindow", "Search"))  
 self.label.setText(\_translate("MainWindow", "Control Sum"))  
 self.label\_2.setText(\_translate("MainWindow", "Matrix - Labyrinth"))  
 self.label\_3.setText(\_translate("MainWindow", "Result"))  
 self.label\_4.setText(\_translate("MainWindow", "Row Count"))  
 self.label\_5.setText(\_translate("MainWindow", "Column Count"))  
  
 def add\_functions(self):  
 self.Load.clicked.connect(self.load)  
 self.Search.clicked.connect(self.search)  
 self.Save.clicked.connect(self.save\_xlsx)  
 self.Save\_txt.clicked.connect(self.save\_txt)  
 self.spinBoxRow.valueChanged.connect(self.row\_in\_table)  
 self.spinBoxColumn.valueChanged.connect(self.column\_in\_table)  
  
 def load(self):  
 file = File()  
 a = eg.fileopenbox()  
 if a!=None:  
 file.set\_workbook(a)  
 file.set\_sheet(0)  
 mass\_from\_file = file.from\_file\_to\_prog()[0]  
 rez\_from\_file = file.from\_file\_to\_prog()[1]  
 self.Control\_sum.setText(str(rez\_from\_file))  
 self.tableWidget.setRowCount(len(mass\_from\_file))  
 self.tableWidget.setColumnCount(len(mass\_from\_file[0]))  
 for i in range(len(mass\_from\_file)):  
 for j in range(len(mass\_from\_file[0])):  
 self.tableWidget.setItem(i, j,QtWidgets.QTableWidgetItem(str(mass\_from\_file[i][j])))  
  
 for i in range(self.tableWidget.rowCount()):  
 delegate = AlignDelegate(self.tableWidget)  
 self.tableWidget.setItemDelegateForRow(i, delegate)  
  
 self.spinBoxRow.setValue(self.tableWidget.rowCount())  
 self.spinBoxColumn.setValue(self.tableWidget.columnCount())  
  
 def search(self):  
 mas = []  
 self.plainTextEdit.clear()  
 rez.clear()  
 path.clear()  
 for i in range(self.tableWidget.rowCount()):  
 path.append([])  
 for j in range(self.tableWidget.columnCount()):  
 path[i].append(int(0))  
  
 alg = Algoritm()  
 if self.Control\_sum.text()=="":  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Warning)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("Control sum fiels is empty")  
 msb.exec\_()  
 else:  
 control\_sum=int(self.Control\_sum.text())  
 if self.tableWidget.rowCount()==0 or self.tableWidget.columnCount()==0:  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Warning)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("Matrix is empty")  
 msb.exec\_()  
 else:  
 for i in range(self.tableWidget.rowCount()):  
 mas.append([])  
 for j in range(self.tableWidget.columnCount()):  
 if self.tableWidget.item(i,j)==None or self.tableWidget.item(i,j).text()=="":  
 self.tableWidget.setItem(i,j,QtWidgets.QTableWidgetItem(self.Control\_sum.text()))  
 mas[i].append(int(self.tableWidget.item(i,j).text()))  
 alg.set\_mass(mas)  
 alg.set\_final\_rez(control\_sum)  
 alg.step(0, 0, 0)  
 for i in range(len(rez)):  
 self.plainTextEdit.appendPlainText(rez[i])  
 if self.plainTextEdit.toPlainText()=="":  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Warning)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("No solutions found")  
 msb.exec\_()  
  
 def save\_txt(self):  
 if self.plainTextEdit.toPlainText()!="":  
 file = File()  
 mass1=[]  
 mass1.append(self.plainTextEdit.toPlainText())  
 a = eg.filesavebox()  
 if a!=None:  
 s = a + ".txt"  
 file.prog\_to\_file(s, mass1)  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Information)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("File Save")  
 msb.exec\_()  
 else:  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Warning)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("plainTextEdit is empty")  
 msb.exec\_()  
  
 def save\_xlsx(self):  
 if self.tableWidget.rowCount() == 0 or self.tableWidget.columnCount() == 0:  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Warning)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("Matrix is empty")  
 msb.exec\_()  
 elif self.Control\_sum.text()=="":  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Warning)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("Control sum fiels is empty")  
 msb.exec\_()  
 else:  
 s = eg.filesavebox()  
 if s != None:  
 book = xlwt.Workbook(s + ".xlsx")  
 sheet = book.add\_worksheet()  
 sheet.write(0, 0,int(self.Control\_sum.text()))  
 for i in range(self.tableWidget.rowCount()):  
 for j in range(self.tableWidget.columnCount()):  
 value=self.tableWidget.item(i, j).text()  
 if (value == self.Control\_sum.text() or value==""):  
 value=""  
 sheet.write(i+1, j+1, value)  
 else:  
 sheet.write(i + 1, j + 1, int(value))  
 book.close()  
  
 msb = QMessageBox()  
 msb.setIcon(QMessageBox.Information)  
 msb.setStyleSheet("QLabel {font-size:20px;}")  
 msb.setStyleSheet("QPushButton {font-size:15px;}")  
 msb.setText("File Save")  
 msb.exec\_()  
  
 def row\_in\_table(self):  
 self.tableWidget.setRowCount(self.spinBoxRow.value())  
 for i in range(self.tableWidget.rowCount()):  
 delegate = AlignDelegate(self.tableWidget)  
 self.tableWidget.setItemDelegateForRow(i, delegate)  
 def column\_in\_table(self):  
 self.tableWidget.setColumnCount(self.spinBoxColumn.value())  
 for i in range(self.tableWidget.columnCount()):  
 delegate = AlignDelegate(self.tableWidget)  
 self.tableWidget.setItemDelegateForColumn(i, delegate)  
  
  
rez = []  
path = []  
class File:  
  
 def \_\_init\_\_(self):  
 self.workbook = ""  
 self.sheet = ""  
  
 def get\_workbook(self):  
 return self.workbook  
  
 def get\_sheet(self):  
 return self.sheet  
  
 def set\_workbook(self, wb):  
 self.workbook = wb  
  
 def set\_sheet(self, sh):  
 self.sheet = sh  
  
 def from\_file\_to\_prog(self):  
 book = xlrd.open\_workbook(self.get\_workbook())  
 sheet = book.sheet\_by\_index(self.get\_sheet())  
 mass\_from\_file = [[sheet.cell\_value(r, c) for c in range(1, sheet.ncols, 1)] for r in range(1, sheet.nrows, 1)]  
 fin\_rez = int(sheet.cell\_value(0, 0))  
  
 for i in range(len(mass\_from\_file)):  
 for j in range(len(mass\_from\_file[0])):  
 if mass\_from\_file[i][j] == '':  
 mass\_from\_file[i][j] = fin\_rez  
 mass\_from\_file[i][j] = int(mass\_from\_file[i][j])  
 return (mass\_from\_file, fin\_rez)  
  
 def prog\_to\_file(self, s, ms):  
 f = open(s, 'w')  
 for i in range(len(ms)):  
 f.write(ms[i])  
 f.write("\n\n")  
 f.close()  
class Algoritm:  
  
 def \_\_init\_\_(self):  
 self.mass = []  
 self.final\_rez = 0  
  
 def get\_mass(self):  
 return self.mass  
  
 def get\_final\_rez(self):  
 return self.final\_rez  
  
 def set\_mass(self, ms):  
 self.mass = ms  
  
 def set\_final\_rez(self, fr):  
 self.final\_rez = fr  
  
 def \_\_str\_\_(self):  
 vivod = ''  
 for i in range(len(self.mass)):  
 for j in range(len(self.mass[0])):  
 if self.mass[i][j] == self.get\_final\_rez():  
 vivod += "X" + " "  
 else:  
 vivod += str(self.mass[i][j]) + " "  
 vivod += "\n"  
 return (vivod)  
  
 def step(self, x, y, sum):  
 vivod = ""  
 sum += self.mass[x][y]  
 if sum > self.get\_final\_rez():  
 return False  
 if x == len(self.mass) - 1 and y == len(self.mass[0]) - 1:  
 if (sum != self.get\_final\_rez()):  
 return False  
 path[x][y] = True  
 for i in range(len(self.mass)):  
 for j in range(len(self.mass[0])):  
 if path[i][j] == True:  
 vivod += str(self.mass[i][j]) + " "  
 else:  
 vivod += "-" + " "  
 vivod += "\n"  
 # print(vivod)  
 rez.append(vivod)  
 path[x][y] = True  
  
 # up  
 if y > 0 and not (path[x][y - 1]):  
 self.step(x, y - 1, sum)  
  
 # down  
 if y < len(self.mass[0])-1 and not (path[x][y + 1]):  
 self.step(x, y + 1, sum)  
  
 # left  
 if x > 0 and not (path[x - 1][y]):  
 self.step(x - 1, y, sum)  
  
 # right  
 if x < len(self.mass)-1 and not (path[x + 1][y]):  
 self.step(x + 1, y, sum)  
  
 path[x][y] = False  
 return False  
  
  
app = QtWidgets.QApplication(sys.argv)  
MainWindow = QtWidgets.QMainWindow()  
ui =Ui\_MainWindow()  
ui.setupUi(MainWindow)  
MainWindow.show()  
sys.exit(app.exec\_())

**Приклад файлів з вхідними даними**

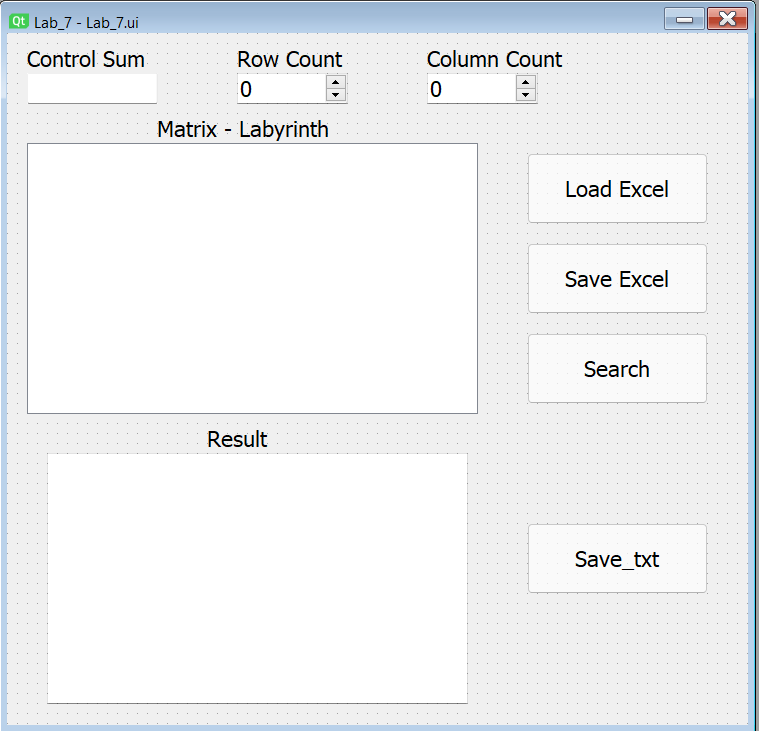


**Встановлення необхідних засобів для реалізації проекту**

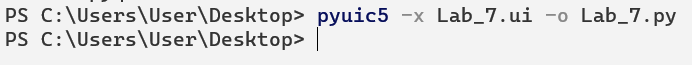




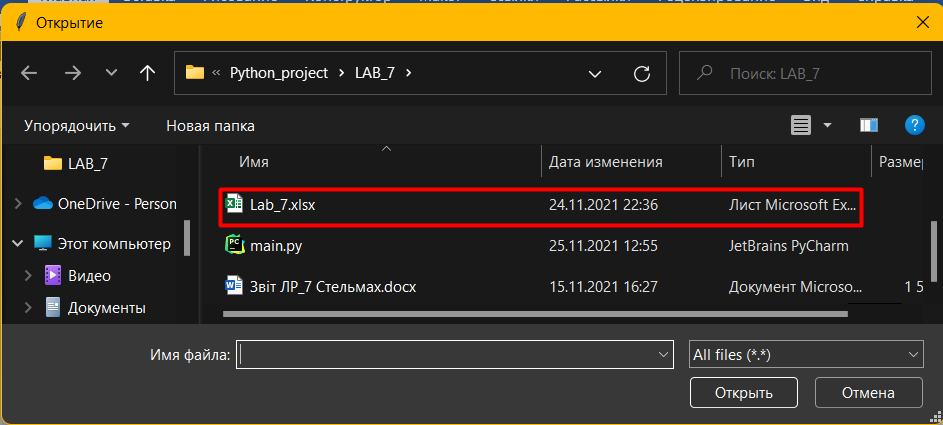
**Створення проекту в QT Designer**

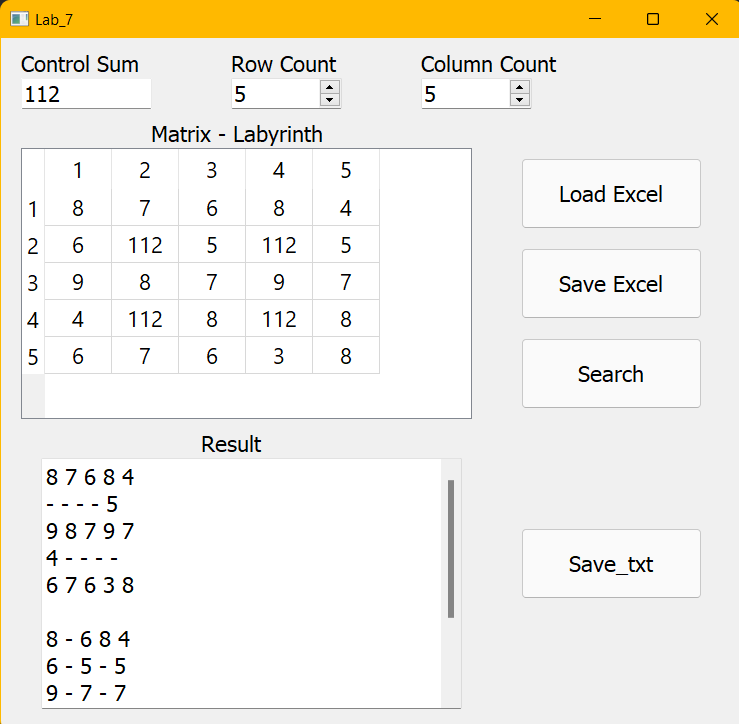


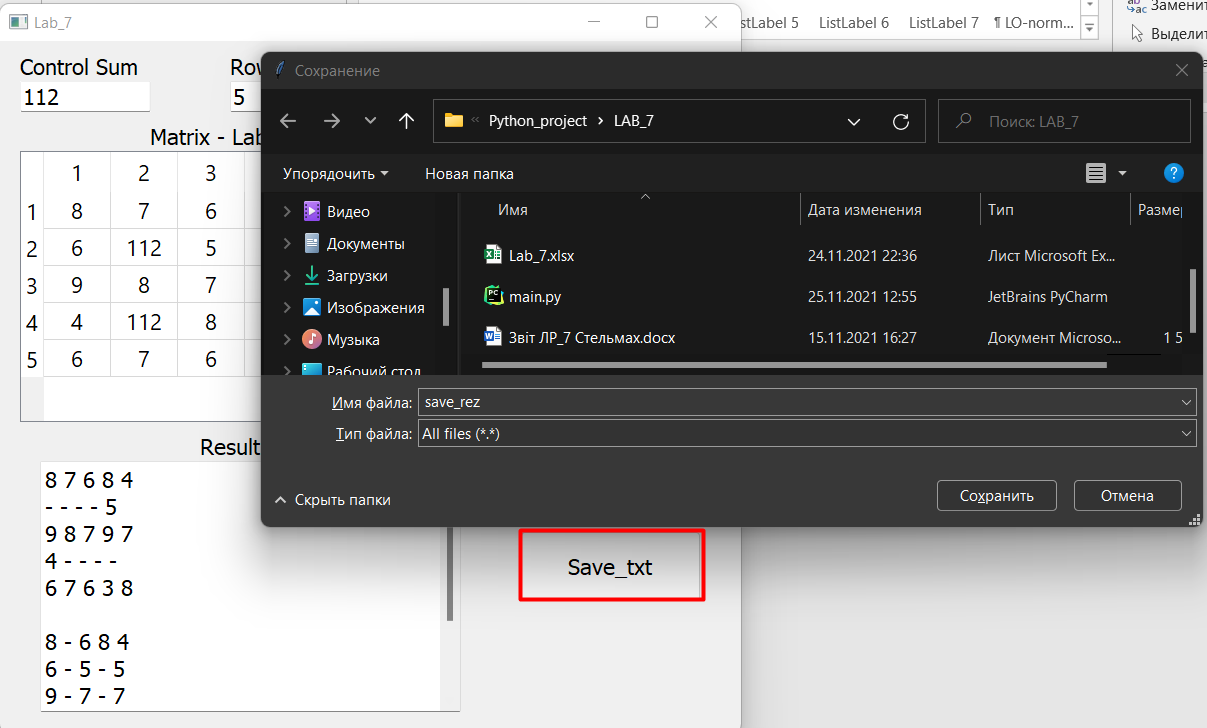
**Перетворення Файлу з дизайном(.ui) в файл (.ру***)*

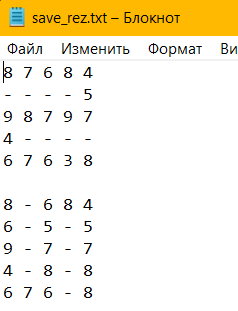


**Результати роботи**









***Висновок:*** На даній лабораторній роботі я ознайомився з можливостями бібліотек мови Python для синтезу та опрацювання зображень, оволодів практичними прийомами побудови графічного інтерфейсу користувача.