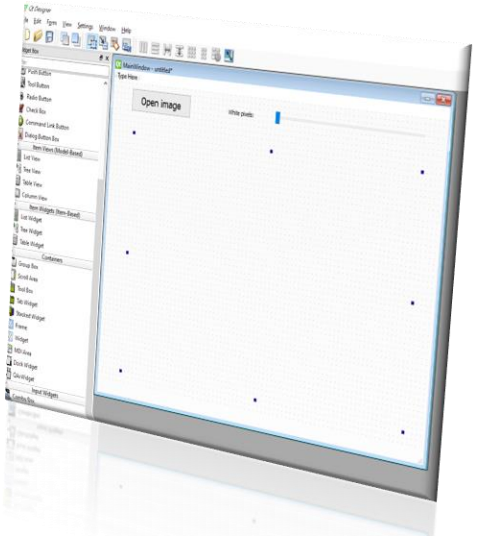
Interfata in Python folosind pyqt5

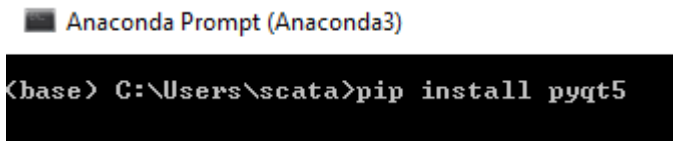
Catalin Stoean

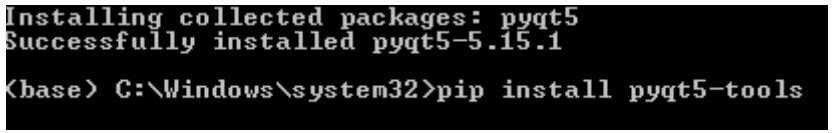
catalin.stoean@inf.ucv.ro

http://inf.ucv.ro/~cstoean



Instalare pyqt5

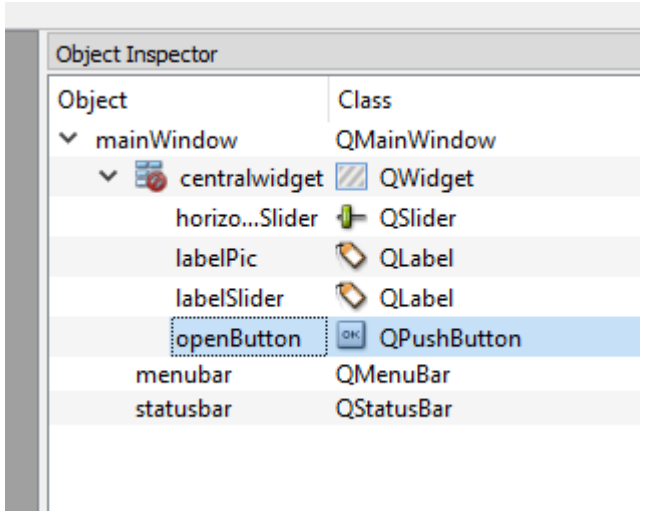


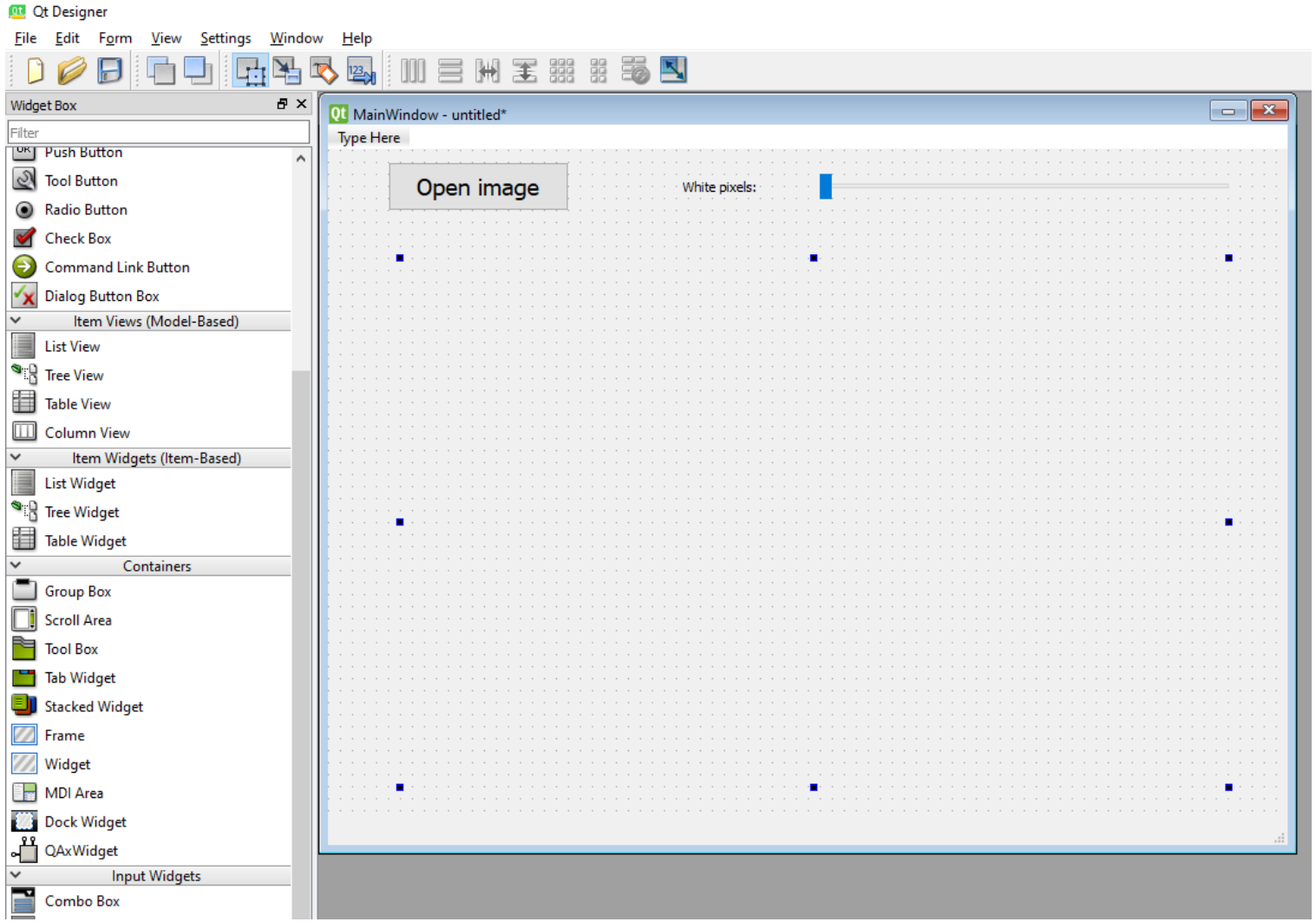


- Daca avem QT Creator deja instalat, putem folosi ferestrele facute cu ajutorul sau ca GUI.
- Dar putem accesa designer.exe din “C:\ProgramData\Anaconda3\Library\bin” sau o cale similara.
  - Puteti cauta designer.exe la butonul de Start din Windows daca nu il gasiti.

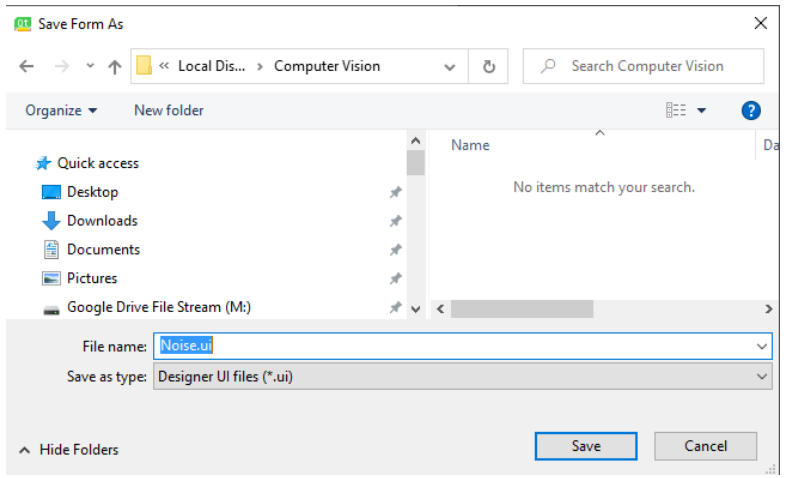
Interfata folosind Designer

- 1 buton pentru deschidere imagine
- 1 slider
- 2 labels

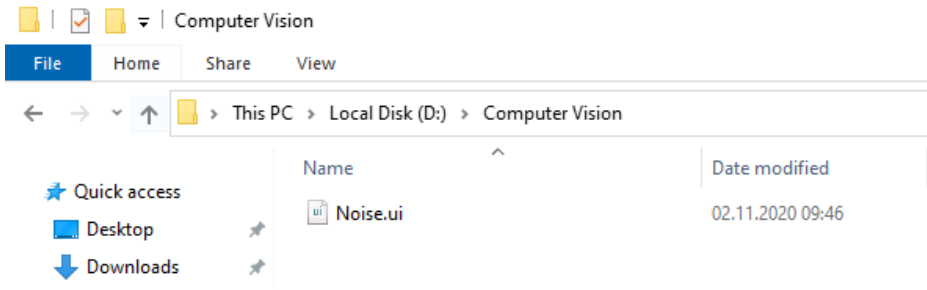


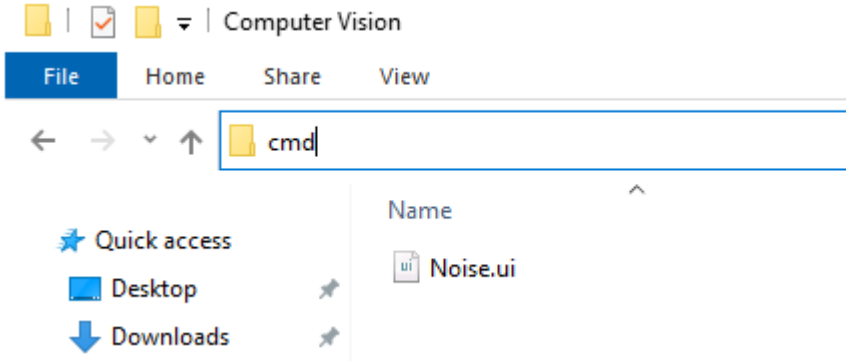


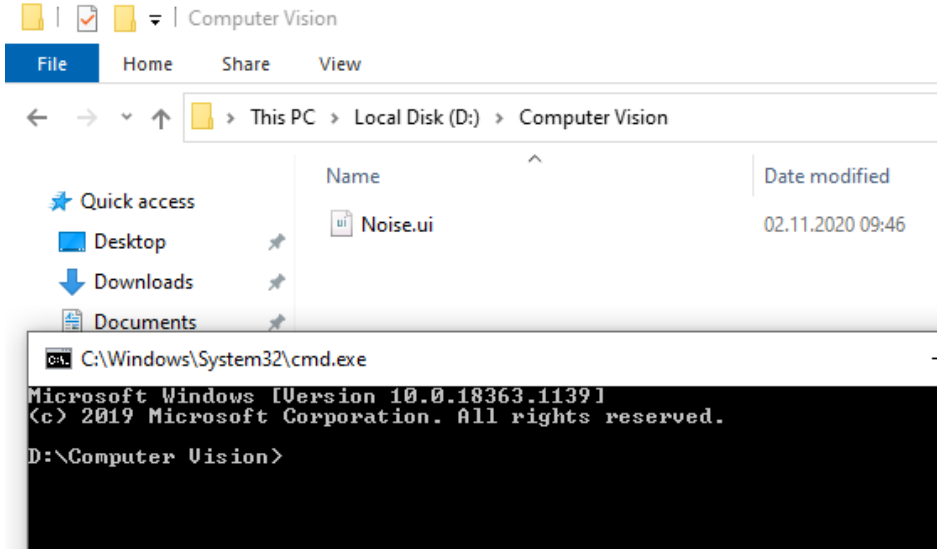
Command prompt



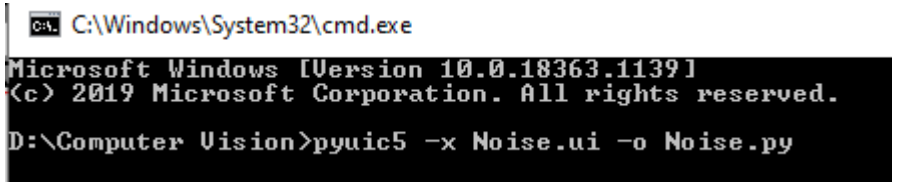
- Deschidem Command Prompt la locatia unde am salvat fisierul .ui.

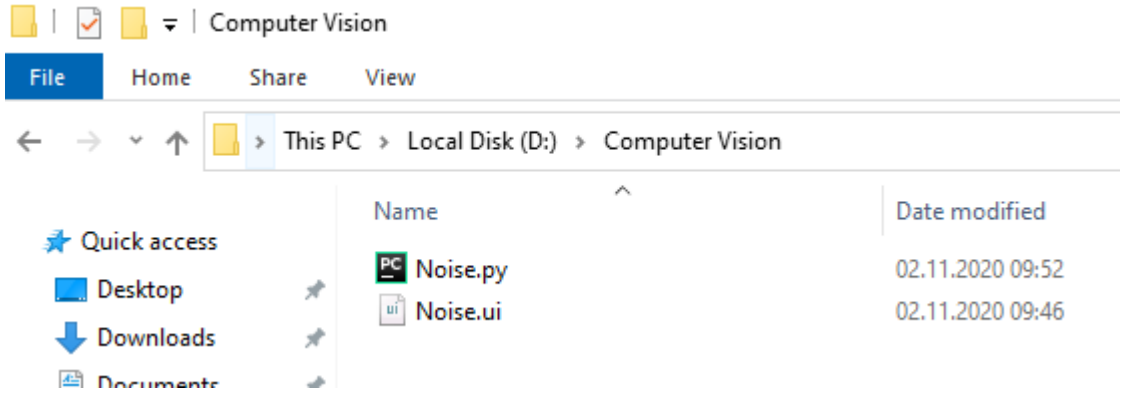






Transformam .ui in .py

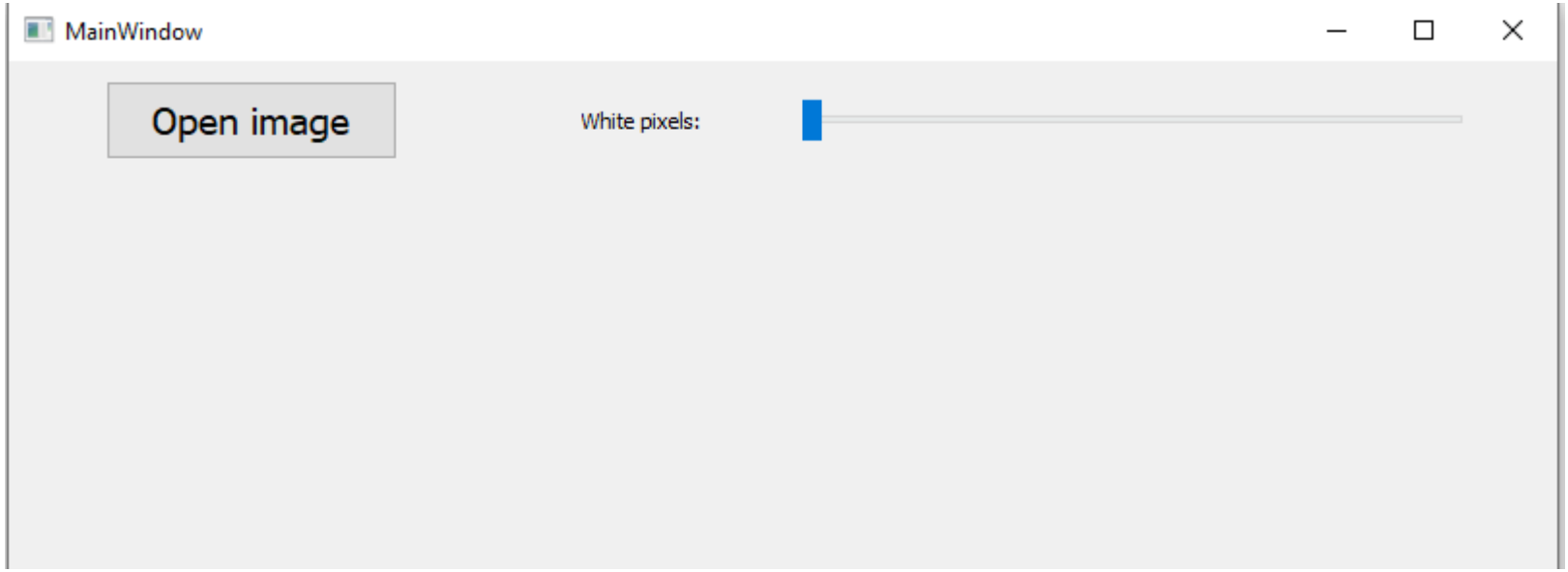





```python
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_mainWindow(object):
    def setupUi(self, mainWindow):
        mainWindow.setObjectName("mainWindow")
        mainWindow.resize(800, 600)
        self.centralwidget = QtWidgets.QWidget(mainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.labelPic = QtWidgets.QLabel(self.centralwidget)
        self.labelPic.setGeometry(QtCore.QRect(60, 90, 691, 441))
        self.labelPic.setText("")
        self.labelPic.setObjectName("labelPic")
        self.horizontalSlider = QtWidgets.QSlider(self.centralwidget)
        self.horizontalSlider.setGeometry(QtCore.QRect(410, 20, 341, 22))
        self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider.setObjectName("horizontalSlider")
        self.openButton = QtWidgets.QPushButton(self.centralwidget)
        self.openButton.setGeometry(QtCore.QRect(50, 10, 151, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.openButton.setFont(font)
        self.openButton.setObjectName("openButton")
        self.labelSlider = QtWidgets.QLabel(self.centralwidget)
        self.labelSlider.setGeometry(QtCore.QRect(296, 20, 91, 21))
        self.labelSlider.setObjectName("labelSlider")
        mainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(mainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 21))
        self.menubar.setObjectName("menubar")
        mainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(mainWindow)
        self.statusbar.setObjectName("statusbar")
        mainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(mainWindow)
        QtCore.QMetaObject.connectSlotsByName(mainWindow)

    def retranslateUi(self, mainWindow):
        _translate = QtCore.QCoreApplication.translate
        mainWindow.setWindowTitle(_translate("mainWindow", "MainWindow"))
        self.openButton.setText(_translate("mainWindow", "Open image"))
        self.labelSlider.setText(_translate("mainWindow", "White pixels: "))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    mainWindow = QtWidgets.QMainWindow()
    ui = Ui_mainWindow()
    ui.setupUi(mainWindow)
    mainWindow.show()
    sys.exit(app.exec_())
```

Programul poate fi rulat

- Dar nu face deocamdata nimic



Daca rulati in Jupyter…

- Pot fi probleme cu fereastra la a doua rulare
- Adaugam if-ul de mai jos:

```
if __name__ == "__main__":
    app=QtWidgets.QApplication.instance() # verifica daca QApplication exista
    if not app: # creaza QApplication daca nu exista
        app = QtWidgets.QApplication(sys.argv)
    #app = QtWidgets.QApplication(sys.argv)

    mainWindow = QtWidgets.QMainWindow()
    ui = Ui_mainWindow()
    ui.setupUi(mainWindow)
    mainWindow.show()

    app.exec_()
    app.quit()
    #sys.exit(app.exec_())
```

- Incheiem cu app.exec_(), urmat de app.quit() pentru a evita avertismente

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QMainWindow
from PyQt5.QtGui import QPixmap
import sys

class Ui_mainWindow(QMainWindow):#inlocuim object cu QMainWindow
    def setupUi(self, mainWindow):
        mainWindow.setObjectName("mainWindow")
        mainWindow.resize(800, 600)
        self.centralwidget = QtWidgets.QWidget(mainWindow)
```

```
    self.labelPic.setObjectName("labelPic")
    self.horizontalSlider = QtWidgets.QSlider(self.centralwidget)
    self.horizontalSlider.setGeometry(QtCore.QRect(410, 20, 341, 22))
    self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
    self.horizontalSlider.setObjectName("horizontalSlider")

    #legatura cu metoda de utilizare valori din slider
    self.horizontalSlider.valueChanged.connect(self.valuechange)

    self.openButton = QtWidgets.QPushButton(self.centralwidget)
    self.openButton.setGeometry(QtCore.QRect(50, 10, 151, 41))
    font = QtGui.QFont()
    font.setPointSize(14)
    self.openButton.setFont(font)
    self.openButton.setObjectName("openButton")

    #legatura cu metoda de deschidere imagine
    self.openButton.clicked.connect(self.getImage)

    self.labelSlider = QtWidgets.QLabel(self.centralwidget)
    self.labelSlider.setGeometry(QtCore.QRect(296, 20, 91, 21))
    self.labelSlider.setObjectName("labelSlider")
```

- Deocamdata citim poza si scriem in eticheta de langa slider ce valoare am selectat.

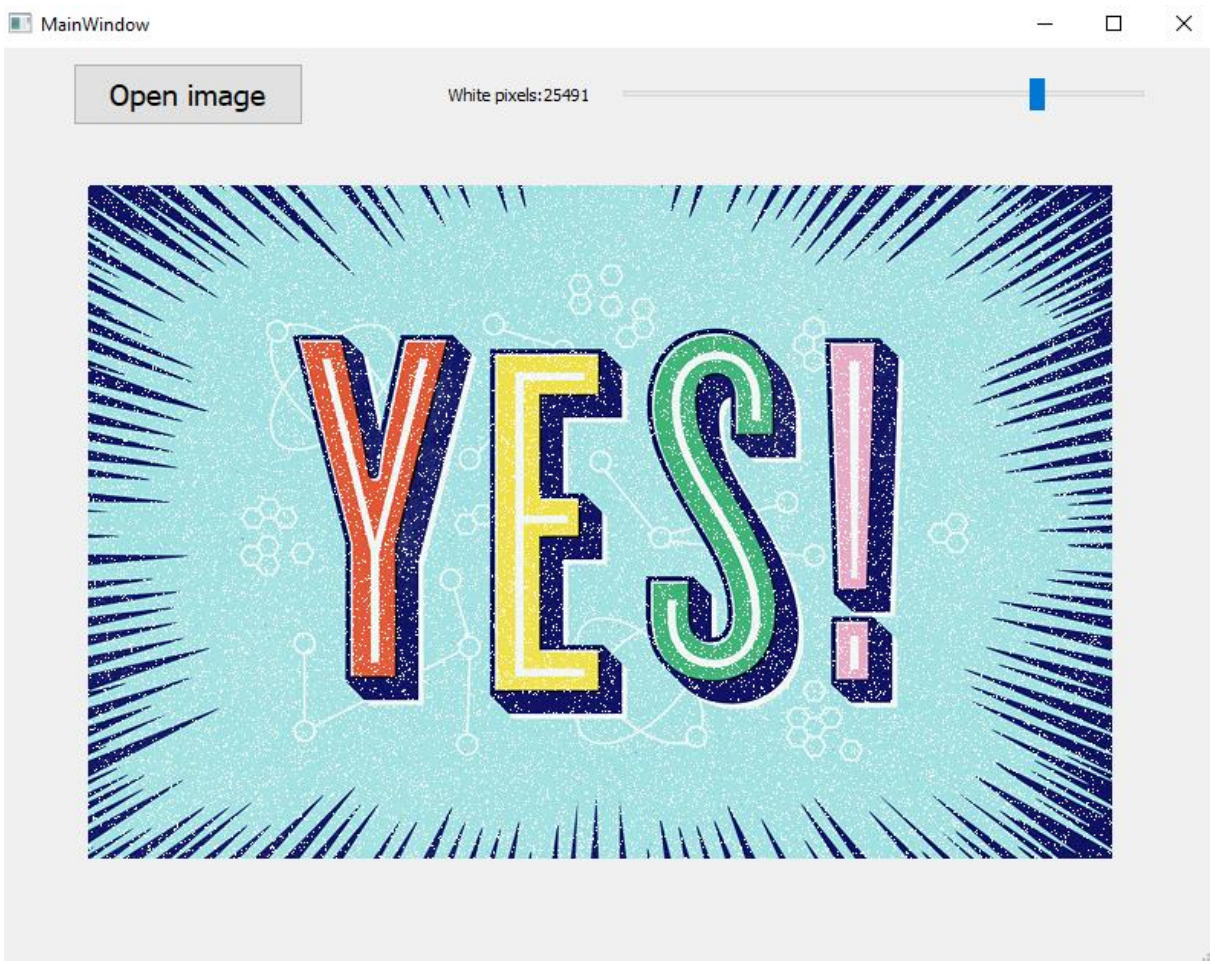
```
def getImage(self):
    filePath = QFileDialog.getOpenFileName(self, 'Open file', 'D:\\',
                                           "Image files (*.jpg *.gif)")

    pixmap = QPixmap(filePath[0])
    self.labelPic.setPixmap(QPixmap(pixmap))
    self.resize(pixmap.width(), pixmap.height())

def valuechange(self):
    pixels = self.horizontalSlider.value()
    self.labelSlider.setText("White pixels:" + str(pixels))
```

Adaugare de zgomot

- Retinem in imageOpenCV imaginea citita initial.
- Cream metoda addWhiteNoise in care adaugam pixeli albi
  - Am presupus ca s-a citit o poza color pentru a nu aglomera codul.

```
def getImage(self):
    filePath = QFileDialog.getOpenFileName(self, 'Open file', 'D:\\',
                                           "Image files (*.jpg *.gif)")
    #avem nevoie de poza pentru a adauga zgomot in ea
    #o declaram globala si o citim cu cv2
    global imageOpenCV
    imageOpenCV = cv2.imread(filePath[0])
    h, w, _ = imageOpenCV.shape
    #setam numarul maxim de pixeli albi la 10% din marimea pozei
    self.horizontalSlider.setMaximum(0.1 * h * w)

    pixmap = QPixmap(filePath[0])
    self.labelPic.setPixmap(QPixmap(pixmap))
    self.resize(pixmap.width(), pixmap.height())

def valuechange(self):
    pixels = self.horizontalSlider.value()
    self.labelSlider.setText("White pixels:" + str(pixels))
    self.addWhiteNoise(pixels)


def addWhiteNoise(self, noOfPixels):
    h, w, _ = imageOpenCV.shape
    imageCopy = imageOpenCV.copy()
    for k in range(noOfPixels):
        i = random.randint(0, h - 1)
        j = random.randint(0, w - 1)
        imageCopy[i, j] = (255, 255, 255)

    imageReady = cv2.cvtColor(imageCopy, cv2.COLOR_BGR2RGB)

    qImg = QImage(imageReady.data, w, h, 3 * w, QImage.Format_RGB888)
    self.labelPic.setPixmap(QPixmap(qImg))
```

Adaugare de zgomot cu pyqt5



GUI cu pyqt5

November 2, 2020

Catalin Stoean

catalin.stoean@inf.ucv.ro

http://inf.ucv.ro/~cstoean

[26]:
```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QMainWindow
from PyQt5.QtGui import QPixmap, QImage
import sys

import cv2
import random

class Ui_mainWindow(QMainWindow):#inlocuim object cu QMainWindow


    def setupUi(self, mainWindow):
        mainWindow.setObjectName("mainWindow")
        mainWindow.resize(800, 600)
        self.centralwidget = QtWidgets.QWidget(mainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.labelPic = QtWidgets.QLabel(self.centralwidget)
        self.labelPic.setGeometry(QtCore.QRect(60, 90, 691, 441))
        self.labelPic.setText("")
        self.labelPic.setObjectName("labelPic")
        self.horizontalSlider = QtWidgets.QSlider(self.centralwidget)
        self.horizontalSlider.setGeometry(QtCore.QRect(410, 20, 341, 22))
        self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider.setObjectName("horizontalSlider")


        #legatura cu metoda de utilizare valori din slider
        self.horizontalSlider.valueChanged.connect(self.valuechange)

        self.openButton = QtWidgets.QPushButton(self.centralwidget)
        self.openButton.setGeometry(QtCore.QRect(50, 10, 151, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
```

```
        self.openButton.setFont(font)
        self.openButton.setObjectName("openButton")

        #legatura cu metoda de deschidere imagine
        self.openButton.clicked.connect(self.getImage)

        self.labelSlider = QtWidgets.QLabel(self.centralwidget)
        self.labelSlider.setGeometry(QtCore.QRect(296, 20, 91, 21))
        self.labelSlider.setObjectName("labelSlider")
        mainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(mainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 21))
        self.menubar.setObjectName("menubar")
        mainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(mainWindow)
        self.statusbar.setObjectName("statusbar")
        mainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(mainWindow)
        QtCore.QMetaObject.connectSlotsByName(mainWindow)

    def getImage(self):
        filePath = QFileDialog.getOpenFileName(self, 'Open file', 'D:\\',
                                               "Image files (*.jpg *.gif)")
        #avem nevoie de poza pentru a adauga zgomot in ea
        #o declaram globala si o citim cu cv2
        global imageOpenCV
        imageOpenCV = cv2.imread(filePath[0])
        h, w, _ = imageOpenCV.shape
        #setam numarul maxim de pixeli albi la 10% din marimea pozei
        self.horizontalSlider.setMaximum(0.1 * h * w)

        pixmap = QPixmap(filePath[0])
        self.labelPic.setPixmap(QPixmap(pixmap))
        self.resize(pixmap.width(), pixmap.height())

    def valuechange(self):
        pixels = self.horizontalSlider.value()
        self.labelSlider.setText("White pixels:" + str(pixels))
        self.addWhiteNoise(pixels)


    def addWhiteNoise(self, noOfPixels):
        h, w, _ = imageOpenCV.shape
        imageCopy = imageOpenCV.copy()
        for k in range(noOfPixels):
            i = random.randint(0, h - 1)
```

```
            j = random.randint(0, w - 1)
            imageCopy[i, j] = (255, 255, 255)

        imageReady = cv2.cvtColor(imageCopy, cv2.COLOR_BGR2RGB)

        qImg = QImage(imageReady.data, w, h, 3 * w, QImage.Format_RGB888)
        self.labelPic.setPixmap(QPixmap(qImg))

    def retranslateUi(self, mainWindow):
        _translate = QtCore.QCoreApplication.translate
        mainWindow.setWindowTitle(_translate("mainWindow", "MainWindow"))
        self.openButton.setText(_translate("mainWindow", "Open image"))
        self.labelSlider.setText(_translate("mainWindow", "White pixels: "))


if __name__ == "__main__":
    app=QtWidgets.QApplication.instance() # verifica daca QApplication exista
    if not app: # creaza QApplication daca nu exista
        app = QtWidgets.QApplication(sys.argv)
    #app = QtWidgets.QApplication(sys.argv)

    mainWindow = QtWidgets.QMainWindow()
    ui = Ui_mainWindow()
    ui.setupUi(mainWindow)
    mainWindow.show()

    app.exec_()
    app.quit()
    #sys.exit(app.exec_())
```

[ ]:

Tkinter

November 2, 2020

Catalin Stoean

catalin.stoean@inf.ucv.ro

http://inf.ucv.ro/~cstoean

1 Citim o imagine si ii adaugam zgomot in GUI folosind Tkinter

[3]:
```
from tkinter import *
from PIL import Image
from PIL import ImageTk
import tkinter.filedialog
import cv2
import random

# Stabilim dimensiunile etichetelor cu poze in functie de
# lungimea dorita pentru eticheta si de dimensiunile pozei
def stabilesteInaltimea(lungimePanou, lungimePoza, inaltimePoza):
    inaltimePanou = (lungimePanou * inaltimePoza) / lungimePoza
    return int(inaltimePanou)

# Metoda ce urmeaza sa fie apelata de catre butonul de incarcare a pozei
def selectareImagine():
    '''
    Punem global pentru cele doua variabile de mai jos pentru a putea
    modifica valorile lor in cadrul acestei metode.
    '''
    global labelA, labelB
    # citim poza de la o locatie selectata de utilizator
    cale = tkinter.filedialog.askopenfilename()

    if len(cale) > 0: #daca avem o cale
        poza = cv2.imread(cale)
        '''
        Citim h si w de la poza.
        Stabilim lungimea etichetei.
        In functie de acestea calculam inaltimea etichetei.
        Redimensionam apoi poza in functie de w si h ale etichetei.
```

```
        '''
        h, w, _ = poza.shape
        lungimePanou = 800
        inaltimePanou = stabilesteInaltimea(lungimePanou, w, h)
        poza = cv2.resize(poza, (lungimePanou, inaltimePanou))
        poza = cv2.cvtColor(poza, cv2.COLOR_BGR2RGB)


        imageCopy = poza.copy()
        for k in range(10000):
            i = random.randint(0, inaltimePanou - 1)
            j = random.randint(0, lungimePanou - 1)
            imageCopy[i, j] = (255, 255, 255)

        # Trecem poza in PIL
        poza = Image.fromarray(poza)
        zgomot = Image.fromarray(imageCopy)

        # Transformam poza in format tkinter
        poza = ImageTk.PhotoImage(poza)
        zgomot = ImageTk.PhotoImage(zgomot)

        # Daca nu este nimic incarcat in etichete
        # Folosim is pentru compararea de obiecte
        if labelA is None or labelB is None:
            # Punem imaginea originala in prima eticheta (stanga)

            labelA = Label(image = poza, width = lungimePanou,
                           height = inaltimePanou)
            labelA.image = poza
            #padx si pady adauga pixeli pe orizontala si verticala langa obiect
            labelA.pack(side="left", padx=10, pady=10)

            # Imaginea ponderata va fi pusa in a doua eticheta
            labelB = Label(image=zgomot, width = lungimePanou,
                           height = inaltimePanou)
            labelB.image = zgomot
            labelB.pack(side="right", padx=10, pady=10)
        # Daca avem deja poze in etichete, trebuie sa folosim "configure"
        else:
            # actualizam etichetele
            labelA.configure(image = poza)
            labelB.configure(image = zgomot)
            labelA.image = poza
            labelB.image = zgomot
    return
```

```
root = Tk()
root.title('Adaugare zgomot')
labelA = None
labelB = None
# Butonul care apeleaza metoda "selectareImagine"
btn = Button(root, text="Incarca o imagine", command=selectareImagine)
#adaugam butonul in partea de jos a ferestrei
btn.pack(side="bottom", fill="both", expand="yes", padx="10", pady="10")

# Lansam interfata
root.mainloop()
```

2 Adaugam si un slider

[5]:
```
from tkinter import *
from PIL import Image
from PIL import ImageTk
import tkinter.filedialog
import cv2

# Stabilim dimensiunile etichetelor cu poze in functie de
# lungimea dorita pentru eticheta si de dimensiunile pozei
def stabilesteInaltimea(lungimePanou, lungimePoza, inaltimePoza):
    inaltimePanou = (lungimePanou * inaltimePoza) / lungimePoza
    return int(inaltimePanou)

# Metoda ce urmeaza sa fie apelata de catre butonul de incarcare a pozei
def selectareImagine():
    '''
    Punem global pentru variabile de mai jos pentru a putea
    modifica valorile lor in cadrul acestei metode.
    '''
    global labelA, labelB, zgomot, pozaInitiala, s
    # citim poza de la o locatie selectata de utilizator
    cale = tkinter.filedialog.askopenfilename()

    if len(cale) > 0: #daca avem o cale
        poza = cv2.imread(cale)
        '''
        Citim h si w de la poza.
        Stabilim lungimea etichetei.
        In functie de acestea calculam inaltimea etichetei.
        Redimensionam apoi poza in functie de w si h ale etichetei.
        '''
        h, w, _ = poza.shape
```

```
        lungimePanou = 800
        inaltimePanou = stabilesteInaltimea(lungimePanou, w, h)

        poza = cv2.resize(poza, (lungimePanou, inaltimePanou))

        poza = cv2.cvtColor(poza, cv2.COLOR_BGR2RGB)
        pozaInitiala = poza.copy()

        imageCopy = pozaInitiala.copy()
        for k in range(s.get()):
            i = random.randint(0, inaltimePanou - 1)
            j = random.randint(0, lungimePanou - 1)
            imageCopy[i, j] = (255, 255, 255)

        # Trecem poza in PIL
        poza = Image.fromarray(poza)
        zgomot = Image.fromarray(imageCopy)

        # Transformam poza in format tkinter
        poza = ImageTk.PhotoImage(poza)
        zgomot = ImageTk.PhotoImage(zgomot)

        # Daca nu este nimic incarcat in etichete
        if labelA is None or labelB is None:
            # Punem imaginea originala in prima eticheta (stanga)

            labelA = Label(image = poza, width = lungimePanou,
                           height = inaltimePanou)
            labelA.image = poza
            labelA.pack(side="left", padx=10, pady=10)

            # Imaginea ponderata va fi pusa in a doua eticheta
            labelB = Label(image=zgomot, width = lungimePanou,
                           height = inaltimePanou)
            labelB.image = zgomot
            labelB.pack(side="right", padx=10, pady=10)
        # Daca avem deja poze in etichete, trebuie sa folosim "configure"
        else:
            labelA.configure(image = poza)
            labelB.configure(image = zgomot)
            labelA.image = poza
            labelB.image = zgomot
    return

# Metoda ce urmeaza sa fie apelata de catre slider
def schimbaPrag(val):
```

```
    global labelB, zgomot
    if labelB == None: #Nu am citit inca poza initiala daca nu avem nimic in␣
 ↪labelB
        pass
    else:#Recalculam imaginea zgomot si o punem in eticheta B
        imageCopy = pozaInitiala.copy()
        h, w, _ = pozaInitiala.shape
        for k in range(s.get()):
            i = random.randint(0, h - 1)
            j = random.randint(0, w - 1)
            imageCopy[i, j] = (255, 255, 255)

        zgomot = Image.fromarray(imageCopy)

        zgomot = ImageTk.PhotoImage(zgomot)
        labelB.configure(image = zgomot)
        labelB.image = zgomot
    return

root = Tk()
root.title('Adaugare zgomot')
labelA = None
labelB = None
pozaInitiala = None #imaginea initiala este globala ca sa o folosim in␣
 ↪schimbaPrag
# Butonul care apeleaza metoda "selectareImagine"
btn = Button(root, text="Incarca o imagine", command = selectareImagine)
#adaugam butonul in partea de jos a ferestrei
btn.pack(side="bottom", fill="both", expand="no", padx="10", pady="10")
#adaugam un slider si pentru acesta o metoda
s = Scale(root, from_=0, to=10000, orient=HORIZONTAL, command = schimbaPrag)
s.set(100)
s.pack()
# Lansam interfata
root.mainloop()
```

Un tutorial cu informatii succinte si exemple relativ la interfete in Python cu Tkinter se gaseste aici: https://www.python-course.eu/python_tkinter.php

[ ]: