گزارش سوم

مقدمه

GUI یا رابط کاربری گرافیکی همان چیزی است که کاربر می‌‎بیند و در زمان نیاز به برنامه‌ی شما با آن تعامل می‌کند. داشتن یک GUI خوب می‌تواند تاثیر به‌سزایی در تجربه‌ی کاربری داشته باشد. به‌همین منظور در این گزارش چندین کتابخانه و ابزار معروف که برای توسعه‌ی برنامه‌های GUI در زبان Python استفاده می‌شوند معرفی خواهد شد.

1-pyqt5

PyQt5 یکی از محبوب‌ترین ابزارهای توسعه‌ی برنامه‌های گرافیکی است که توسط Riverbank Computing توسعه داده شده و می‌توان گفت این ابزار ادغام شده‌ی زبان Python با فریم‌ورک Qt است که از آن برای توسعه‌ی برنامه‌های کراس پلتفرم تحت دستکاپ استفاده می‌شود. البته PyQt فراتر از یک ابزار برای توسعه‌ی برنامه‌های گرافیکی است و امکانات بسیار زیادی را در اختیار شما قرار می‌دهد.

در ادامه برنامه ای با این کتابخانه به طور کامل نوشته خواهد شد و جزئیات بخش های آن شرح داده می­شود.

2-thinker

Tkinter یکی دیگر از محبوب‌ترین کتابخانه‌های توسعه‌ی رابط کاربری گرافیکی در زبان Python است که جزو اولین گزینه‌های مبتدیان برای یادگیری توسعه‌ی GUI محسوب می‌شود زیرا سینتکس آن برای یادگیری بسیار ساده است. همچنین این کتابخانه ویجت‌های متنوعی مانند labels، buttons، text fields، checkboxes و scroll buttons را ارائه می‌دهد و با پشتیبانی از طراحی grid به آسان‌تر شدن فرایند در توسعه‌ی یک طراحی پیچیده کمک می‌کند.

3-kivy

Kivy یک فریم‌ورک متن باز Python است که برای توسعه‌ی سریع برنامه‌های گرافیکی مورد استفاده قرار می‌گیرد و این فریم‌ورک قابلیت multi-touch را به‌همراه ویجت‌های مختلف برای برنامه‌ی شما فراهم می‌کند. همچنین شما می‌توانید Kivy را روی پلتفرم‌های مختلفی مانند Windows، Linux، OS X، Android، iOS و Raspberry Pi اجرا کنید.

4-wxPython

wxPython یک ابزار کراس پلتفرم برای توسعه‌ی برنامه‌های GUI در زبان Python است که خودش با زبان C++ توسعه داده شده و به‌کمک آن می‌توانید رابط‌های کاربری گرافیکی بهینه‌ای را با کم‌ترین زحمت ایجاد کنید. در حال حاضر این ابزار از پلتفرم‌های Windows، MacOS، Mac OS X و Linux پشتیبانی می‌کند.

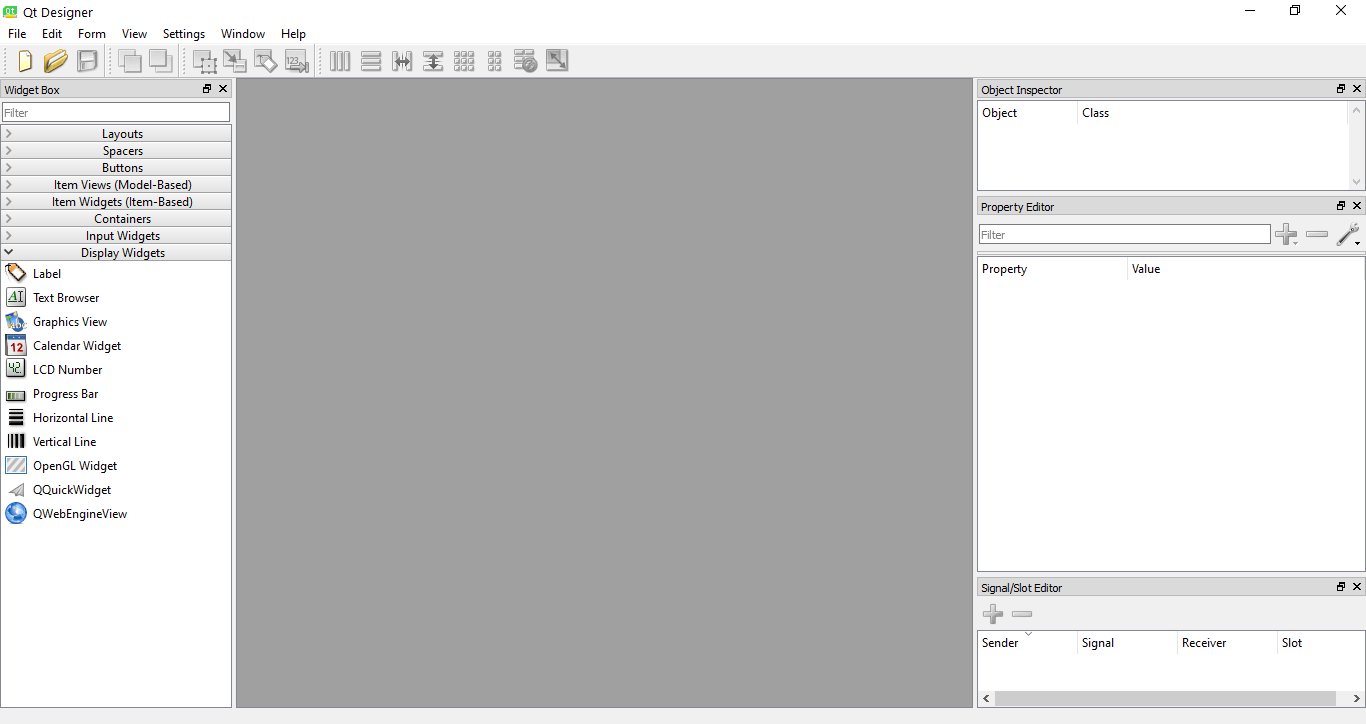
5-PysimpleGUI

PySimpleGUI یک فریم‌ورک GUI در زبان Python است که از Python 3 پشتیبانی می‌کند و به شما امکان می‌دهد که رابط‌‎های کاربری گرافیکی سفارشی‌سازی شده‌ و جذابی را توسعه دهید. همچنین PySimpleGUI با توسعه‌ی برخی کدهای آماده در هسته‌ی این فریم‌ورک باعث شده تا توسعه‌دهندگان مسیر هموارتری داشته باشند.

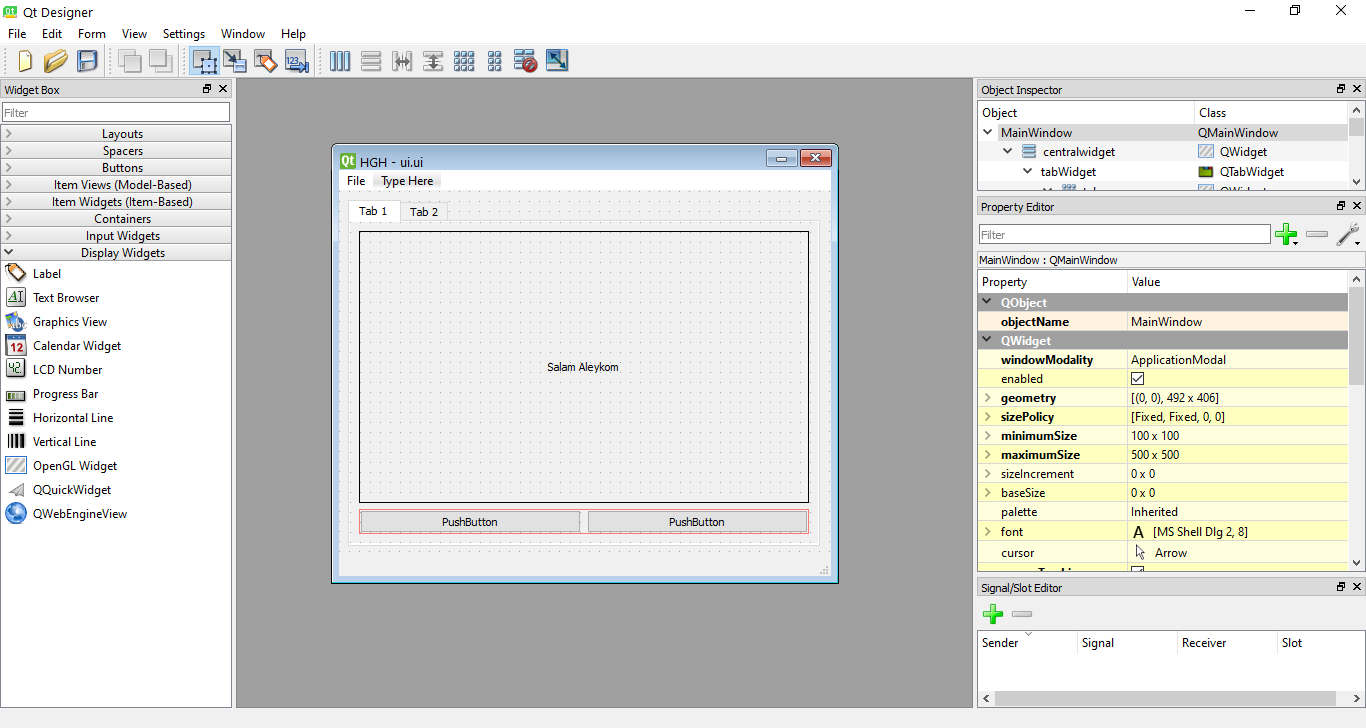
نحوه طراحی GUI توسط pyqt5

برنامه های pyqt را به صورت کلی میتوان به دو شیوه طراحی نمود، شیوه اول به صورت کد نویسی ui و شیوه دوم با کمک نرم افزار designer . با کمک این نرم افزار میتوان به صورت drag and drop، ui مورد نظر را طراحی نمود و سپس با کمک یک دستور ui طراحی شده را به صورت کد در آورد.

محیط کاری نرم افزار فوق به صورت زیر است:



و به عنوان نمونه پنجره زیر توسط این نرم افزار ساخته شده است:



پس ساخت ui توسط این نرم افزار با کمک دستور زیر میتوان فایل طراحی شده را به صورت کد در آورد.

python -m PyQt5.uic.pyuic -x [FILENAME].ui -o [FILENAME].py

به عنوان مثال کد ساخته شده برای شکل فوق به صورت زیر در می­آید

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.setWindowModality(QtCore.Qt.ApplicationModal)

        MainWindow.resize(492, 406)

        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Fixed)

        sizePolicy.setHorizontalStretch(0)

        sizePolicy.setVerticalStretch(0)

        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())

        MainWindow.setSizePolicy(sizePolicy)

        MainWindow.setMinimumSize(QtCore.QSize(100, 100))

        MainWindow.setMaximumSize(QtCore.QSize(500, 500))

        MainWindow.setMouseTracking(True)

        MainWindow.setTabletTracking(False)

        MainWindow.setWindowOpacity(1.0)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.centralwidget)

        self.verticalLayout\_3.setObjectName("verticalLayout\_3")

        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)

        self.tabWidget.setObjectName("tabWidget")

        self.tab = QtWidgets.QWidget()

        self.tab.setObjectName("tab")

        self.gridLayout = QtWidgets.QGridLayout(self.tab)

        self.gridLayout.setObjectName("gridLayout")

        self.label = QtWidgets.QLabel(self.tab)

        self.label.setEnabled(True)

        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

        sizePolicy.setHorizontalStretch(0)

        sizePolicy.setVerticalStretch(0)

        sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())

        self.label.setSizePolicy(sizePolicy)

        self.label.setFrameShape(QtWidgets.QFrame.Panel)

        self.label.setTextFormat(QtCore.Qt.RichText)

        self.label.setAlignment(QtCore.Qt.AlignCenter)

        self.label.setObjectName("label")

        self.gridLayout.addWidget(self.label, 0, 0, 1, 1)

        self.horizontalLayout = QtWidgets.QHBoxLayout()

        self.horizontalLayout.setObjectName("horizontalLayout")

        self.pushButton\_2 = QtWidgets.QPushButton(self.tab)

        self.pushButton\_2.setObjectName("pushButton\_2")

        self.horizontalLayout.addWidget(self.pushButton\_2)

        self.pushButton = QtWidgets.QPushButton(self.tab)

        self.pushButton.setCheckable(False)

    def retranslateUi(self, MainWindow):

        \_translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(\_translate("MainWindow", "HGH"))

        MainWindow.setStatusTip(\_translate("MainWindow", "salam"))

        MainWindow.setWhatsThis(\_translate("MainWindow", "this is ..."))

        self.label.setText(\_translate("MainWindow", "Salam Aleykom"))

        self.pushButton\_2.setText(\_translate("MainWindow", "PushButton"))

        self.pushButton.setText(\_translate("MainWindow", "PushButton"))

        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), \_translate("MainWindow", "Tab 1"))

        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_2), \_translate("MainWindow", "Tab 2"))

        self.menuNew.setTitle(\_translate("MainWindow", "File"))

        self.statusbar.setStatusTip(\_translate("MainWindow", "an"))

        self.actionNew.setText(\_translate("MainWindow", "New"))

        self.actionNew.setStatusTip(\_translate("MainWindow", "New Pesar"))

        self.actionExit.setText(\_translate("MainWindow", "Exit"))

حال در یک برنامه پایتون دیگر میتوان یک کلاس ساخت و یک برنامه از جنس Qwidget تعریف نمود، و پنجره اصلی آن را با محتوای کلاس ساخته شده فوق پر کرد و برنامه پایتون خود را مستقل از ui نوشت تا در صورت تغییر ui برنامه نوشته شده دچار تداخل و اشکال نشود.

همچنین هنگام ساخت این کلاس جدید جزئیاتی از ui اصلی که در نرم افزار انجام شده را تغییر داد. به مثال زیر توجه فرمایید:

import sys

from PyQt5 import QtWidgets

from ui import  Ui\_MainWindow

class RunDesignerGUI():

    def \_\_init\_\_(self):

        app = QtWidgets.QApplication(sys.argv)

        self.MainWindow = QtWidgets.QMainWindow()

        self.ui = Ui\_MainWindow()

        self.ui.setupUi(self.MainWindow)

        self.updateWidget()

        self.widgetsAction()

        self.MainWindow.show()

        sys.exit(app.exec\_())

    def updateWidget(self):

        self.MainWindow.setWindowTitle('SALAM1')

        self.ui.label.setText('SALAM2')

    def close\_GUI(self):

        self.MainWindow.close()

if \_\_name\_\_ == "\_\_main\_\_":

    RunDesignerGUI()

در این برنامه کلاس Ui\_MainWindow که پیشتر توسط qt designer ساخته شد در self.ui قرار گرفت سپس MainWindow که از کلاس QtWidgets ساخته شده است را به آن لینک کرده است به همین منظور برنامه ای که در qt designer ساخته شد، با ساختن یک شی از این کلاس اجرا خواهد شد.

در ادامه با کمک تابع updateWidget جزئیاتی که در qt designer ساخته شد، تغییر می­یابد.