**INF 1511 – Revision notes – Chapter 7**

1. **Qt tool kit**

The Qt tool kit is a framework for developing **user interfaces (UI)** using a **graphical user interface (GUI)**. This means you can develop forms by dragging and dropping objects such as buttons, text boxes etc.

1. **PyQt**

PyQt is a version of Qt that works with Python. PyQt is installed by configuring it to work with the version of Python installed on your workstation. Thus it is important to ensure that the version of Python installed is compatible with the version of PyQt you want to install. Additionally the configuration should take into account the folders in which both Python and PyQt are installed. Where possible, an installation package .exe file should be used to install both PyQt. Usually in the installation package, all the necessary configurations have been included. For this course, a separate installation instruction manual is provided.

1. **Creating a GUI Application with Code**

The code required to create a GUI application can be written without using a drag and drop **integrated development environment (IDE**) such as Qt (see example on page 203 of the textbook).

Note that the interpreter for GUI programs is **Pythonw.exe**, and because of that, GUI programs are saved with extension .**pyw**, compared to **.py** for the console (command line) programs which are interpreted by **Python.exe**.

1. **Using Qt Designer**

Qt designer is the GUI developer that comes with PyQt.

The objects in QT are called widgets. Qt uses widgets to define the top level class in the program created, i.e. your program will inherit from this class. The following are the three top level widgets used; **QDialog, Qwidget and QMainWindow.** Each of these are used as options for templates for the creation of a new application.

Qt is made up of the following main windows;

**Object Inspector -** Displays a hierarchical list of all the objects present on the form.

**Property Editor -** to view and change the properties of the form and widgets.

**Resource Browser** – used to manage resources like images, audio, video, etc.

**Widget Box** - displays a categorized list of widgets and other objects that you can use for designing a user interface.

**Signal/Slot Editor -** displays the signal/slot connections between objects.

**Action Editor -** used to manage the actions of your applications.

1. **The Widgets**

Found in the Widget window, the widgets are the objects used to design GUIs. The following are the different categories of widgets in Python and a brief description of the features and/or uses;

**Layouts** are used for arranging and sizing the widgets in a desired manner.

**Spacers** are used for inserting spaces between widgets or groups of widgets.

**Buttons** are used to initiate an action. Most of the coding in the applications are placed in the buttons so that the program can react to users when they click the buttons.

**Item Views widgets** are used for displaying large volumes of data. Item views may be model based or item based.

**Container widgets** are used to control a collection of objects on a form.

**Input Widgets** are for used for interacting with the user. These are used to collect input from the user into the program for processing.

**Display widgets** are used for displaying information or messages to the user.

**Phonon** is a multimedia (application programming interface) API that provides an abstraction layer for capturing, mixing, processing, and playing audio and video.

1. **Toolbar**

The toolbar has icons used to pass commands to the IDE.

1. **Event Handling in PyQt**

An event in this context occurs when something happens in the program in execution (run) state, for example the user may click a button or press enter or enter text or a certain time or date may be reached. The program would be designed to handle (react and do something) when the appropriate event occurs, depending on the purpose of the program.

In PyQt, the event-handling mechanism is also known as **signals and slots.** Every widget emits signals when its state changes.

To perform a task in response to a signal, the signal has to be connected to a slot, which is the method containing the code that you want to be executed on occurrence of a signal. Some objects have predefined slots and these can be connected to signals in QT designer. To do this, the signals and connections mode of Qt should be selected.

Custom slots can be developed in Python by writing a method and calling it when a widget sends a signal, e.g.

QtCore.QObject.connect(self.ui.ClickMeButton, QtCore.SIGNAL('clicked()'),self.dispmessage)

Connects the signal sent by ClickMeButton to the method dispmessage. See the textbook page 231 to 232 for the complete program.

1. **Converting Data Types**

The widgets that collect input from users, e.g. text boxes collect string data. You may want to use this data as numerical data in your program.

The methods below can be used to convert the data to numerical (int()) data for calculations in the program and back to string data (int()) for displaying on the UI.

int(): Converts the passed argument into integer data type.

str(): Converts the passed argument into string data type.

1. **Defining Buddies**

Buddies are used to make your program user friendly, so that users may use shortcut keys such as ctrl + l for example to navigate to a particular widget in the program.

Label widgets can create short cut keys if & is placed before a letter in the text of the label. In the Buddy editing mode, other widgets, e.g. a text box can then be connected to the label widget, allowing the user to use the label’s shortcut to go to the text box.

1. **Setting Tab Order**

The tab order is the order in which the widgets in the program will get focus (the cursor) as the user type and presses the tab button. If a user is entering First Name, Surname and Date of Birth on a form, if the tab order is not set up correctly, after typing the First Name, the cursor may go to Date of Birth before the user can type Surname. The default tab order is the order in which the widgets are added to the form.

To set the tab order in Python, select Tab Order editing mode on the toolbar in Qt and change the numbers next to the widgets so that the sequence is what you want the order to be when the user is running the program.