# **DOCUMENTATION**

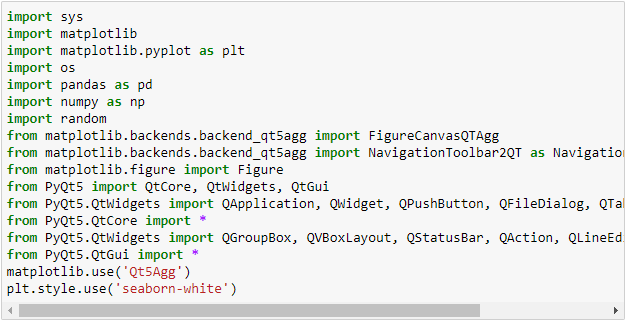
## Sample UI Design using PyQt5 for plotting data

UI is the most essential medium of communication between the user/client with the product. UI gives the first impression and the usability of the program. The simpler the UI, the easier to use. In this notebook we, a basic UI is being built, which will be able to upload a dataset and plot the curve of the data. The plot should contain all the basic functionalities like zoom function, etc.

In Python, in order to build an UI, some packages are required. For executing the codes, Jupyter notebook is used. The packages required for this module are as follows:

* sys, os: to read the system path of the data and provide inbuilt system functionalities like execute.
* pandas: to read and convert the data into a Data Frame.
* numpy: to perform some numerical functionalities.
* matplotlib: to plot and visualize the data.
* PyQt5: to create the UI and manage layouts.

There are numerous sub-packages if the above-mentioned packages, which are required in the program. In order to import the required packages, the following code is used.



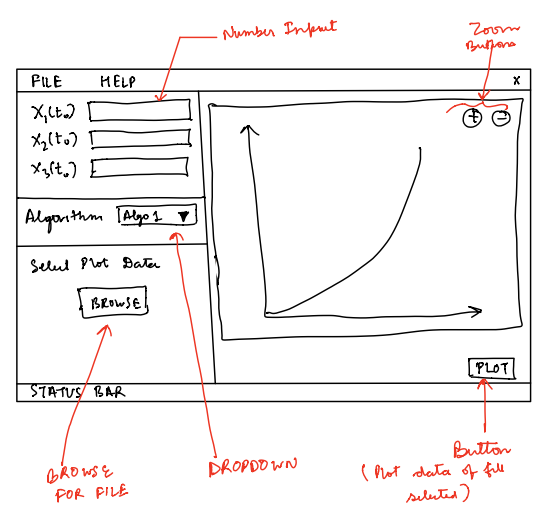
After Importing the required documents, now the Ui Design phase begins. The Design phase consists of 4 parts:

1. Create the widget.
2. Design the widget.
3. Add functionalities.
4. Position in a layout.

These steps are followed throughout the building phase of the UI.

### **Problem statement:**

The aim is to design a UI which can be able to browse and select a dataset from the system and display the plot. The general idea of the design and functionalities can be visualized from the rough diagram given below.



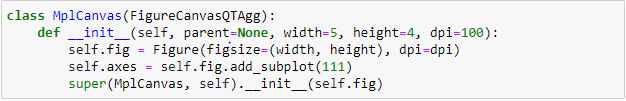
This approach of this design is as follows: -

* A canvas is created using matplotlib library, which is used for plotting the data.
* A plot button is created, which will trigger the plot function when clicked.
* These tow widgets (canvas, button) are bind together into a group box.
* Another group box is created for holding the browsing button to select the dataset from the system path.
* Two other group box holds the algorithm and the input parts.
* Now these four group boxes are put into a grid layout in the main Ui window.
* This Ui window is then placed into the main window, which contains the menu and the status bar.

### **Coding:**

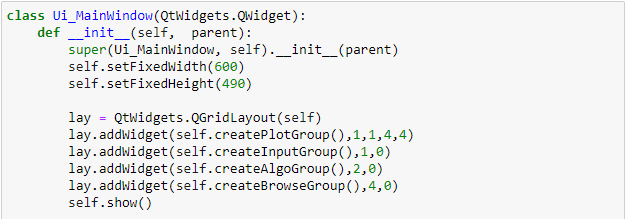
#### **Building the Matplotlib canvas class.**

The class named MplCanvas initializes a matplotlib figure canvas with default dimensions (5,4). It initializes the figure when an object of this class is created.



#### **Building the Ui\_MainWindow class.**

This class is responsible for creating and binding the widgets using the group box and positioning the group boxes. We will be following a top-down approach i.e. creating the base layout grid. We will first initialize the Ui window and create the grid where the widgets will be placed.

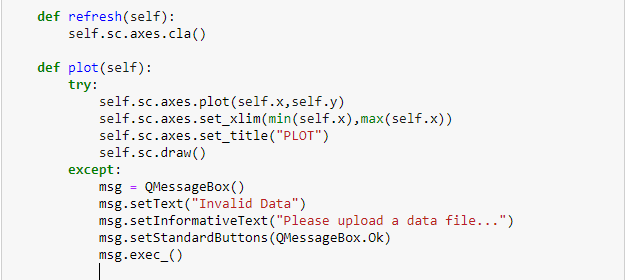


#### Making the createPlotGroup()

The createPlotGroup() function returns the group box containing the widgets. There are mainly 3 widgets in the group box namely the canvas, plot button, refresh button. The canvas is a white space where the graph will be plotted. The plot button, when clicked triggers an event to plot the graph inside the canvas. The refresh button is used to clear the axes in case of a new plot. Each button is redesigned using stylesheet to give an interactive look.

The code of the following is given beside:

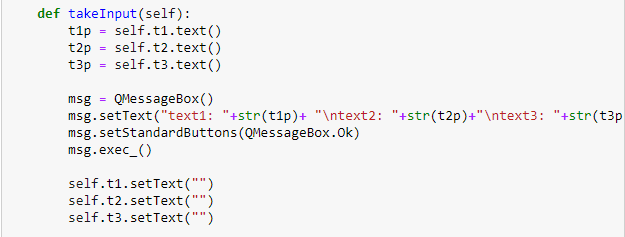
Both the buttons have an onclick event defined in the following functions:



#### Making the createInputGroup()

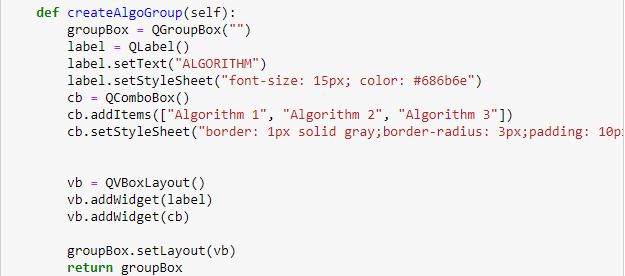
The createInputGroup() function creates the group containing the input boxes and its label. The base layout used here is a grid layout where each label is followed by an input box in that same line. At the end of the group box is a button, on click of which triggers an event to take the input from the input box and display it in a pop-up message box. The code of this segment is given below.

The button on click event function (takeInput()) is given below.



#### Making the createAlgoGroup()

The createAlgoGroup() function returns a group box containing a label and a combo box inflated with names of algorithms. This layout can be implemented in the following way.

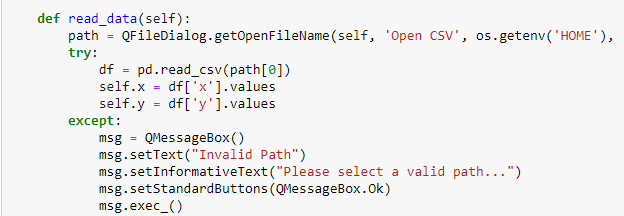


#### Making the createBrowseGroup()

The createBrowseGroup() function returns a group box consisting of a label followed by a button. On click of that button opens a browser window from where a dataset can be uploaded for plotting. This even reads the system path of the dataset and loads the data using the pandas library. The code of the group box is given below.



The On click function of the button calls a function called read\_data(), which reads the data from the path and creates the data frame:



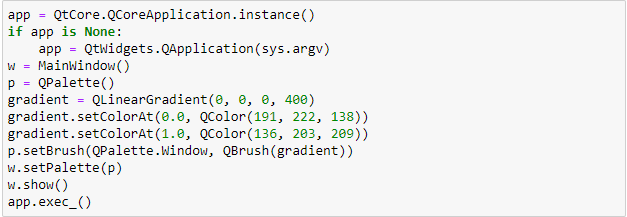
**Building the MainWindow class.**

The MainWindow builds the parent window of the UI\_MainWindow class. This class creates the menu bar and the status bar for the UI and holds the main UI as the central widget of the window. This class also sets the title of the window. From this class’s constructor function (\_\_init\_\_()) the object of the Ui\_MainWindow class is created and then set as a central widget. The code for the MainWindow class is given below.



## **Executing the classes with some background design using QPallete:**

The final step is to create an instance of the MainWindow class and execute the application. While executing the application, some background colouring is also added for better look.



## **Output:**

