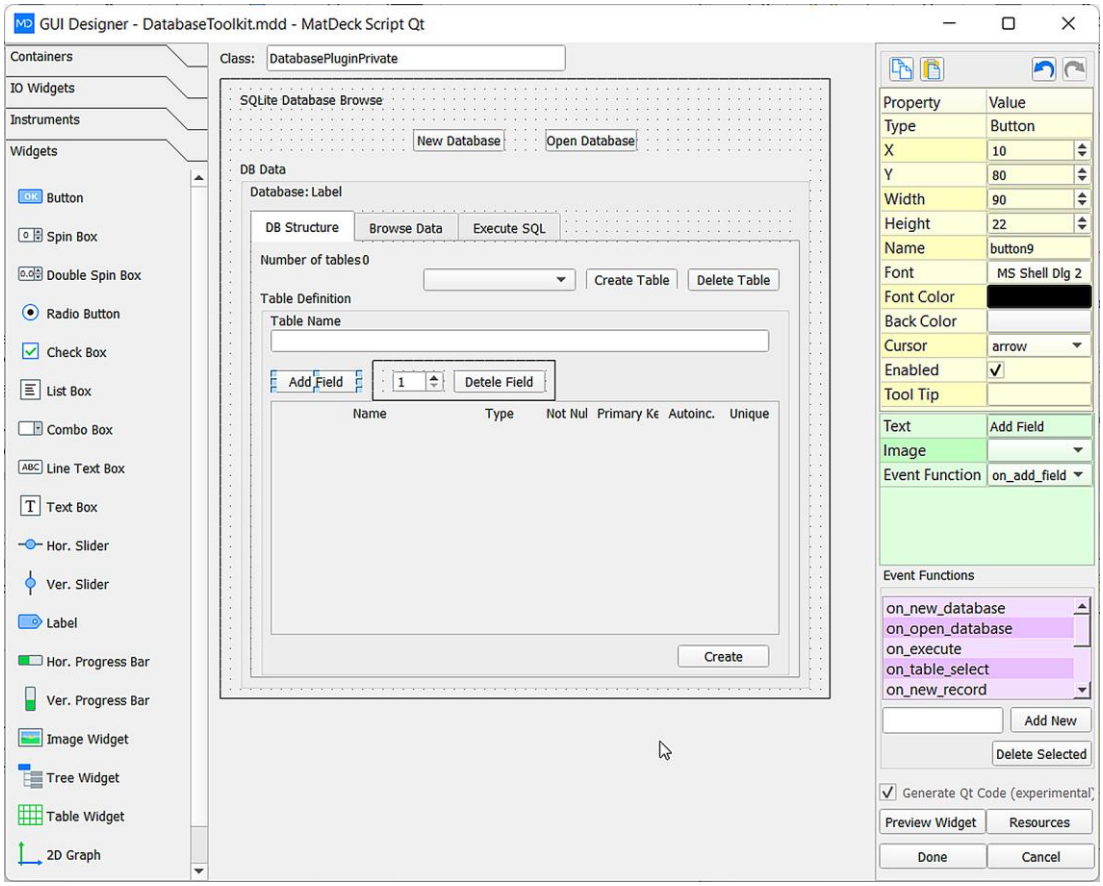
MatDeck

Flet GUI Designer

Drag-and-drop GUI building for MatDeck · User manual



Contents

Introduction

Available widgets

Opening a MD GUI Designer

Containers

Event functions

Widget settings

Using tables and 2D graphs

Finishing up

Using GUI Designer code in a document

Interacting with generated GUI code

Executing and outputting GUIs

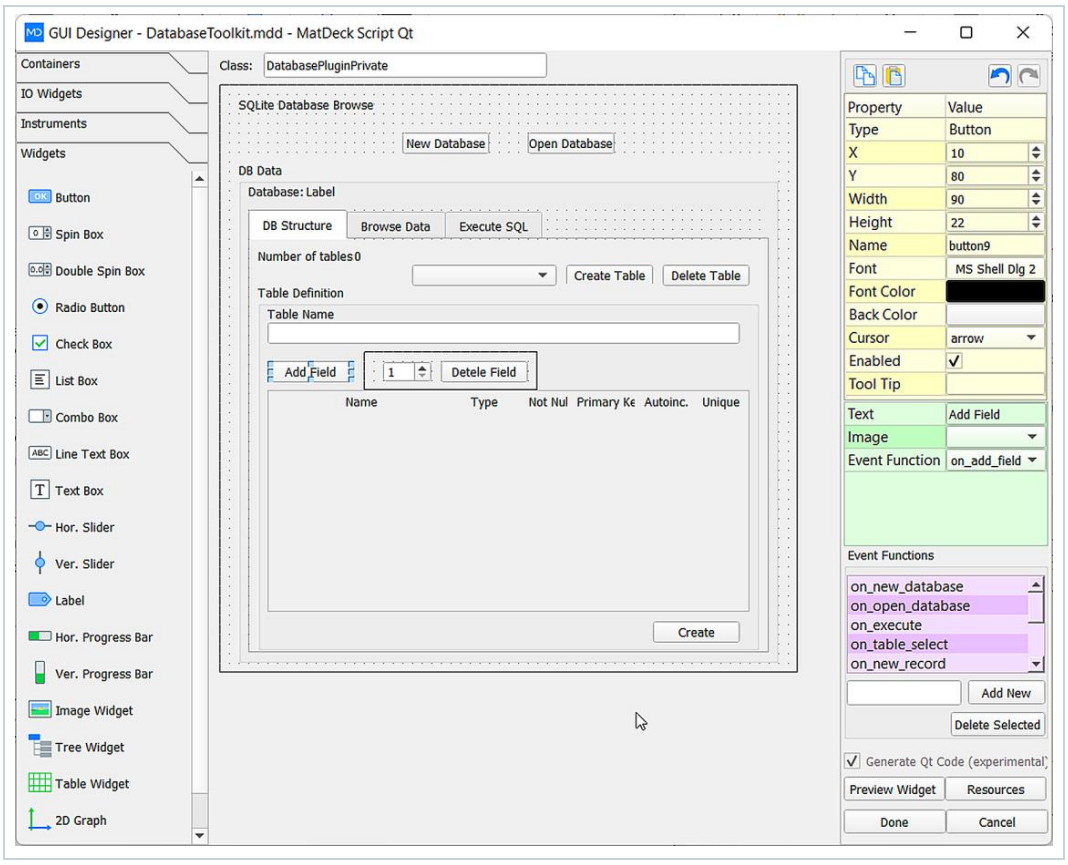
Introduction

The Unlimited Flet GUI Designer is available in MD Python Designer, Engineering Designer, Visionary Deck, MatDeck Student, MatDeck Academic, MatDeck Home and MatDeck. Its limited form is available in Lite MD Python Designer.

MatDeck products now offer nine GUI Designers — MatDeck Script (Qt runtime), MD Python, PySide2, Tkinter, Custom Tkinter, Kivy, Flet, Julia (QML) and Rust (egui). MD Python Designer and higher MD products have unlimited access to every GUI Designer, whereas Lite MD Python Designer is limited to only a few elements and MatDeck Free does not include any MD GUI Designer.

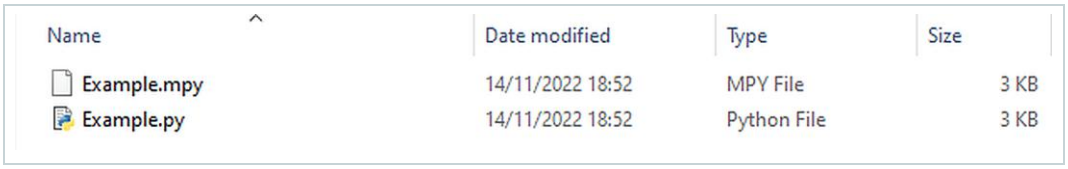
MD GUI Designers use drag-and-drop visual aids to build a fully functioning GUI with a sleek, modern interface; the widgets and components are generated automatically, leaving no boilerplate work to the user. MD Python Designer ships with every language, library, module and extension already installed, so no extra setup is needed and you can start creating straight away.

MD Python Designer also comes with a specialised IDE and GUI Designer for Flet. As mentioned above, all necessary libraries and modules come preinstalled.



This is what a MD GUI Designer looks like. As you can see, the form itself contains no code and was created solely by dragging and dropping GUI elements into the work area.

> **Please note** Our GUI Designers generate two files. One is a normal Python file; the other is used by the GUI Designer to save and remember what your GUI looked like. This second file has the .mpy extension and cannot be opened like a Python file.



Available widgets

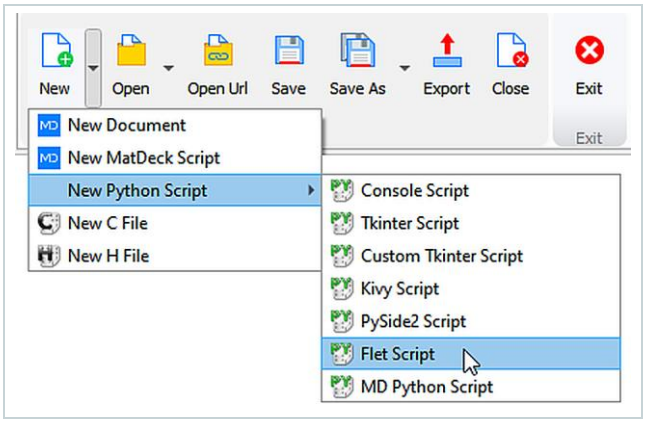
The Widgets tab of the Flet GUI Designer always provides the core controls, and the Containers, Inputs & data and Graphs groups add the elements listed below.

Category	Widgets available
Core widgets	Button, Label, Radio Button, Check Box, Combo Box, Line Edit, Text Box, Horizontal Slider, Vertical Slider, Horizontal Progress Bar
Containers	Root Widget, Group Box, Tab Widget, Scroll Area, Frame, Menu Bar, Tool Bar, Custom Widget
Inputs & data	Spin Box, Double Spin Box, List Box, Table, Tree Widget, Image, Vertical Progress Bar, Circular Progress, Toggle Switch, Spinner, Divider, Icon Button
Graphs	2D Graph

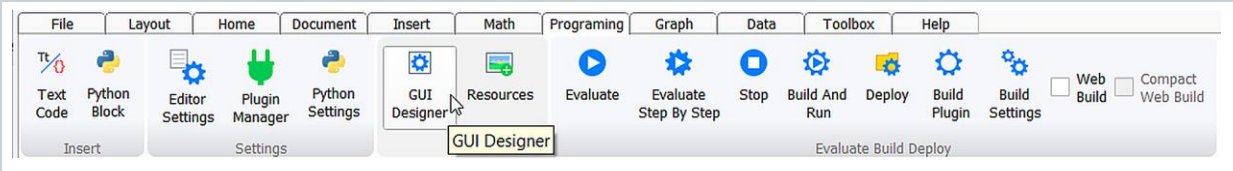
Feature support: Event functions and advanced element properties (tooltips, cursor, enabled state, fonts and colours) are fully supported in this designer.

Opening a MD GUI Designer

The first thing to do is select the Flet Script. This is done by hovering over the small triangle next to the New icon.

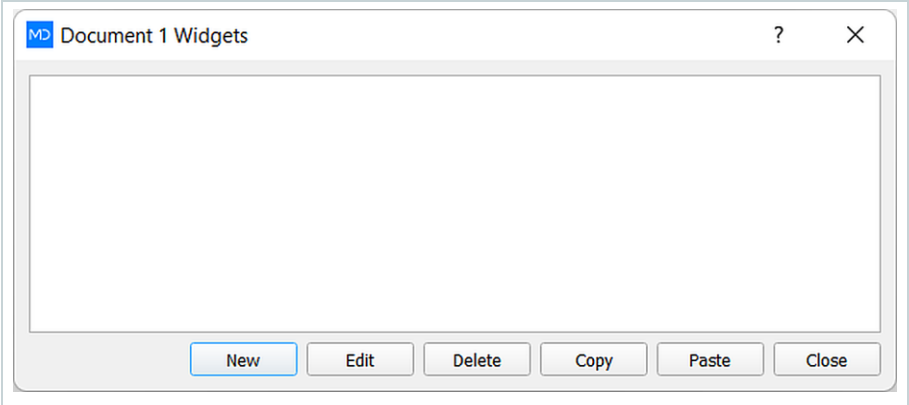


To open the Flet GUI Designer, go to the Programming tab and press the ‘GUI Designer’ button (Picture 1).



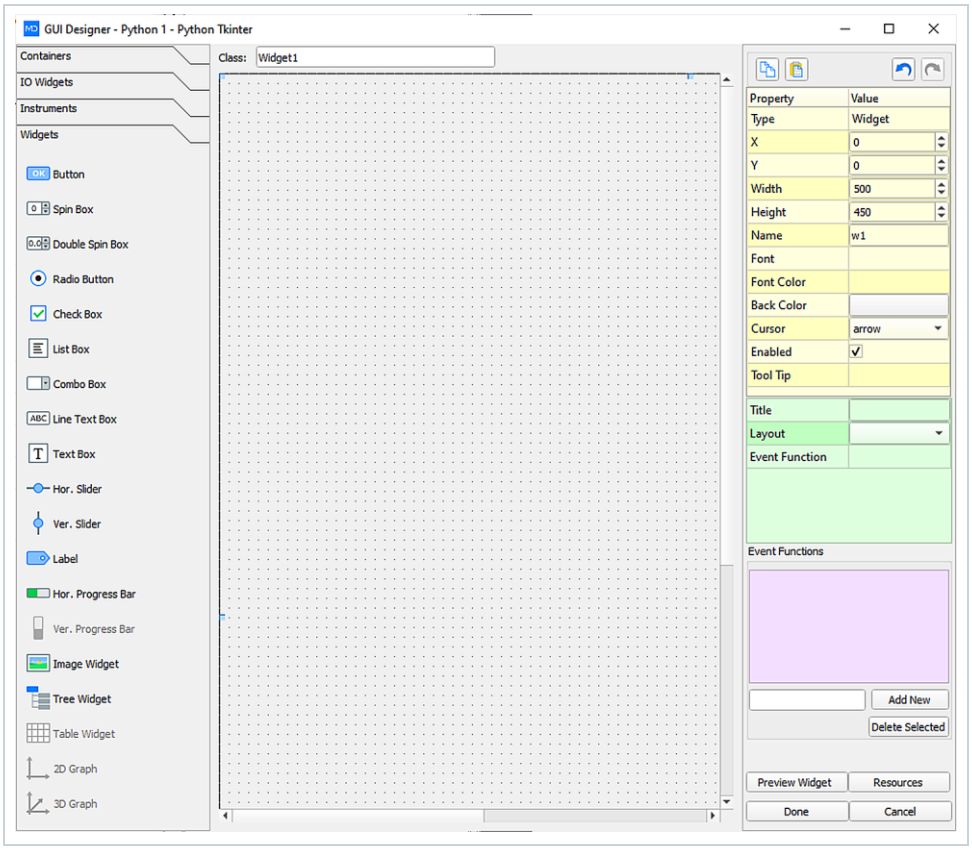
Picture 1: Widgets button

A new window opens containing a list of all widgets created in the current Flet Script (Picture 2). From this window you can create a new widget, or edit or delete an existing one. To open the Flet GUI Designer, click New; the Edit button only opens the designer when a widget already exists.



Picture 2: Document widgets editor

To create a GUI, press New and an empty Flet GUI Designer tab opens. When doing so, make sure no existing widget is selected in the window. To edit or delete a pre-existing widget, select it first, then press the appropriate button at the bottom of the tab. The default look of an empty Flet GUI Designer is shown below.

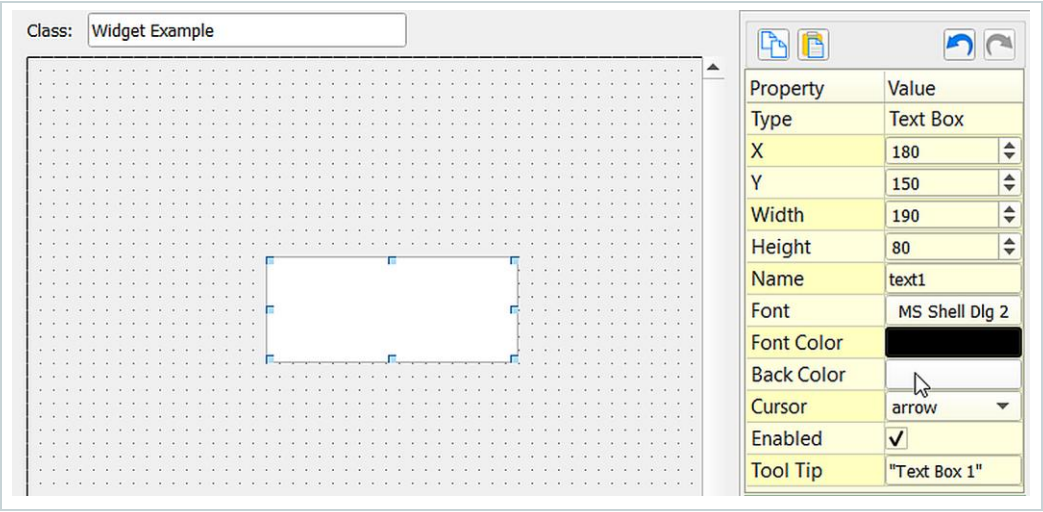


Picture 3: GUI Designer

On the left side of the Flet GUI Designer are all the GUI elements; the middle is the working area, where widgets and elements are placed. To place an element, click the preferred GUI element and then click where on the working area you want it.

For example, to add a Text Box, click the Text Box button on the left and then click anywhere on the working area.

On the right-hand side, the properties of the selected element are displayed and can be modified (Picture 4). In the example below we have also set the widget name to 'Widget Example' — you do this by typing the preferred name in the field above the main working area.

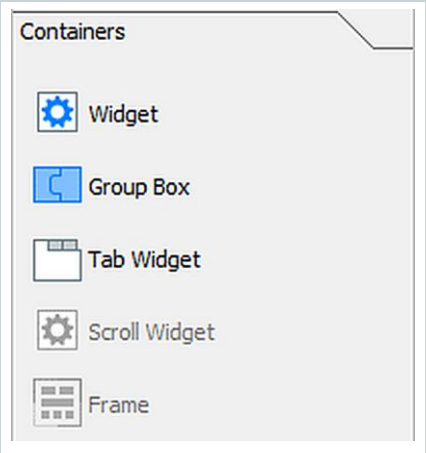


Picture 4: GUI Element Property

Please note that to connect or interact between GUI elements you must add the interaction code yourself by editing the generated code. We focus on saving you time, money and effort creating the GUI and on simplifying the required coding; what your code ultimately does is left to your imagination.

Containers

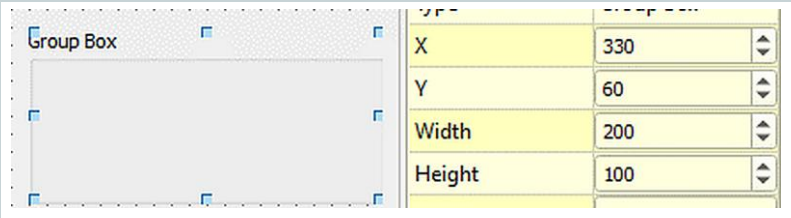
Opening the Containers menu from the top-left of the Flet GUI Designer reveals a list of container elements. Containers are placed on the main widget area and act as carriers for other GUI elements.



Picture 5: Containers

To insert a container, use the same method as any other GUI element: select the container, then click where on the working area you want to place it.

When a container or element is selected it can be resized by dragging the blue handles around it, or by changing its Width and Height values in the Property table on the right.

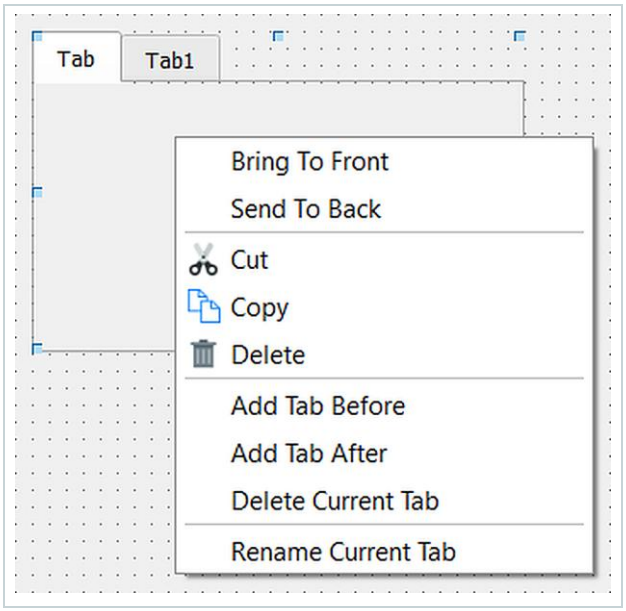


Picture 6: Resizing of object

To preview your widget and check your progress at any point, press the ‘Preview Widget’ button in the bottom-right corner. It opens a new interactive window where you can test your widget.

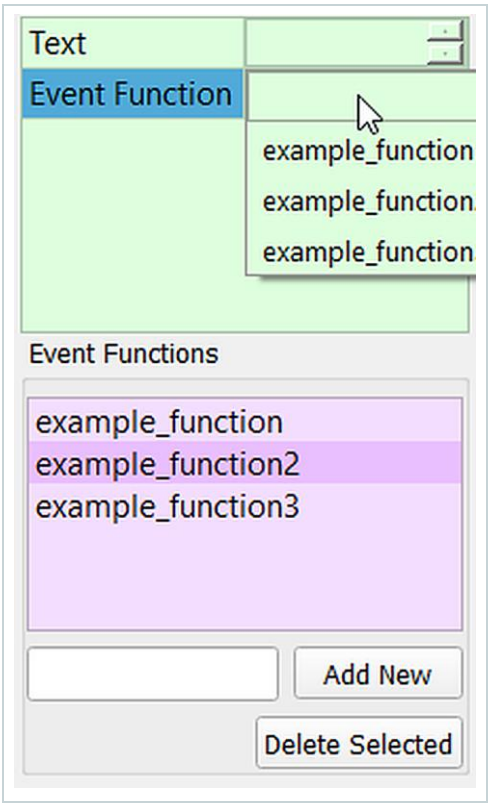
Event functions

When a Tab widget is used, the context menu lets you rename the current tab or insert a new one — right-click the widget to open it.



Tab widget context menu

You can add event functions to selected GUI elements. The Event Function list is empty on every newly created widget; define a new one in the Event Functions area of the designer, which is colour-coded green and purple. Type a name in the empty field and press ‘Add New’; the new function then appears in the Event Function list for the selected element (Picture 7).



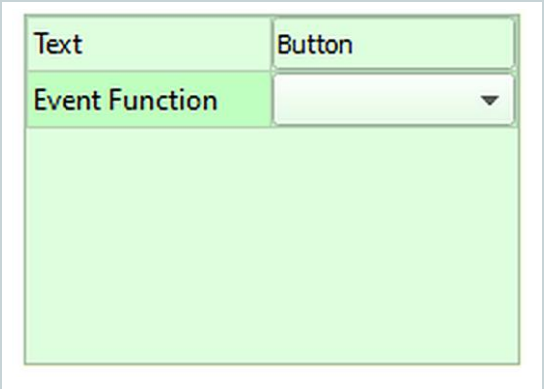
Picture 7: Event Functions

The Event Functions container is a global container holding every event function created for the widget you are editing. In the ‘Event Function’ field you can select one of the defined functions from the drop-down list.

The green panel is used to edit certain aspects of GUI elements — primarily the event function — and is not available for every element. Not all elements share the same options; some have settings specific only to that element. Each case is explained below.

Widget settings

Button



When a Button is inserted, the green panel above appears on the left. The ‘Text’ field sets the visible label on the button, and you can choose the Event Function that runs on the button’s click event.

Check Box

A Check Box lets the user toggle an option on or off. The ‘Text’ field sets its label (default ‘Check Box’) and the ‘Value’ attribute sets whether it starts checked. The Event Function runs when the check box is toggled.

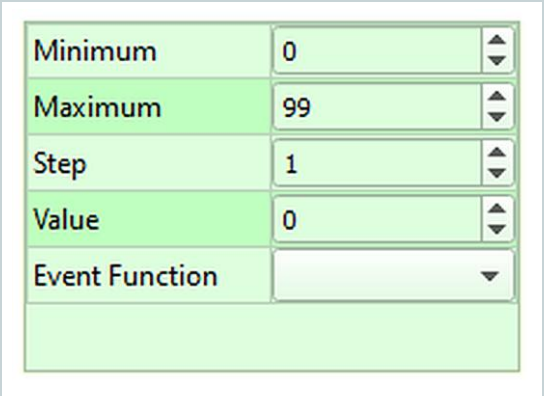
Combo Box

A Combo Box presents a drop-down list of selectable entries. Its rows are populated from code, and the Event Function runs when the selection changes.

Line Edit and Text Box

A Line Edit (Line Text Box) accepts a single line of typed text, while a Text Box accepts multiple lines. Both read and write their contents from code, and a Line Edit’s Event Function runs when its contents change.

Spin Box

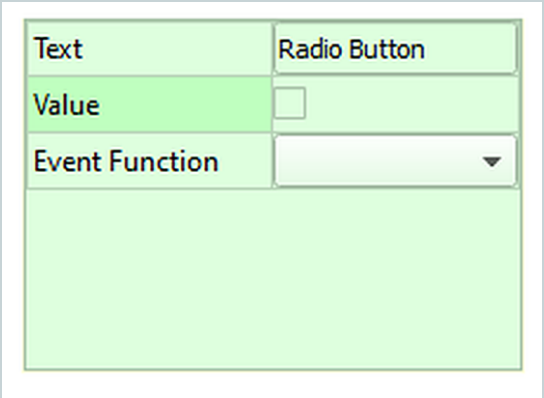


When a Spin Box is inserted, element-specific settings appear. Set the range with the ‘Minimum’ and ‘Maximum’ fields, the increment with ‘Step’, and the starting value with ‘Value’. An Event Function can be attached through the Event Function combo box.

Double Spin Box

A Double Spin Box behaves like a Spin Box but accepts fractional (floating-point) values. Use ‘Minimum’, ‘Maximum’, ‘Step’ and ‘Value’ to configure its range and precision.

Radio Button

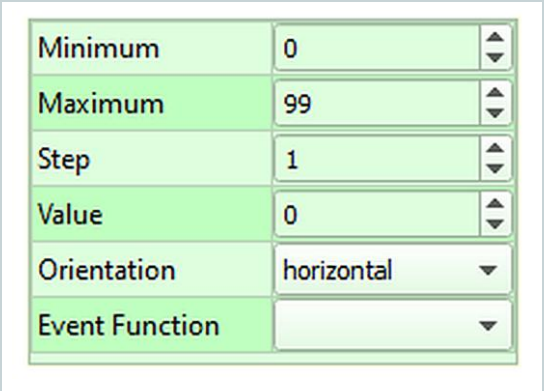


When a Radio Button is inserted, the settings above appear. Set its label in the ‘Text’ field (default ‘Radio Button’) and its default state with the ‘Value’ attribute. Within a container that holds several radio buttons, only one can be checked at a time. The Event Function runs when the radio button is checked.

List Box

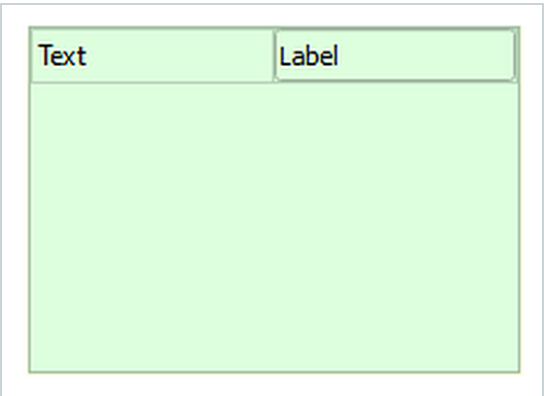
When a List Box is inserted, the following options appear. Use the ‘Value’ attribute to define its default value. The Event Function runs when one of the list rows is selected.

Horizontal and Vertical Slider



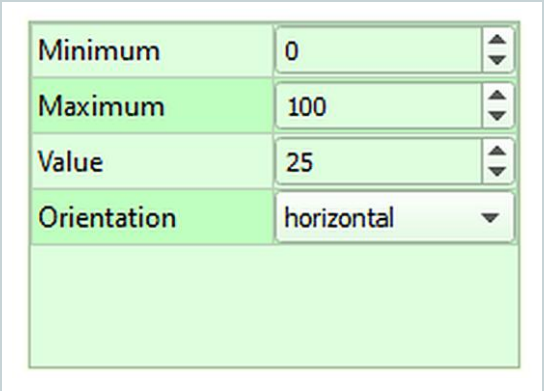
When a Slider is inserted, these options appear. Set the range with ‘Minimum’ and ‘Maximum’, the increment with ‘Step’, and the starting value with ‘Value’. The ‘Orientation’ drop-down switches between horizontal and vertical — the only two orientations available. When set, the Event Function runs whenever the slider value changes. Both horizontal and vertical sliders are on the Widgets tab.

Label



A Label has a single customisable setting: its visible text, which you set in the ‘Text’ field at the top.

Progress Bars



When a Progress Bar is inserted, set the allowed range with ‘Minimum’ and ‘Maximum’ and the starting value with ‘Value’. The ‘Orientation’ drop-down switches between horizontal and vertical; both the horizontal and vertical progress bars are available in this designer.

Image

When an Image element is inserted, set the picture with the ‘Image’ field at the top. The Event Function runs when the image is clicked.

Circular Progress

A Circular Progress indicator shows completion as a ring rather than a bar. Set its range with ‘Minimum’ and ‘Maximum’ and its current value with ‘Value’.

Toggle Switch

A Toggle Switch is an on/off control styled as a sliding switch. The ‘Value’ attribute sets its default state and the Event Function runs when it is toggled.

Divider

A Divider draws a thin separating line to group parts of the layout; it has no interactive value.

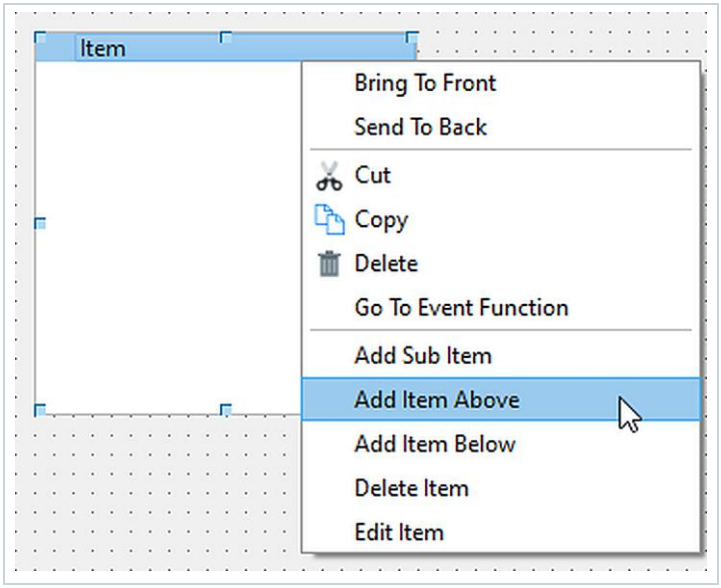
Icon Button

An Icon Button behaves like a Button but shows an icon instead of, or alongside, a text label. Set its icon with the ‘Image’ field and attach an Event Function for its click event.

Spinner

A Spinner is an animated busy indicator that shows work is in progress; it has no value to set.

Tree Widget



Tree widget context menu

When a Tree Widget is inserted, set its icon with the ‘Image’ field at the top. The Event Function runs when a tree item is clicked. To add an item, right-click the tree and choose Add Item; to add a sub-item, right-click the parent item and choose Add Sub Item; to rename an item, right-click it and choose Edit Item.

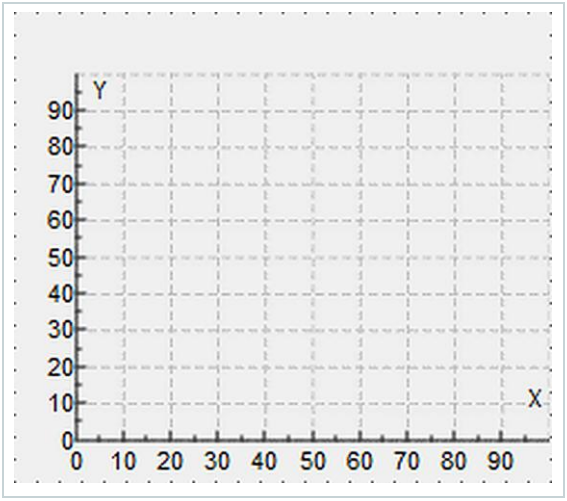
Tables and 2D/3D graphs have no property-panel settings; all of their properties are set from their context menus, and their values are assigned from code (see below).

Using tables and 2D graphs

Graphs and tables cannot be populated in the GUI Designer itself; they are customised there, but their values are only assigned from code.

2D graphs

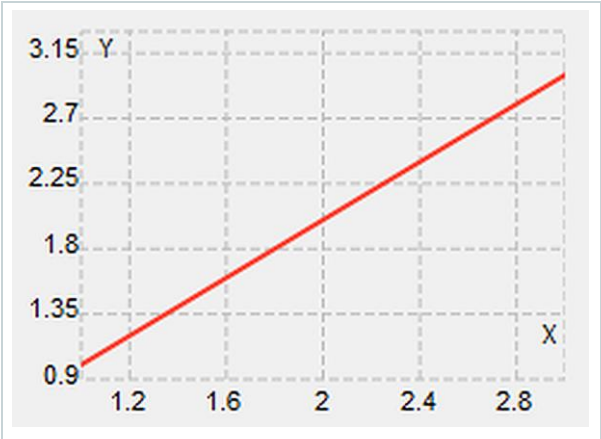
Assign a 2D graph’s value with an array. The first line is MatDeck Script; the second is Python using the MD Python library.



2D graph

```
set_widget_value(graph2d1, [1, 2, 3; 1, 2, 3])
set_widget_value(self.graph2d1, [[1, 2, 3], [1, 2, 3]])
```

With its value assigned it looks like this:

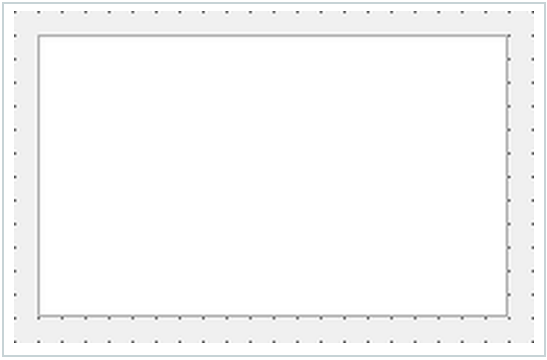


2D graph with values

3D graphs are available in the MatDeck Script and MD Python designers.

Tables

Assign a table's value with an array, just like a 2D graph. Columns are assigned automatically; change them with the table_create() function.



Table

```
set_widget_value(table1, [1, 2, 3; 1, 2, 3])
set_widget_value(self.table1, [[1, 2, 3], [1, 2, 3]])
```

With its value assigned it looks like this:

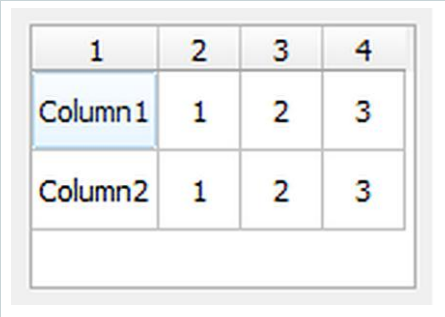
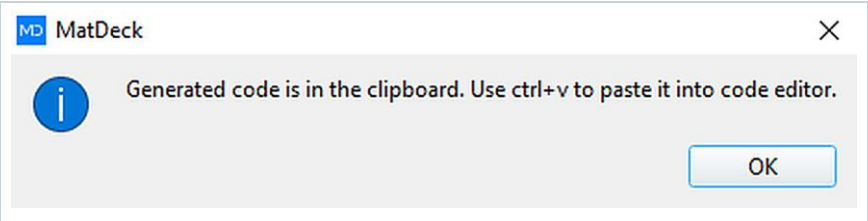


Table with values

Finishing up

When you finish your widget design, press ‘Done’. The designer generates the widget code and pastes it into the script you are working in; if you are working in a MatDeck document, the code is copied to your clipboard so you can paste it anywhere. The first time you press ‘Done’ for a new widget, the notification window (Picture 8) appears. After that first time, code updates silently.



Picture 8: Notification window

As noted, pressing Done in a script generates the code directly into that script.

Using GUI Designer code in a document

To place the generated code in a MatDeck document, put the cursor at the desired position and turn on Code mode from the Programming tab (Text/Code), or press Ctrl + I. With Text Code mode active, paste the code from the clipboard into the document.

The generated code is copied to your clipboard as soon as you press OK. To paste it into a MatDeck script instead, open a new script file (New ▸ New Script), then paste and use the code as you wish.

```
class Widget_Example
{
  w1 := 0
  text1 := 0
  group1 := 0
  //your members
  //

  Widget_Example(parent)
  {
```

```
    gui(parent)
  }

  gui(parent)
  {
    if(w1 != 0)
    {
      return(w1)
    }
    ...
  }
}
```

Widget code

For every event function created in the GUI Designer, a matching function appears with an empty body for you to fill in.

```
Example_function()
{

}

a := Widget_Example(0)
show(a.w1)
```

Event function in code

At the end of the code a constructor is added so the widget can be shown as a stand-alone application when it is evaluated.

> **Notice** The preferred workflow is to create and change widget designs in the GUI Designer, then generate or update the code from there. If you edit the code directly and afterwards open and save changes in the GUI Designer, any edits made directly in the code will be lost.

Interacting with generated GUI code

The GUI Designer generates the code that creates the visual side of the GUI — element positions, titles and event functions are all coded exactly as designed. Any interaction or change beyond that must be added by you. Two functions are used most often.

Retrieving the value of a widget — widget_value()

Use widget_value() to read and store the value of a widget; it takes one argument, the widget name.

```
A := widget_value(ExampleWidget)
```

Here the value of the widget ExampleWidget is read and stored in the variable A.

Connecting widget values — set_widget_value()

Use set_widget_value() to display one widget's value on another. It takes two arguments: the target widget and the value to display.

```
set_widget_value(instrument1, widget_value(vslider1))
```

Here the target is a circular instrument gauge and the value comes from a vertical slider, read with widget_value(). After the call, the instrument shows the slider's value.

Interactions should be placed inside the relevant event function — here, inside the slider's event function:

```
SliderEventFunction()
{
    set_widget_value(instrument1, widget_value(vslider1))
}
```

Executing and outputting GUIs

Once a design is complete, the generated code is copied to the clipboard or pasted straight into the script. It can be used in MatDeck documents, Python scripts and MatDeck Script. GUIs can be output in the following ways:

- Evaluation of generated code
- Embedding the widget

Evaluation of generated code

Evaluating the generated code runs the GUI application. By default it appears as a stand-alone window separate from the MatDeck document it was evaluated in. Place the code in a programming block, MatDeck Script or Python Script, then press the ‘Evaluate’ icon in the Programming toolbar.

The Programming toolbar — Evaluate