# FreeCAD GUI Modern API

| META | VALUE |
|---|---|
| **generated** | 2025-12-24 09:05:44.173262 |
| **author** | Frank David Martínez Muñoz |
| **copyright** | (c) 2024 Frank David Martínez Muñoz. |
| **license** | LGPL 2.1 |
| **version** | 1.0.0-beta5 |
| **min_python** | 3.10 |
| **min_freecad** | 0.22 |

# TABLE OF CONTENTS

# Preliminaries

## Disclaimer

All of the following information is the result of my own research and usage of the FreeCAD's Python APIs and Qt/PySide along several years. It reflects my very own view, coding style and limited understanding of FreeCAD internals. All the content is based on official docs, forum discussions, development of my own extensions, reading code of existing extensions and FreeCAD sources.

This document does not cover 100% of the Qt/PySide API because it is huge and not really required for common Macros in FreeCAD.

## Audience

This is a technical document for developers of FreeCAD extensions using Qt/PySide as they GUI Framework.

General programming experience, some basic FreeCAD know-how and a minimalistic comprehension of Python are sufficient, as long as you can search the internet for a basic grasp of classes, functions, decorators, type hints, etc...;)

It is also expected that the readers are FreeCAD users, and have a good understanding of the basic usage of it.

## Goals

- The API must be developer friendly, consistent, maintainable and compatible with FC 0.21+
- The API must be an overlay on top of the existing PySide API, so no conflicts with existing code.
- The API must be 100% documented.

## Non Goals

- It is not intended to replace anything in the existing FreeCAD APIs.
- It is not intended to require any refactoring of existing python code.
- It is not intended to require any refactoring of existing C/C++ code.

# Features

- √ Declarative Layout
    - √ Code layout reflects GUI Structure
- √ Custom input widgets
    - √ InputText
    - √ Numeric inputs: InputInt, InputFloat, InputQuantity
    - √ List inputs: InputFloatList
    - √ InputBoolean
    - √ InputVector
    - √ InputOptions
    - √ Selection input: InputSelectOne, InputSelectMany
    - √ Buttons
- √ Custom view widgets
    - √ Html
    - √ Image
    - √ Svg
    - √ Table
    - √ Canvas
- √ Documentation in markdown format.

# Widgets

## Containers / Layouts

Widget: Dialog

### Signature / Dialog

```python
def Dialog(
        title: str | None=None,
        *,
        size: tuple[int, int] | None=None,
        show: bool=True,
        modal: bool=True,
        parent: QWidget=None,
        **
        kwargs: KwArgs
    ) -> Generator[DialogWidget, Any, Any]: ...
```

### Gui / Dialog


### Docs / Dialog

| RETURN TYPE | DESCRIPTION |
| --- | --- |
| QDialog | The Dialog |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| title | str | window title, defaults to None |
| size | tuple[int, int] | dialog size, defaults to None |
| show | bool | show automatically, defaults to True |
| modal | bool | window modality, defaults to True |
| parent | QWidget | parent widget, defaults to None |
| **kwargs | dict[str, Any] | Qt properties |

Dialog context manager/widget.

Example:

> *../examples/ui/widgets.py (Dialog)*

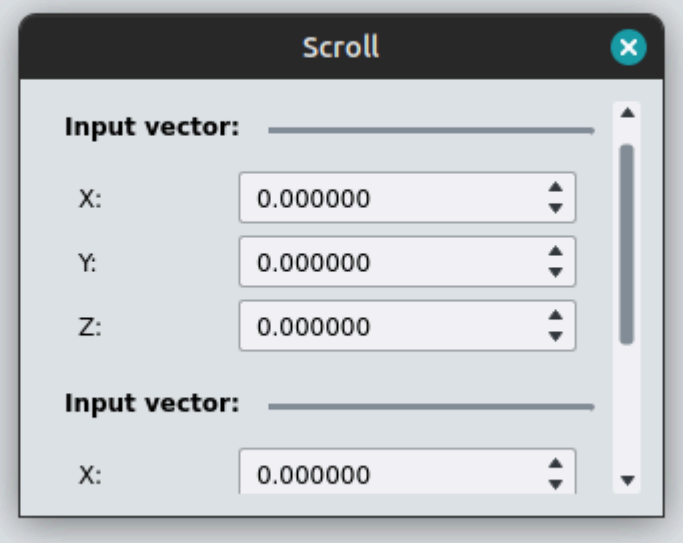
```python
def demo_Dialog():
    with ui.Dialog("Dialog"):
        with ui.Col():
            ui.TextLabel("This a simple text", alignment=Qt.AlignCenter)

```

## Widget: Scroll

### Signature / Scroll

```python
def Scroll(
        *,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> Generator[QScrollArea, Any, Any]: ...
```

## Gui / Scroll




## Docs / Scroll

| RETURN TYPE | DESCRIPTION |
|---|---|
| QScrollArea | Scroll widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| add | bool | add to context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Scrollable area context manager/widget.

Example:

> *../examples/ui/widgets.py (Scroll)*

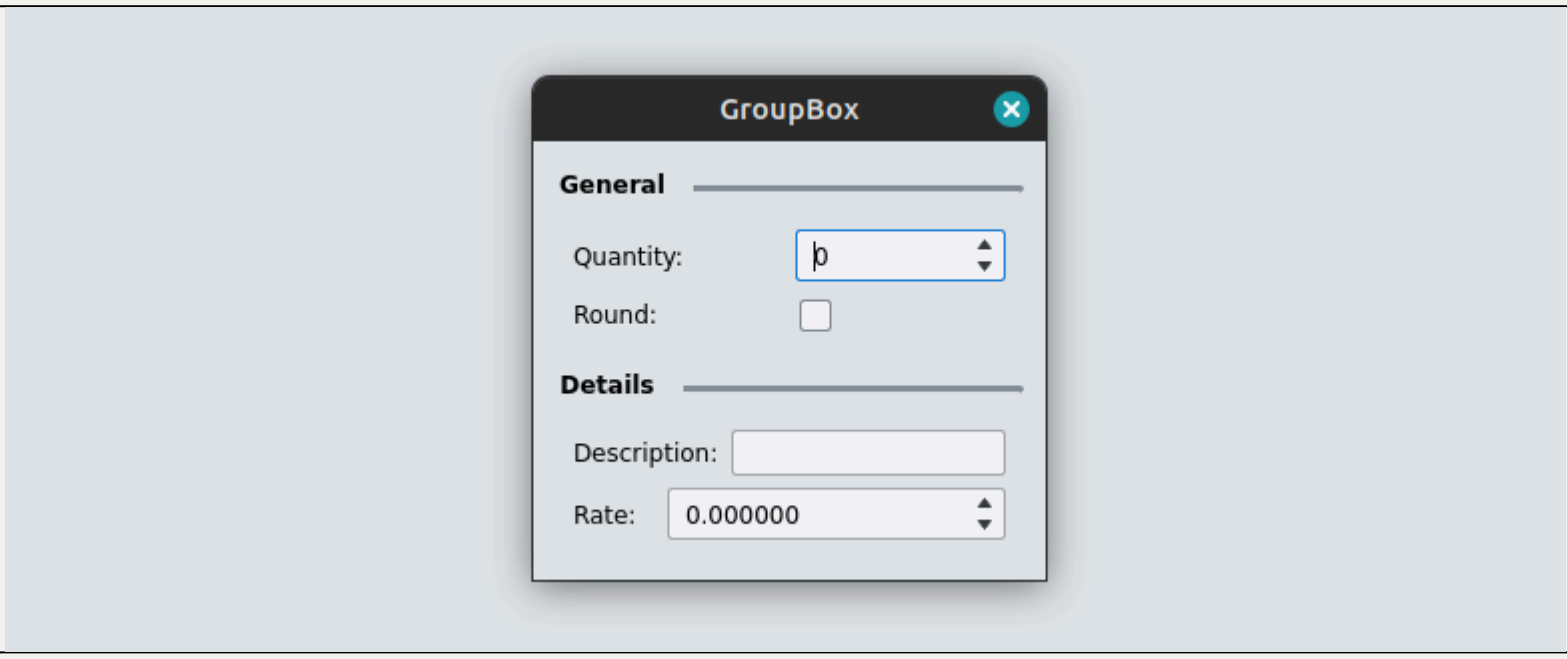
```
def demo_Scroll():
    with ui.Dialog("Scroll"):
        with ui.Scroll(widgetResizable=True):
            with ui.Container():
                _v1 = ui.InputVector(label="Input vector:")
                _v2 = ui.InputVector(label="Input vector:")

```

# Widget: GroupBox

## Signature / GroupBox

```
def GroupBox(
        title: str | None=None,
        *,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> Generator[QGroupBox, Any, Any]: ...
```

## Gui / GroupBox




## Docs / GroupBox

| RETURN TYPE | DESCRIPTION |
|---|---|
| QGroupBox | The group widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| title | str | Group title, defaults to None |
| add | bool | add to context, defaults to True |
| add | bool | add to context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

GroupBox context manager/widget.

Example:

> *../examples/ui/widgets.py (GroupBox)*

```python
def demo_GroupBox():
    with ui.Dialog("GroupBox"):
        with ui.GroupBox("General"):
            _quant = ui.InputInt(label="Quantity:")
            _round = ui.InputBoolean(label="Round:")
        with ui.GroupBox("Details"):
            _desc = ui.InputText(label="Description:")
            _rate = ui.InputFloat(label="Rate:")

```

## Widget: Container

### Signature / Container

```python
def Container(
        *,
        add: bool=True,
        top: bool=False,
        q_widget: type[QWidget] | None=None,
        **
        kwargs: KwArgs
    ) -> Generator[QWidget, Any, Any]: ...
```

### Docs / Container

| RETURN TYPE | DESCRIPTION |
|---|---|
| QFrame | The container widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| add | bool | add to context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Simple container context/widget.

Example

> *../examples/ui/widgets.py (Scroll)*

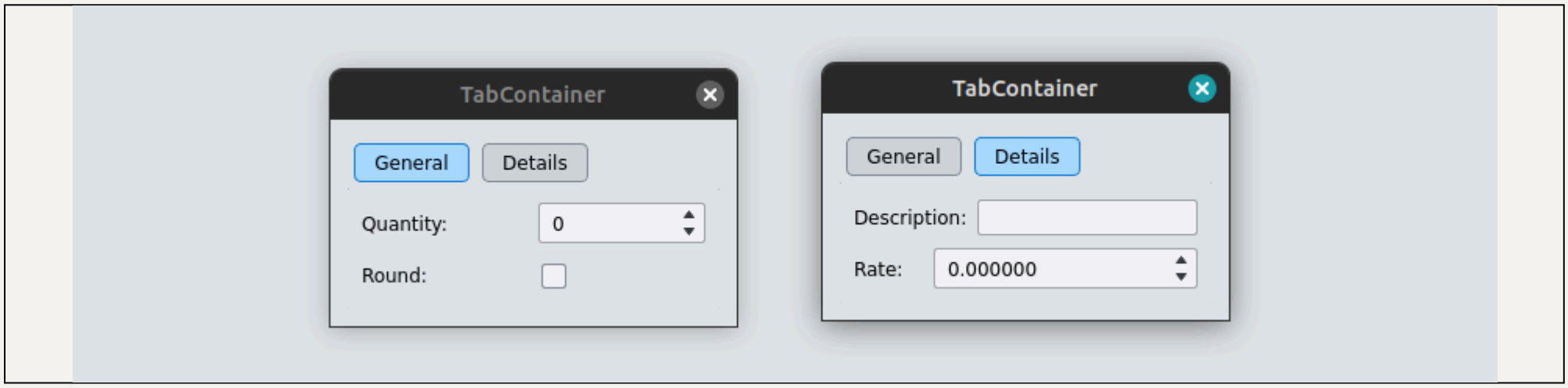
```python
def demo_Scroll():
    with ui.Dialog("Scroll"):
        with ui.Scroll(widgetResizable=True):
            with ui.Container():
                _v1 = ui.InputVector(label="Input vector:")
                _v2 = ui.InputVector(label="Input vector:")

```

# Widget: TabContainer

### Signature / TabContainer

```python
def TabContainer(
        *,
        stretch: int=0,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> Generator[QTabWidget, Any, Any]: ...
```

### Gui / TabContainer




### Docs / TabContainer

| RETURN TYPE | DESCRIPTION |
|---|---|
| QTabWidget | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| stretch | int | stretch, defaults to 0 |
| add | bool | add to the context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Tab Container context/widget

Example:

> *../examples/ui/widgets.py (TabContainer)*

```
def demo_TabContainer():
    with ui.Dialog("TabContainer"):
        with ui.TabContainer():
            with ui.Tab("General"):
                _quant = ui.InputInt(label="Quantity:")
                _round = ui.InputBoolean(label="Round:")
            with ui.Tab("Details"):
                _desc = ui.InputText(label="Description:")
                _rate = ui.InputFloat(label="Rate:")

```

## Function: Tab

### Signature / Tab

```
def Tab(
        title: str,
        *,
        icon: QIcon=None,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> Generator[QWidget, Any, Any]: ...
```

### Docs / Tab

| RETURN TYPE | DESCRIPTION |
|---|---|
| QWidget | the widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| title | str | Tab's title |
| icon | QIcon | Icon, defaults to None |
| add | bool | add to the context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Tab widget/context in a tab container

Example:

> *../examples/ui/widgets.py (TabContainer)*

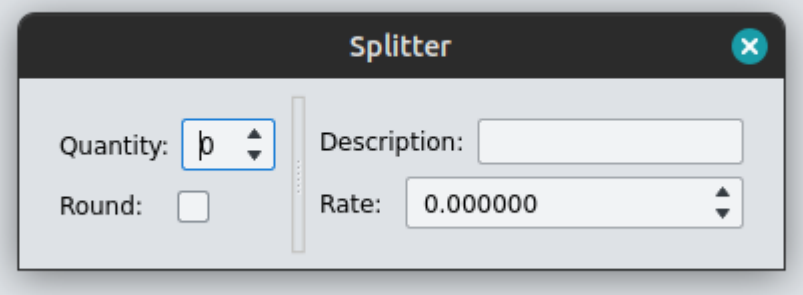
```python
def demo_TabContainer():
    with ui.Dialog("TabContainer"):
        with ui.TabContainer():
            with ui.Tab("General"):
                _quant = ui.InputInt(label="Quantity:")
                _round = ui.InputBoolean(label="Round:")
            with ui.Tab("Details"):
                _desc = ui.InputText(label="Description:")
                _rate = ui.InputFloat(label="Rate:")

```

## Widget: Splitter

### Signature / Splitter

```python
def Splitter(*, add: bool=True, **kwargs: KwArgs) -> Generator[QSplitter, Any, Any]: ...
```

### Gui / Splitter




### Docs / Splitter

| RETURN TYPE | DESCRIPTION |
|---|---|
| QSplitter | The splitter widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Split context/container

Example:

> *../examples/ui/widgets.py (Splitter)*

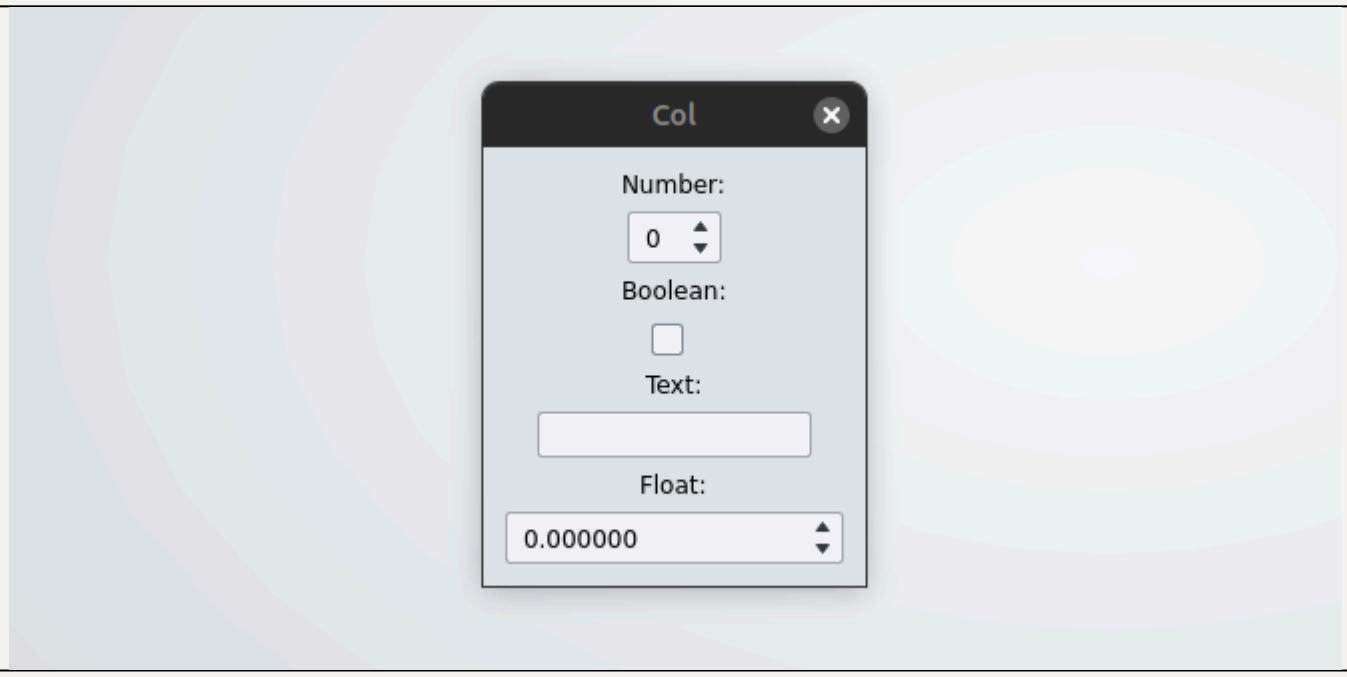
```python
def demo_Splitter():
    with ui.Dialog("Splitter"):
        with ui.Splitter():
            with ui.Container():
                _quant = ui.InputInt(label="Quantity:")
                _round = ui.InputBoolean(label="Round:")
            with ui.Container():
                _desc = ui.InputText(label="Description:")
                _rate = ui.InputFloat(label="Rate:")

```

# Widget: Col

### Signature / Col

```python
def Col(*, add: bool=True, **kwargs: KwArgs) -> Generator[QWidget, Any, Any]: ...
```

### Gui / Col




### Docs / Col

| RETURN TYPE | DESCRIPTION |
|---|---|
| QWidget | A container widget with Vertical layout |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Vertical context/layout

Example:

> *../examples/ui/widgets.py (Col)*

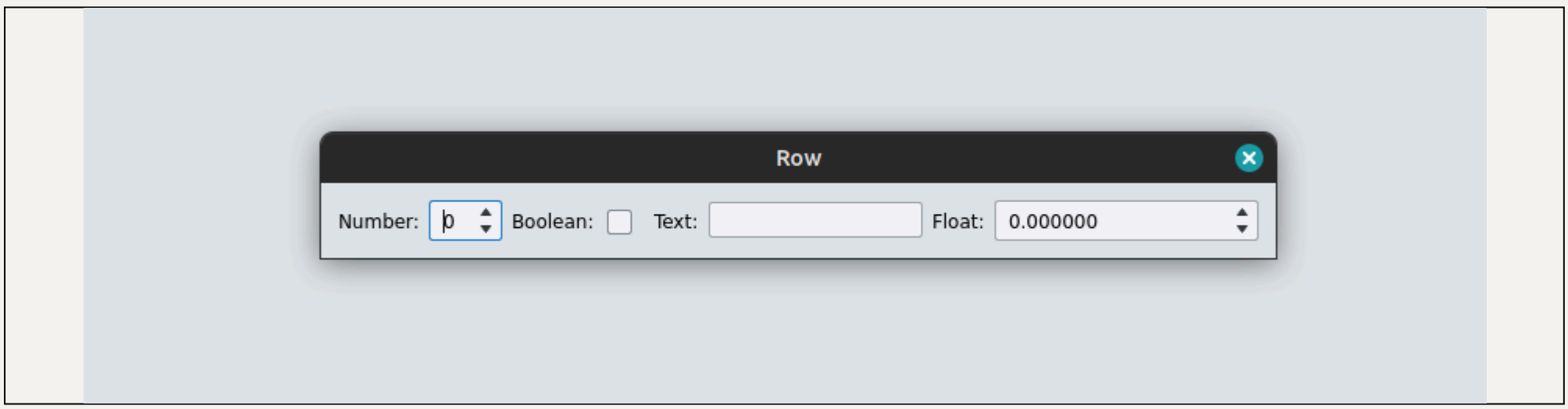
```python
def demo_Col():
    with ui.Dialog("Col"):
        with ui.Col():
            ui.TextLabel("Number:", alignment=Qt.AlignCenter)
            ui.InputInt(alignment=Qt.AlignCenter)
            ui.TextLabel("Boolean:", alignment=Qt.AlignCenter)
            ui.InputBoolean(alignment=Qt.AlignCenter)
            ui.TextLabel("Text:", alignment=Qt.AlignCenter)
            ui.InputText(alignment=Qt.AlignCenter)
            ui.TextLabel("Float:", alignment=Qt.AlignCenter)
            ui.InputFloat(alignment=Qt.AlignCenter)

```

## Widget: Row

### Signature / Row

```python
def Row(*, add: bool=True, **kwargs: KwArgs) -> Generator[QWidget, None, None]: ...
```

### Gui / Row




### Docs / Row

| RETURN TYPE | DESCRIPTION |
| --- | --- |
| QWidget | A container widget with Horizontal layout |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Horizontal context/layout

Example:

> *../examples/ui/widgets.py (Row)*

```
def demo_Row():
    with ui.Dialog("Row"):
        with ui.Row():
            ui.TextLabel("Number:")
            ui.InputInt()
            ui.TextLabel("Boolean:")
            ui.InputBoolean()
            ui.TextLabel("Text:")
            ui.InputText()
            ui.TextLabel("Float:")
            ui.InputFloat()

```

## Output

Widget: TextLabel

### Signature / TextLabel

```
def TextLabel(
        text: str='',
        *,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> QLabel: ...
```

## Gui / TextLabel


## Docs / TextLabel

| RETURN TYPE | DESCRIPTION |
|---|---|
| QLabel | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| text | str | text, defaults to "" |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties of QLabel |

Simple text label widget.

Example:

> *../examples/ui/widgets.py (TextLabel)*

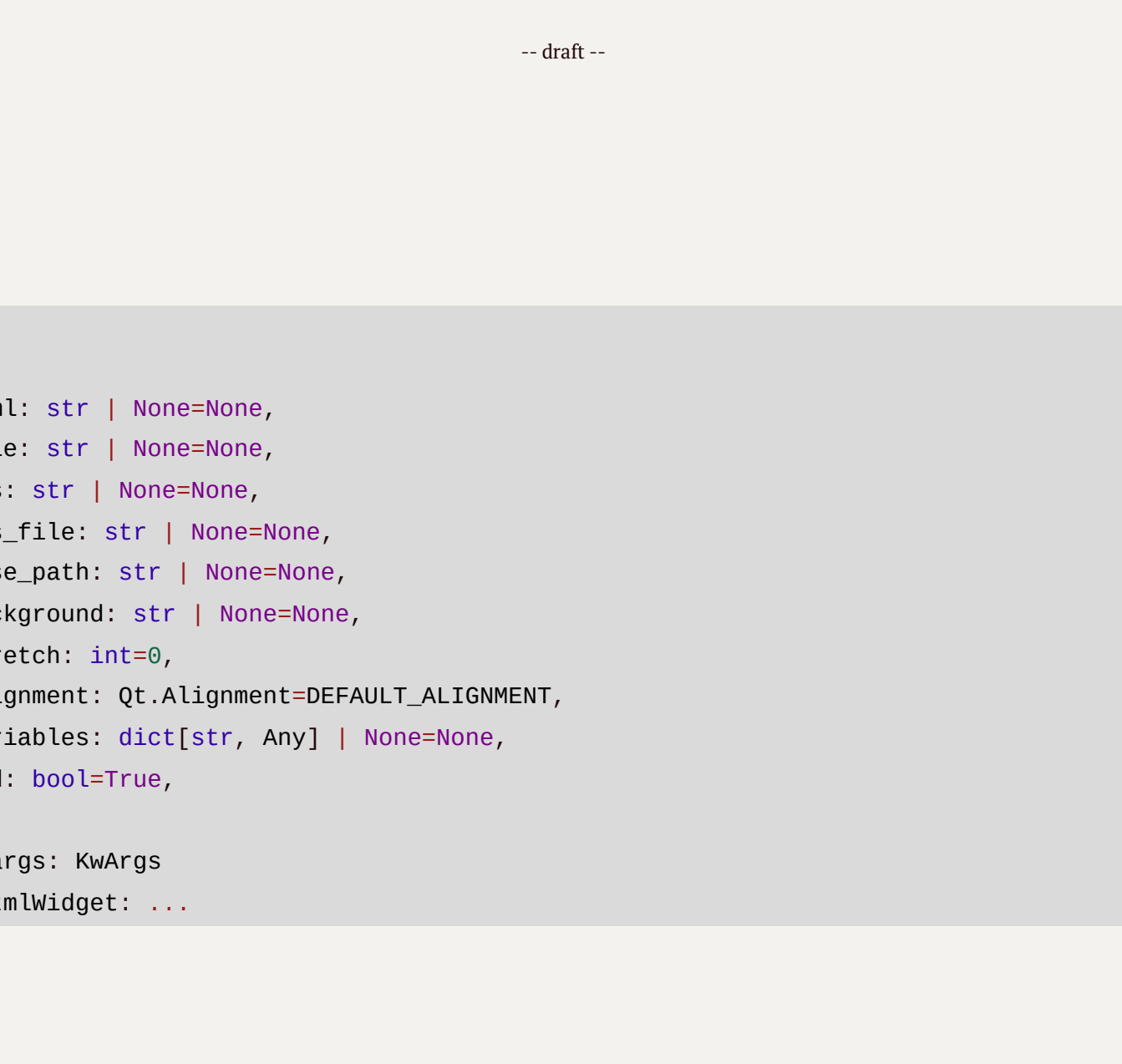
```
def demo_TextLabel():
    with ui.Dialog("TextLabel"):
        with ui.Col():
            ui.TextLabel("This is a text label", alignment=Qt.AlignCenter)

```

# Widget: Html

## Signature / Html

```
def Html(
        *,
        html: str | None=None,
        file: str | None=None,
        css: str | None=None,
        css_file: str | None=None,
        base_path: str | None=None,
        background: str | None=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        variables: dict[str, Any] | None=None,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> HtmlWidget: ...
```

## Gui / Html




## Docs / Html

| RETURN TYPE | DESCRIPTION |
|---|---|
| HtmlWidget | Html Widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| html | str | raw html content, defaults to None |
| file | str | path to html file, defaults to None |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| css | str | raw css code, defaults to None |
| css_file | str | path to css file, defaults to None |
| base_path | str | base dir for loading resources, defaults to None |
| background | str | background color code, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| variables | dict[str, Any] | interpolation variables, defaults to None |
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties of QLabel |

Basic HTML Render widget.

Example:

> *../examples/ui/widgets.py (Html)*

```
def demo_Html():
    with ui.Dialog("Html"):
        with ui.Col():
            ui.Html(
                background="#ffffff",
                html="""
                <h1>Header</h1>
                <ul>
                    <li>Item 1</li>
                    <li>Item 1</li>
                    <li>Item 1</li>
                </ul>
                <br />
                <p>
                <strong>Be Happy!!</strong>
                </p>
                """,
            )

```

## Widget: ImageView

### Signature / ImageView

```
def ImageView(
        uri: str,
        *,
```

```
        label: str | None=None,
        name: str | None=None,
        background: str | QColor | None=None,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> ImageViewWidget: ...
```

### Gui / ImageView


### Docs / ImageView

| RETURN TYPE | DESCRIPTION |
| --- | --- |
| ImageViewWidget | *description* |

| ARGUMENT | TYPE | DESCRIPTION |
| --- | --- | --- |
| uri | str | path to load the image from |
| label | str | gui label, defaults to None |
| name | str | objectName, defaults to None |
| background | Union[str, QColor] | background color, defaults to None |
| add | bool | *description*, defaults to True |

Image render widget.

Example:

> *../examples/ui/widgets.py (ImageView)*

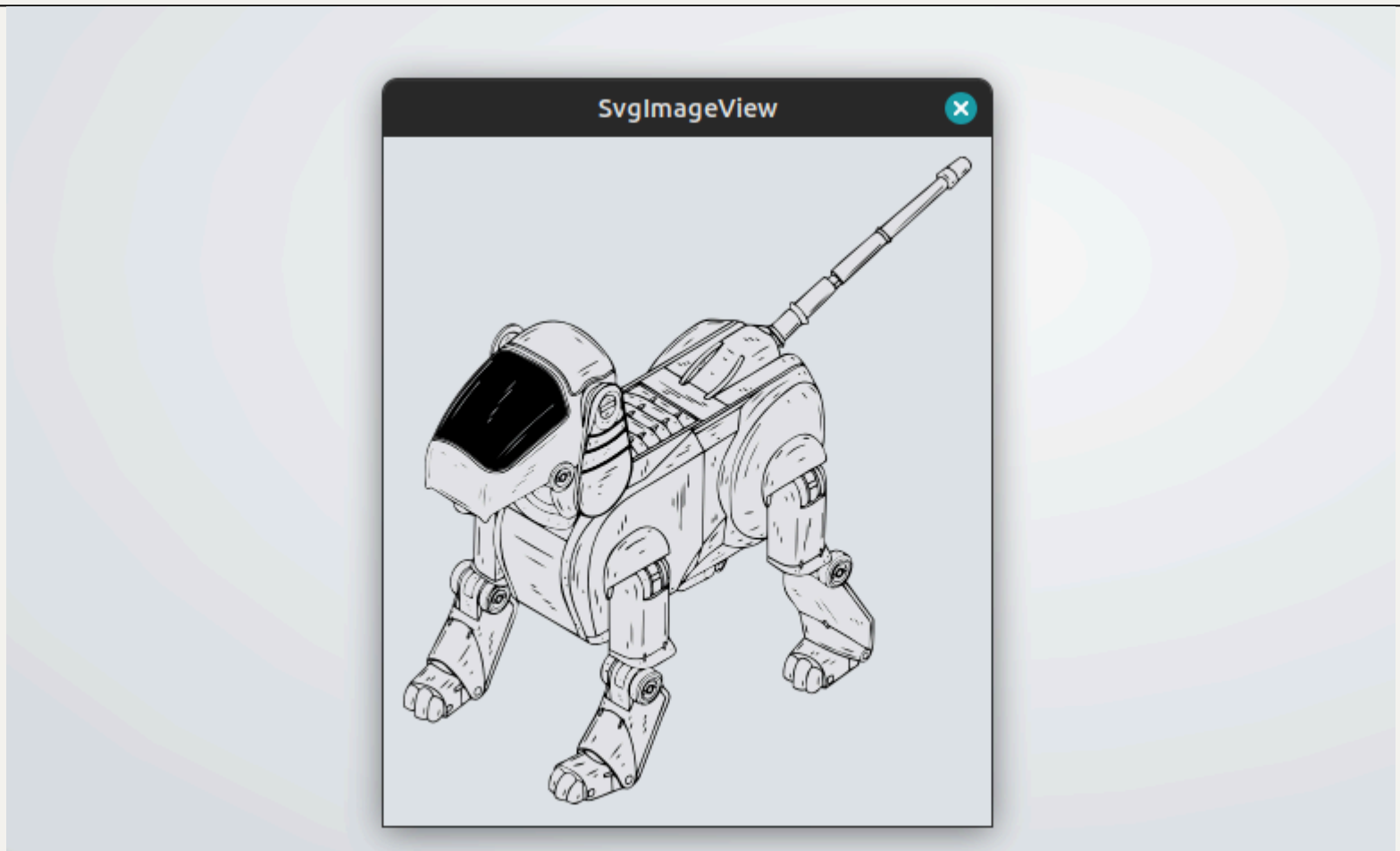
```
def demo_ImageView():
    with ui.Dialog("ImageView", size=(400, 300)):
        with ui.Col():
            ui.ImageView(
                str(Path(__file__).parent / "image.png"),
            )

```

## Widget: SvgImageView

### Signature / SvgImageView

```
def SvgImageView(
        uri: str,
        label: str | None=None,
        *,
        name: str | None=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        **
        kwargs: KwArgs
    ) -> SvgImageViewWidget: ...
```

### Gui / SvgImageView



### Docs / SvgImageView

| RETURN TYPE | DESCRIPTION |
|---|---|
| SvgImageViewWidget | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| uri | str | file path if the svg |
| name | str | Widget name, defaults to None |

High resolution Svg Image box.

Example:

> *../examples/ui/widgets.py (SvgImageView)*

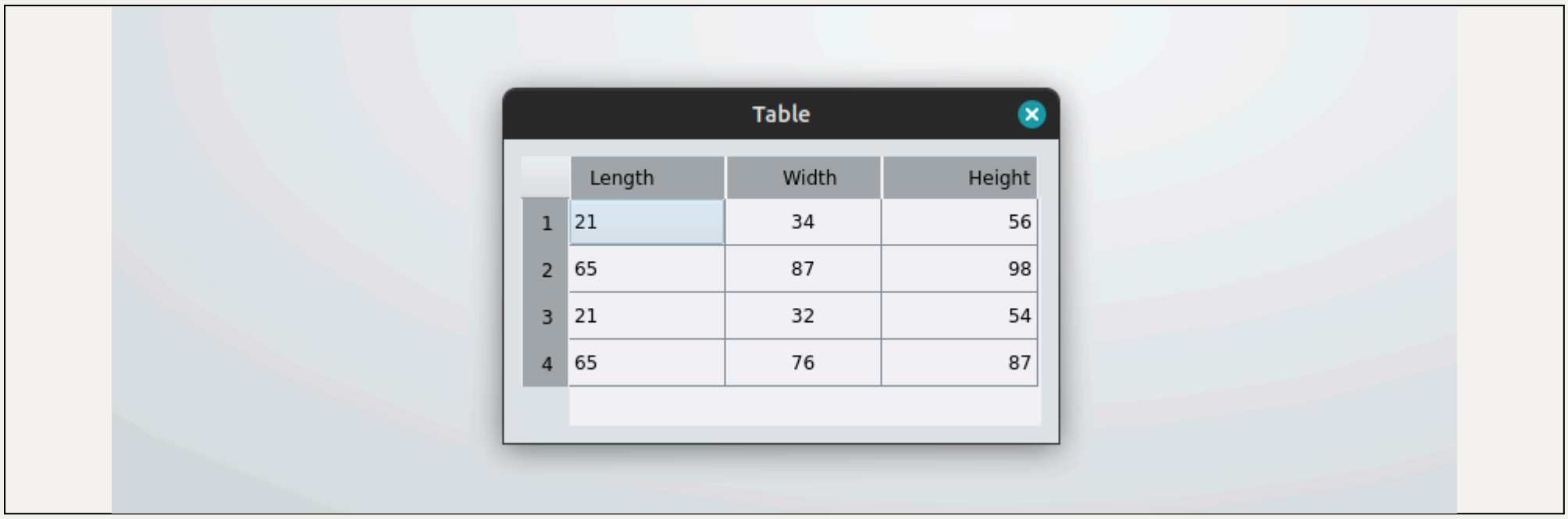
```
def demo_SvgImageView():
    with ui.Dialog("SvgImageView"):
        with ui.Col():
            ui.SvgImageView(
                str(Path(__file__).parent / "vector.svg"),
            )

```

## Widget: Table

### Signature / Table

```
def Table(
        headers: list[str],
        rows: list[list[Any]],
        *,
        name: str | None=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> TableWidget: ...
```

## Gui / Table




## Docs / Table

| RETURN TYPE | DESCRIPTION |
|---|---|
| TableWidget | The table widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| headers | list[str] | column headers |
| rows | list[list[Any]] | data |
| name | str | objectName, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| add | bool | add to current context, defaults to True |

Simple Table output widget.

Example:

> *../examples/ui/widgets.py (Table)*

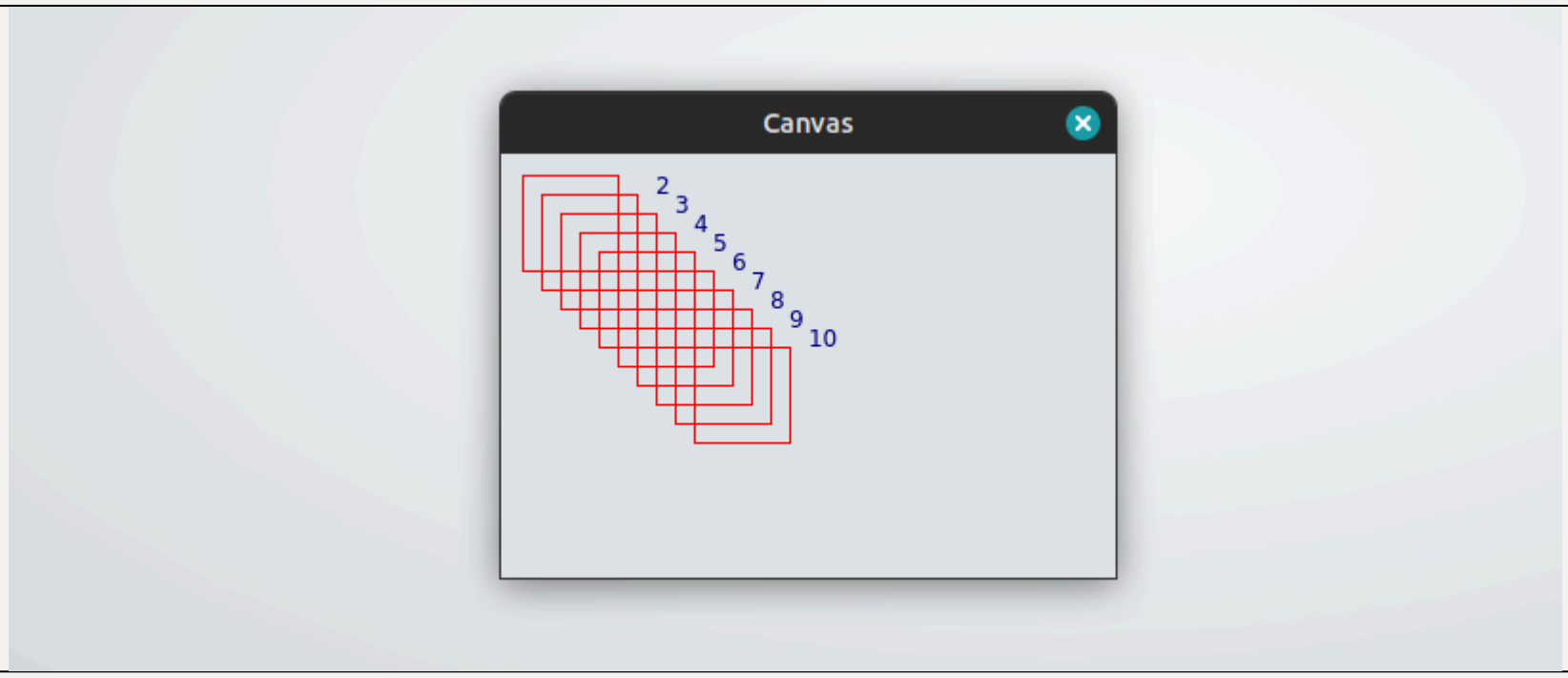
```
def demo_Table():
    with ui.Dialog("Table"):
        with ui.Col():
            ui.Table(
                headers=("Length", "^Width", ">Height"),
                rows=[
                    [21, 34, 56],
                    [65, 87, 98],
                    [21, 32, 54],
                    [65, 76, 87],
                ],
            )

```

## Widget: Canvas

### Signature / Canvas

```
def Canvas(
        paint: Callable[[QWidget, QPainter, QPaintEvent], None],
        *,
        setup: Callable[[QWidget, QPainter, QPaintEvent], None] | None=None,
        name: str | None=None,
        stretch: int=0,
        width: int=0,
        height: int=0,
        **
        kwargs: KwArgs
    ) -> CanvasWidget: ...
```

## Gui / Canvas




## Docs / Canvas

| RETURN TYPE | DESCRIPTION |
|---|---|
| CanvasWidget | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| paint | Callable[[QWidget, QPainter, QPaintEvent], None] | function to paint |
| setup | Callable[[QWidget, QPainter, QPaintEvent], None] | function to setup canvas |
| name | str | objectName, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| width | int | minimum width, defaults to 0 |
| height | int | minimum height, defaults to 0 |

Canvas Widget to allow custom painting.

Example:

> *../examples/ui/widgets.py (Canvas)*

```python
def demo_Canvas():

    def render(_widget: QWidget, qp: QPainter, ch: ui.CanvasHelper):
        for i in range(10):
            with ch.pen(color=Qt.red, width=1):
                qp.drawRect(QRect(i*10, i*10, 50, 50))
            with ch.pen(color=Qt.darkBlue, width=2):
                qp.drawText(i*10+60, i*10, str(i+1))

    with ui.Dialog("Canvas"):
        ui.Canvas(render, width=300, height=200)


```

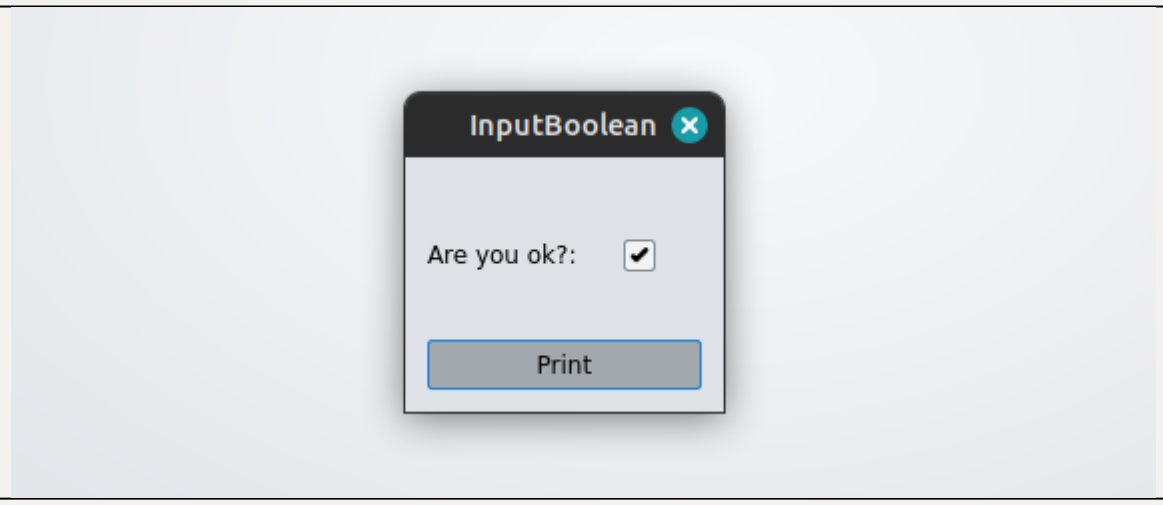
# Inputs

## Widget: InputBoolean

### Signature / InputBoolean

```python
def InputBoolean(
        value: bool=False,
        *,
        name: str | None=None,
        label: QWidget | str=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> QCheckBoxExt: ...
```

### Gui / InputBoolean



### Docs / InputBoolean

| RETURN TYPE | DESCRIPTION |
|---|---|
| QCheckBoxExt | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| value | bool | initial value, defaults to False |
| name | str | objectName, defaults to None |
| label | Union[QWidget, str] | ui label, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| add | bool | add to the gui, defaults to True |
| **kwargs | dict[str, Any] | settable QCheckBox properties |

Input boolean widget as QCheckBox.

Example:

> *../examples/ui/widgets.py (InputBoolean)*

```
def demo_InputBoolean():
    with ui.Dialog("InputBoolean"):
        val = ui.InputBoolean(label="Are you ok?:", alignment=Qt.AlignCenter)

        @ui.button("Print")
        def btn_print():
            ui.print_log(val.value())

```

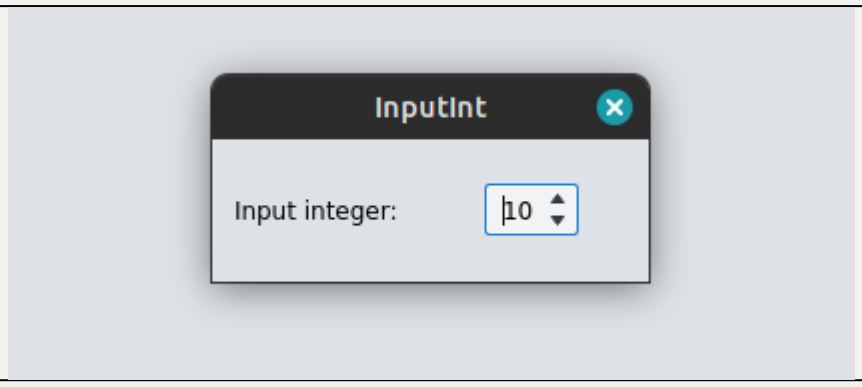
## Widget: InputInt

### Signature / InputInt

```
def InputInt(
        value: int=0,
        *,
        name: str | None=None,
        min: int=0,
        max: int | None=None,
        step: int=1,
        label: QWidget | str=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
```

```
        **
        kwargs: KwArgs
    ) -> QSpinBox: ...
```

### Gui / InputInt




### Docs / InputInt

| RETURN TYPE | DESCRIPTION |
|---|---|
| QSpinBox | The input widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| value | int | initial value, defaults to 0 |
| name | str | objectName, defaults to None |
| min | int | minimum accepted value, defaults to 0 |
| max | int | maximum accepted value, defaults to None |
| step | int | spin steps, defaults to 1 |
| label | Union[QWidget, str] | ui label, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Input int widget.

Example:

> *../examples/ui/widgets.py (InputInt)*

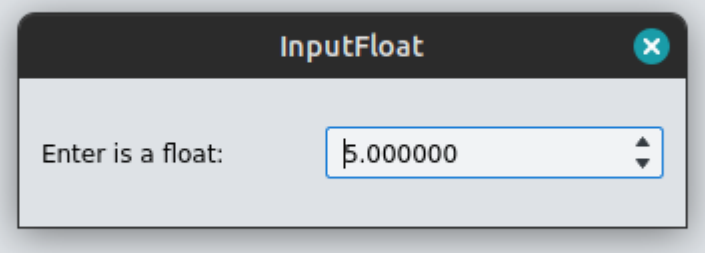
```python
def demo_InputInt():
    with ui.Dialog("InputInt"):
        val = ui.InputInt(value=5, label="Enter is an int:", alignment=Qt.AlignCenter)

        @ui.button("Print")
        def print_int():
            ui.print_log(val.value())

```

# Widget: InputFloat

### Signature / InputFloat

```python
def InputFloat(
        value: float=0.0,
        *,
        name: str | None=None,
        min: float=0.0,
        max: float=sys.float_info.max,
        decimals: int=6,
        step: float=0.01,
        label: QWidget | str | None=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> QDoubleSpinBox: ...
```

### Gui / InputFloat



### Docs / InputFloat

| RETURN TYPE | DESCRIPTION |
|---|---|
| QDoubleSpinBox | The input widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| value | float | initial value, defaults to 0.0 |
| name | str | objectName, defaults to None |
| min | float | minimum accepted value, defaults to 0.0 |
| max | float | maximum accepted value, defaults to sys.float_info.max |
| decimals | int | decimal digits, defaults to 6 |
| step | float | spin steps, defaults to 0.01 |
| label | Union[QWidget, str] | ui label, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Input float widget.

Example:

> *../examples/ui/widgets.py (InputFloat)*

```
def demo_InputFloat():
    with ui.Dialog("InputFloat"):
        val = ui.InputFloat(value=5.0, label="Enter is a float:", alignment=Qt.AlignCenter)

        @ui.button("Print")
        def print_float():
            ui.print_log(val.value())

```

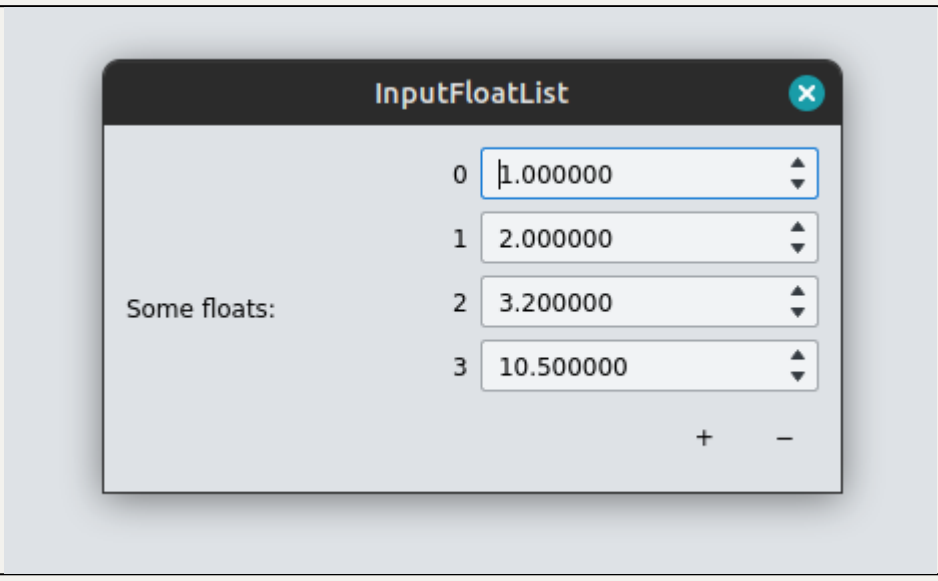
## Widget: InputFloatList

### Signature / InputFloatList

```
def InputFloatList(
        values: list[float] | None=None,
        label: QWidget | str | None=None,
        *,
        name: str | None=None,
        label_fn: Callable[[int], str] | None=None,
```

```
        count: int=0,
        resizable: bool=False,
        min_count: int=0,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> InputFloatListWidget: ...
```

### Gui / InputFloatList




### Docs / InputFloatList

| RETURN TYPE | DESCRIPTION |
|---|---|
| InputFloatListWidget | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| values | list[float] | initial values, defaults to None |
| label | Union[QWidget, str] | ui label, defaults to None |
| name | str | objectName, defaults to None |
| label_fn | Callable[[int], str] | label provider (for custom row labels), defaults to None |
| count | int | number of autogenerated rows, defaults to 0 |
| resizable | bool | allow resizing the list, defaults to False |
| min_count | int | minimum number of rows, defaults to 0 |
| add | bool | add to the gui, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Widget to accept lists of float numbers.

Example:

> *../examples/ui/widgets.py (InputFloatList)*

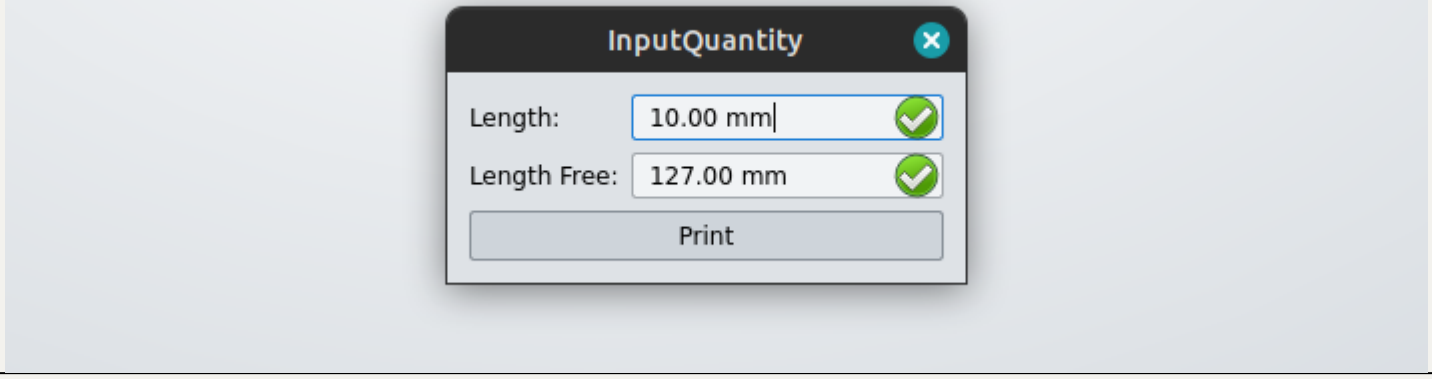
```python
def demo_InputFloatList():
    with ui.Dialog("InputFloatList"):
        with ui.Col():
            val = ui.InputFloatList(
                values=[1.0, 2.0, 3.2, 10.5],
                label="Some floats:",
                resizable=True,
            )

            @ui.button("Print")
            def btn_print():
                ui.print_log(val.value())

```

## Widget: InputQuantity

### Signature / InputQuantity

```python
def InputQuantity(
        value: Numeric=None,
        *,
        name: str | None=None,
        min: Numeric=None,
        max: Numeric=None,
        step: Numeric=1.0,
        label: QWidget | str=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        unit: str | None=None,
        obj: DocumentObject | None=None,
        property: str | None=None,
        auto_apply: bool=True,
        add: bool=True,
        widget_type: str='Gui::InputField',
        accessor_adapter: PropertyAccessorAdapter | None=None,
        **
        kwargs: KwArgs
    ) -> InputQuantityWidget: ...
```

## Gui / InputQuantity




## Docs / InputQuantity

| RETURN TYPE | DESCRIPTION |
|---|---|
| InputQuantityWidget | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| value | Numeric | Initial value, defaults to None |
| name | str | objectName (Qt), defaults to None |
| min | Numeric | Minimum accepted value, defaults to None |
| max | Numeric | Maximum accepted value, defaults to None |
| step | Numeric | Spin step, defaults to 1.0 |
| label | Union[QWidget, str] | gui label, defaults to None |
| stretch | int | Layout stretch, defaults to 0 |
| alignment | Qt.Alignment | Layout alignment, defaults to Qt.Alignment() |
| unit | str | quantity unit (i.e. mm, in, ...), defaults to None |
| obj | DocumentObject | Object to bind the expression engine, defaults to None |
| property | str | Property name of the bounded DocumentObject if any, defaults to None |
| add | bool | Add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Input Quantity Widget with unit and expressions support.

Example:

> *../examples/ui/widgets.py (InputQuantity)*

```
def demo_InputQuantity():
    App.activeDocument().addObject("Part::Box", "Box1")
    with ui.Dialog("InputQuantity"):
        quantity = ui.InputQuantity(
            label="Length:",
```

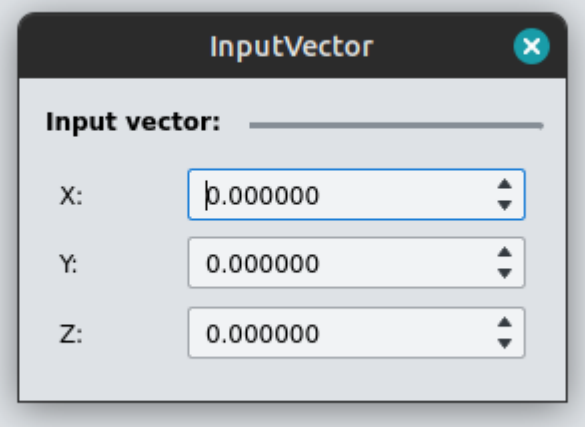
```
            obj=App.activeDocument().getObject("Box1"),
            property="Length",
            unit="in",
            alignment=Qt.AlignCenter,
        )

        quantity2 = ui.InputQuantity(
            value=5.0, label="Length Free:", unit="in", alignment=Qt.AlignCenter,
        )

        @ui.button("Print")
        def print_q():
            ui.print_log(quantity.value(), quantity2.value())

```

## Widget: InputVector

### Signature / InputVector

```
def InputVector(
        label: QWidget | str | None=None,
        value: tuple | Vector=(0.0, 0.0, 0.0)
    ) -> InputVectorWrapper: ...
```

### Gui / InputVector




### Docs / InputVector

| RETURN TYPE | DESCRIPTION |
| --- | --- |
| InputVectorWrapper | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| label | Union[QWidget, str] | ui label, defaults to None |
| value | Union[tuple, Vector] | vector value, defaults to (0.0, 0.0, 0.0) |

Widget to accept a vector.

Example:

> *../examples/ui/widgets.py (InputVector)*

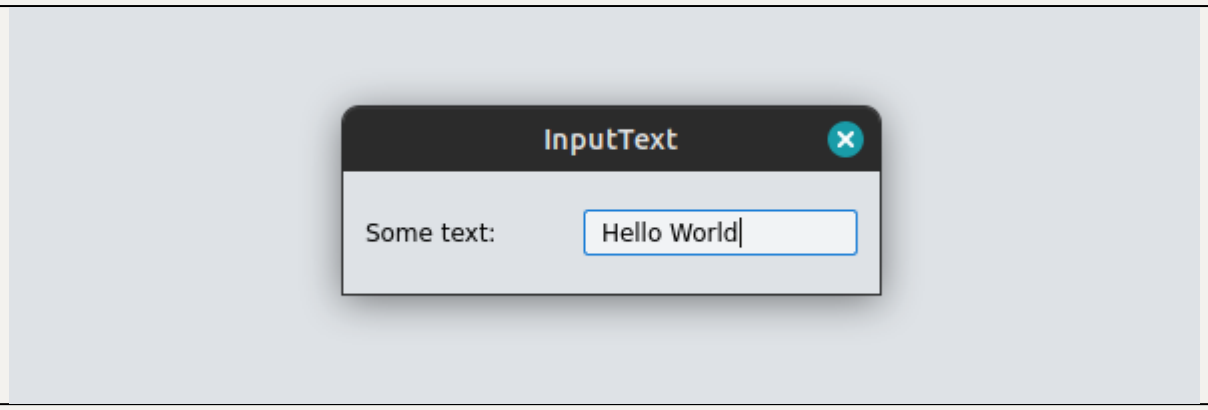
```
def demo_InputVector():
    with ui.Dialog("InputVector"):
        with ui.Col():
            val = ui.InputVector(label="Input vector:")

            @ui.button("Print")
            def btn_print():
                ui.print_log(val.value())

```

## Widget: InputText

### Signature / InputText

```
def InputText(
        value: str='',
        *,
        name: str | None=None,
        label: QWidget | str=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> InputTextWidget: ...
```

## Gui / InputText




## Docs / InputText

| RETURN TYPE | DESCRIPTION |
|---|---|
| InputTextWidget | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| value | str | initial value, defaults to "" |
| name | str | objectName, defaults to None |
| label | Union[QWidget, str] | gui label, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| add | bool | add to current context, defaults to True |
| **kwargs | dict[str, Any] | Qt properties |

Input text widget.

Example:

> *../examples/ui/widgets.py (InputText)*

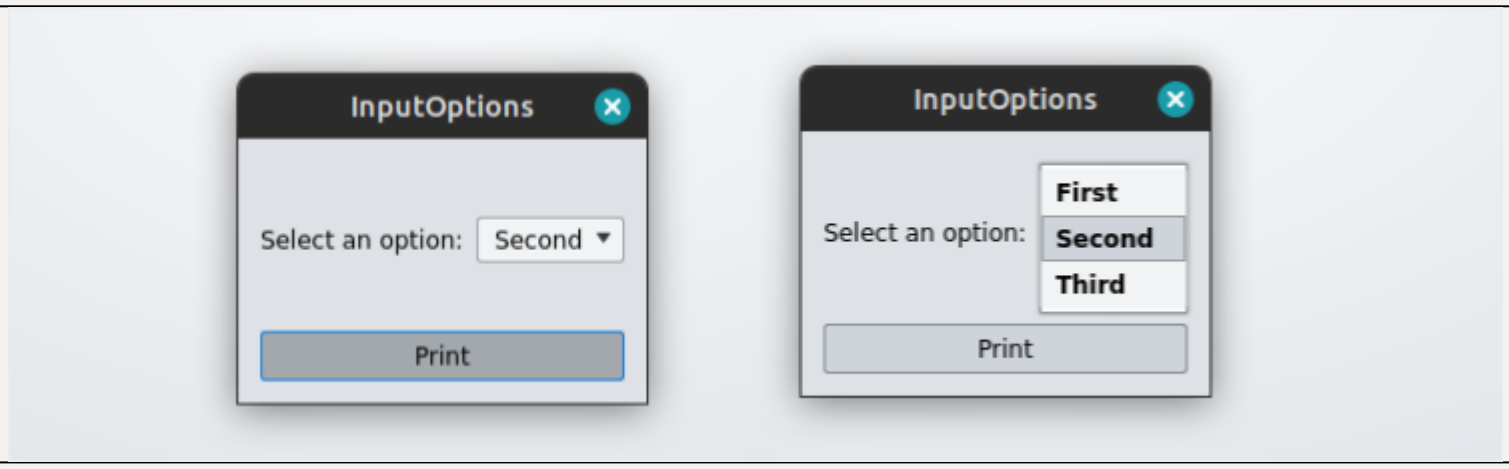
```
def demo_InputText():
    with ui.Dialog("InputText"):
        text = ui.InputText(value="Hello World", label="Some text:", alignment=Qt.AlignCenter)

        @ui.button("Print")
        def print_text():
            ui.print_log(text.value())

```

# Widget: InputOptions

## Signature / InputOptions

```
def InputOptions(
        options: dict[str, Hashable],
        value: Hashable=None,
        label: str | None=None,
        *,
        name: str | None=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
        select_on_focus: bool=False,
        **
        kwargs: KwArgs
    ) -> InputOptionsWidget: ...
```

## Gui / InputOptions




## Docs / InputOptions

| RETURN TYPE | DESCRIPTION |
|---|---|
| InputOptionsWidget | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| options | dict[str, Hashable] | label to value mapping |
| value | Hashable | initial value, defaults to None |
| label | str | gui label, defaults to None |
| name | str | objectName, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| **kwargs | dict[str, Any] | Qt properties |

ComboBox widget.

Example:

> *../examples/ui/widgets.py (InputOptions)*

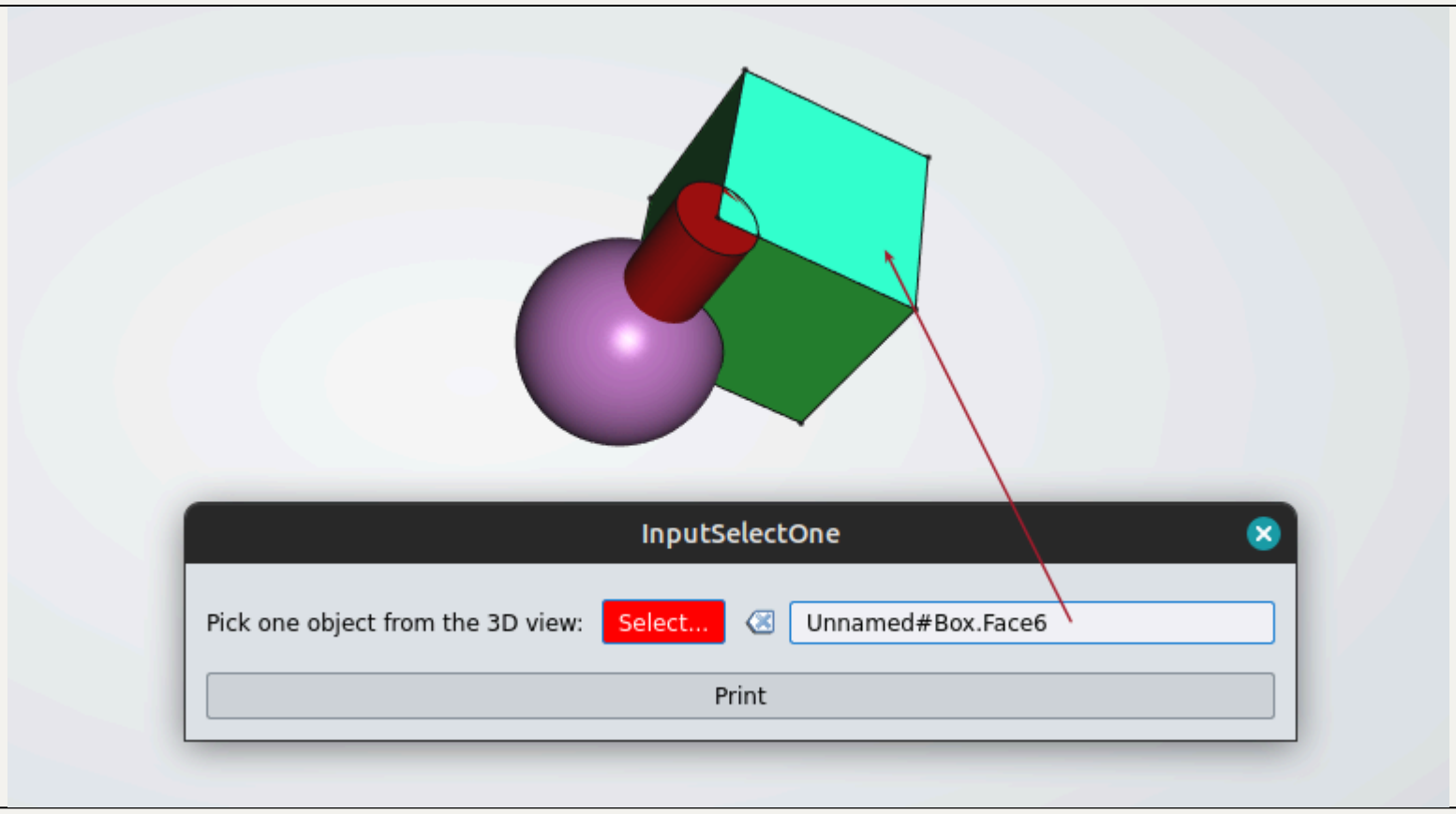
```
def demo_InputOptions():
    with ui.Dialog("InputOptions"):
        val = ui.InputOptions(
            options={
                "First": 10,
                "Second": 20,
                "Third": 30,
            },
            value=20,
            label="Select an option:",
            alignment=Qt.AlignCenter,
        )

        @ui.button("Print")
        def print_opt():
            ui.print_log(val.value())

```

## Widget: InputSelectOne

### Signature / InputSelectOne

```
def InputSelectOne(
        label: str | None=None,
        *,
        name: str | None=None,
        active: bool=False,
        auto_deactivate: bool=True
    ) -> None: ...
```

## Gui / InputSelectOne




## Docs / InputSelectOne

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| label | str | gui label, defaults to None |
| name | str | objectName, defaults to None |
| active | bool | activated by default, defaults to False |
| auto_deactivate | bool | *description*, defaults to True |

3D Selection widget. Allows to pick an object.

Example:

> *../examples/ui/widgets.py (InputSelectOne)*

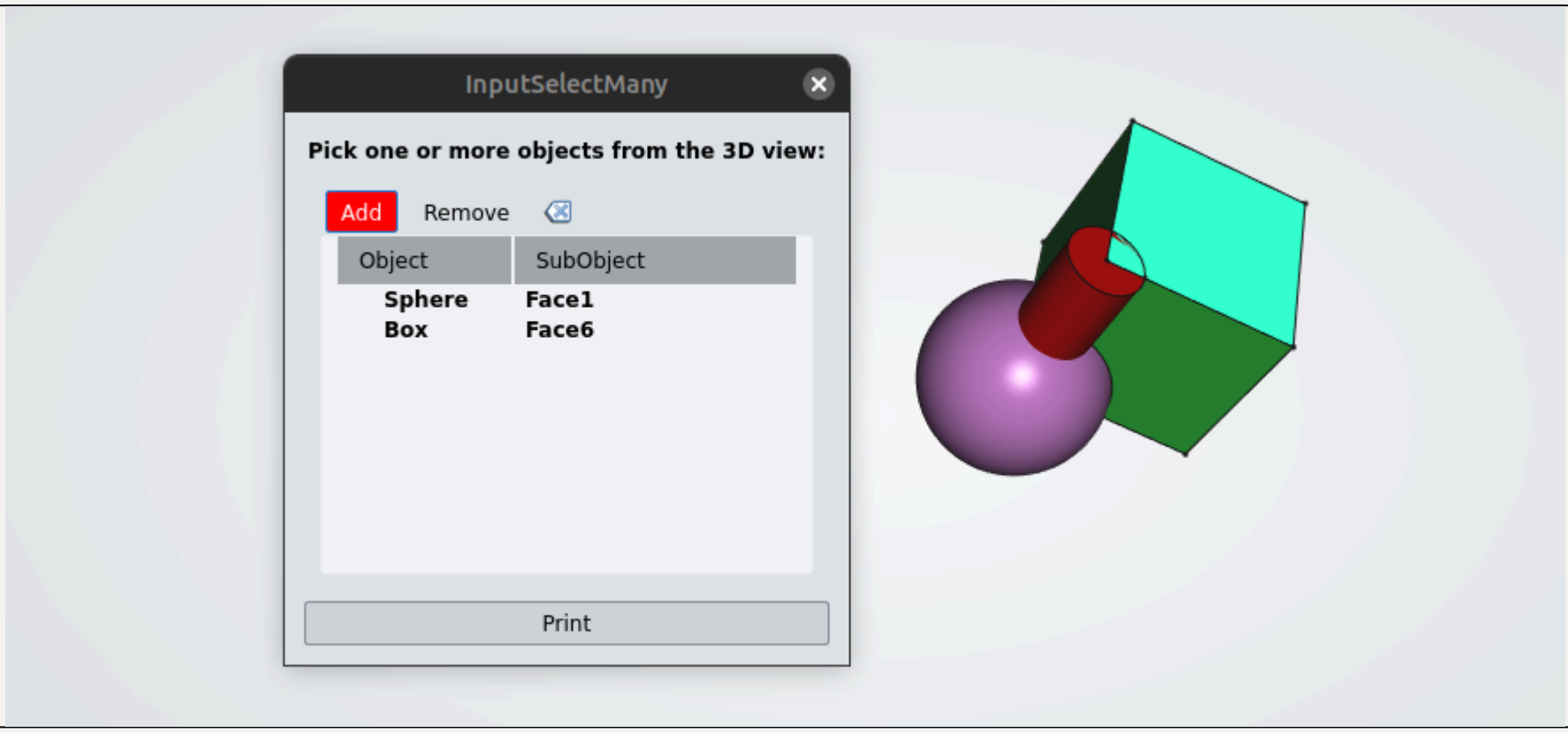
```
def demo_InputSelectOne():
    with ui.Dialog("InputSelectOne", modal=False):
        val = ui.InputSelectOne(label="Pick one object from the 3D view:")

        @ui.button("Print")
        def print_sel():
            ui.print_log(val.value())

```

# Widget: InputSelectMany

## Signature / InputSelectMany

```
def InputSelectMany(
        label: str | None=None,
        *,
        name: str | None=None,
        active: bool=False
    ) -> None: ...
```

## Gui / InputSelectMany




## Docs / InputSelectMany

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| label | str | gui label, defaults to None |
| name | str | objectName, defaults to None |
| active | bool | active by default or not, defaults to False |

3D Multi-Selection Widget.

Example:

> *../examples/ui/widgets.py (InputSelectMany)*

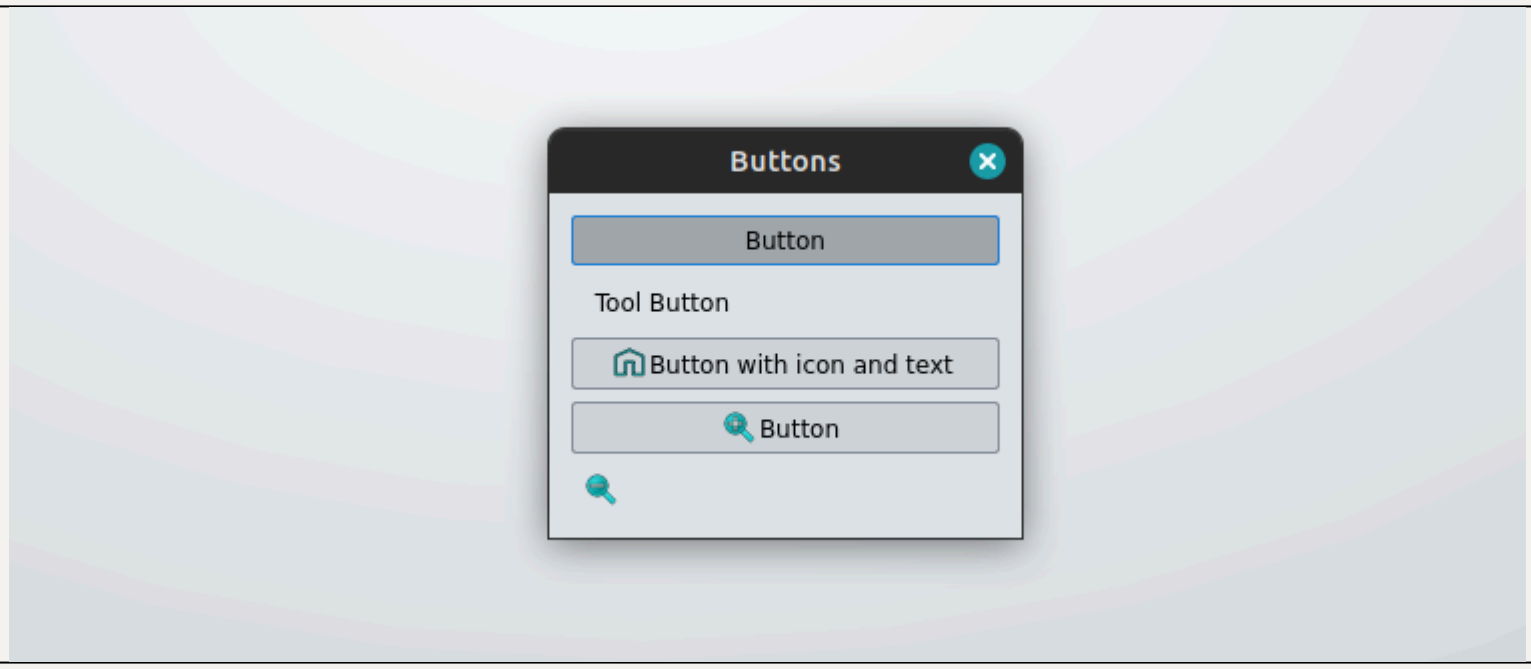
```python
def demo_InputSelectMany():
    with ui.Dialog("InputSelectMany", modal=False):
        val = ui.InputSelectMany(label="Pick one or more objects from the 3D view:")

        @ui.button("Print")
        def print_sel():
            ui.print_log(val.value())

```

## Widget: button

### Signature / button

```python
def button(
        label: str | None=None,
        *,
        tool: bool=False,
        icon: QIcon | str | None=None,
        stretch: int=0,
        alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
        add: bool=True,
        **
        kwargs: KwArgs
    ) -> Callable[[Callable[[], None]], QAbstractButton]: ...
```

### Gui / button



## Docs / button

| RETURN TYPE | DESCRIPTION |
|---|---|
| QAbstractButton | The widget |

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| label | str | text of the button, defaults to None |
| tool | bool | use tool style button, defaults to False |
| icon | Union[QIcon, str] | icon, defaults to None |
| stretch | int | layout stretch, defaults to 0 |
| alignment | Qt.Alignment | layout alignment, defaults to Qt.Alignment() |
| add | bool | add to current context, defaults to True |

Button Widget.

Example:

> *../examples/ui/widgets.py (buttons)*

```python
def demo_buttons():
    with ui.Dialog("Buttons", modal=False):

        @ui.button("Button")
        def btn1():
            ui.print_log("Hello btn1")

        @ui.button("Tool Button", tool=True)
        def btn2():
            ui.print_log("Hello btn2")

        @ui.button("Button with icon and text", icon=QIcon(":icons/Std_ViewHome.svg"))
        def btn3():
            ui.print_log("Hello btn3")

        @ui.button(icon=QIcon(":icons/zoom-in.svg"))
        def btn4():
            ui.print_log("Hello btn4")

        @ui.button(icon=QIcon(":icons/zoom-out.svg"), tool=True)
        def btn5():
            ui.print_log("Hello btn4")

```

# Layout tools

## Function: Stretch

### Signature / Stretch

```
def Stretch(stretch: int=0) -> None: ...
```

### Docs / Stretch

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| stretch | int | 0-100 stretch factor, defaults to 0 |

Add stretch factor to the current layout.

## Function: Spacing

### Signature / Spacing

```
def Spacing(size: int) -> None: ...
```

### Docs / Spacing

| ARGUMENT | TYPE | DESCRIPTION |
|---|---|---|
| size | int | spacing |

Adds spacing ro the current layout.