

FreeCAD GUI Modern API

META	VALUE
generated	2025-04-22 08:45:55.432739
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

- [FreeCAD GUI Modern API](#)
- [Preliminaries](#)
 - [Disclaimer](#)
 - [Audience](#)
 - [Goals](#)
 - [Non Goals](#)
- [Features](#)
- [Widgets](#)
 - [Containers / Layouts](#)
 - [Widget: Dialog](#)
 - [Widget: Scroll](#)
 - [Widget: GroupBox](#)
 - [Widget: Container](#)
 - [Widget: TabContainer](#)
 - [Function: Tab](#)
 - [Widget: Splitter](#)
 - [Widget: Col](#)
 - [Widget: Row](#)
 - [Output](#)

- Widget: TextLabel
 - Widget: Html
 - Widget: ImageView
 - Widget: SvgImageView
 - Widget: Table
 - Widget: Canvas
- Inputs
 - Widget: InputBoolean
 - Widget: InputInt
 - Widget: InputFloat
 - Widget: InputFloatList
 - Widget: InputQuantity
 - Widget: InputVector
 - Widget: InputText
 - Widget: InputOptions
 - Widget: InputSelectOne
 - Widget: InputSelectMany
 - Widget: button
- Layout tools
 - Function: Stretch
 - Function: Spacing

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

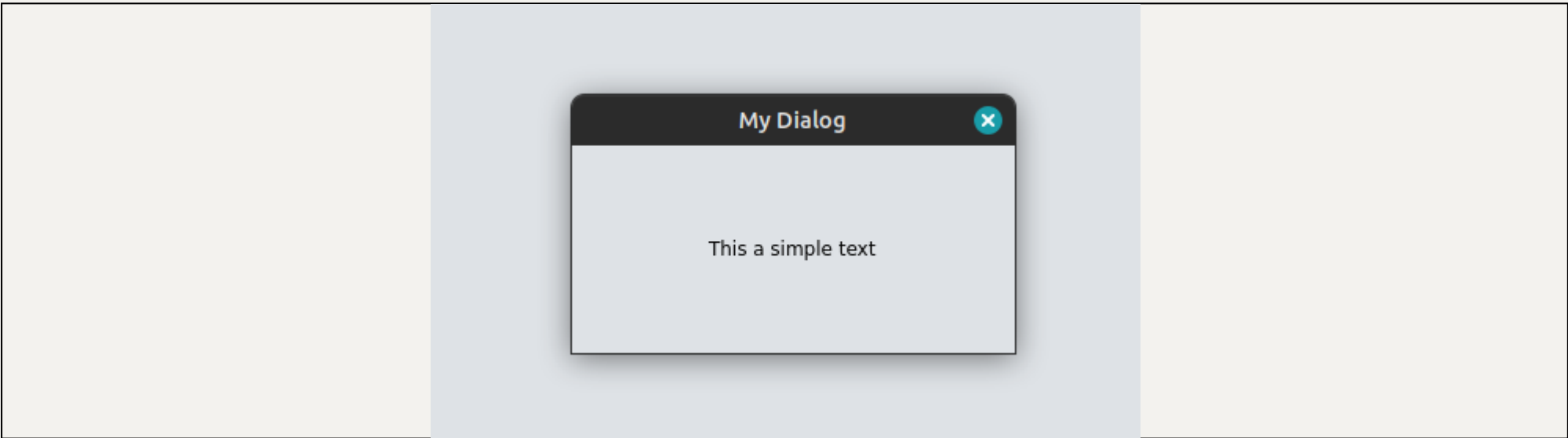
Containers / Layouts

Widget: Dialog

Signature / Dialog

```
1 def Dialog(  
2     title: str | None=None,  
3     *,  
4     size: tuple[int, int] | None=None,  
5     show: bool=True,  
6     modal: bool=True,  
7     parent: QWidget=None,  
8     **  
9     kwargs: KwArgs  
10 ) -> Generator[QDialog, 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)
```

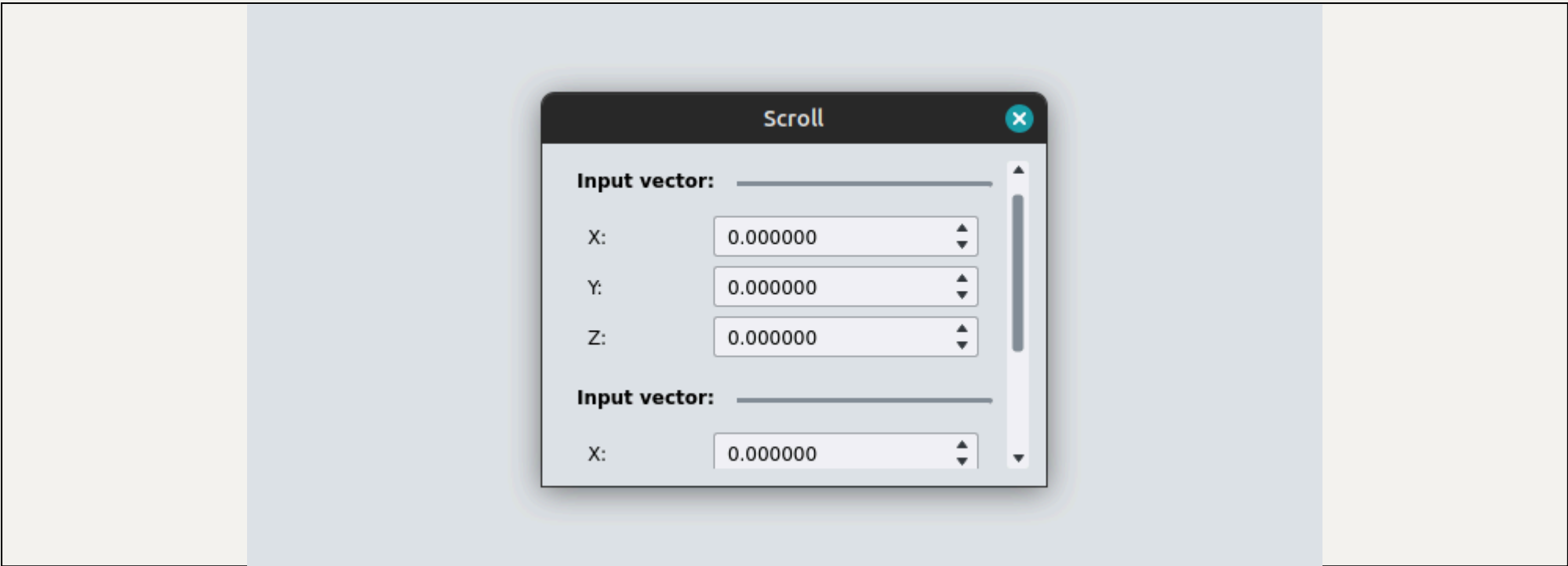
```
1 def demo_Dialog():
2     with ui.Dialog("Dialog"):
3         with ui.Col():
4             ui.TextLabel("This a simple text", alignment=Qt.AlignCenter)
5
```

Widget: Scroll

Signature / Scroll

```
1 def Scroll(
2     *,
3     add: bool=True,
4     **
5     kwargs: KwArgs
6 ) -> 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*)

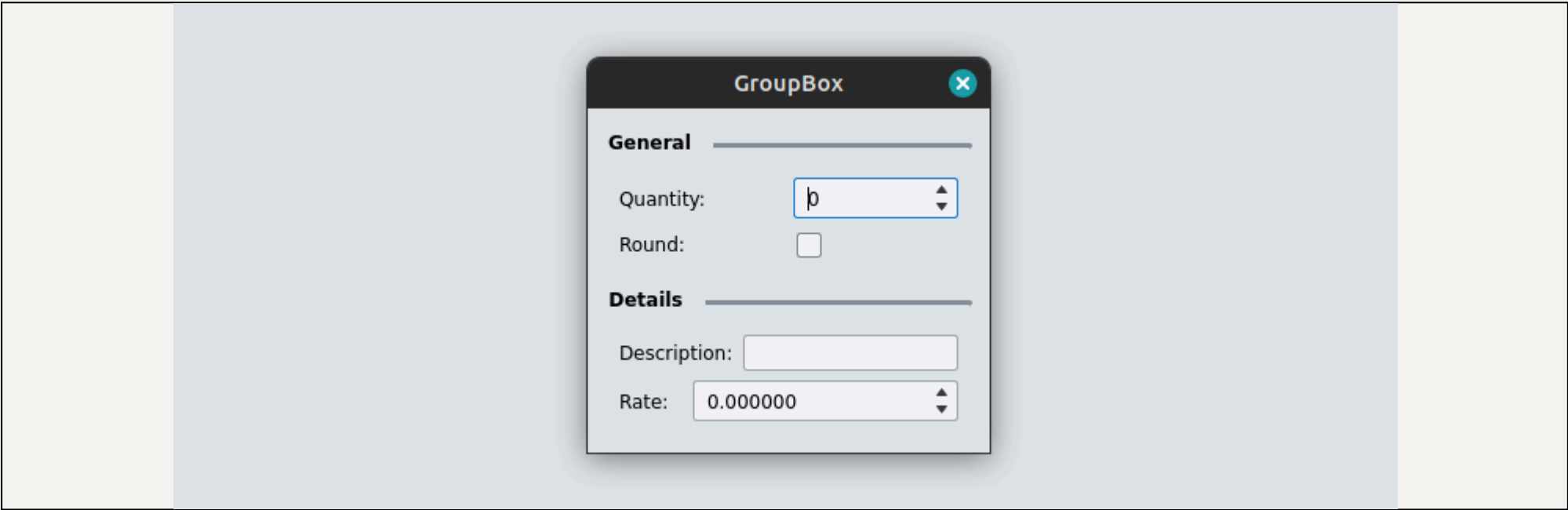
```
1 def demo_Scroll():
2     with ui.Dialog("Scroll"):
3         with ui.Scroll(widgetResizable=True):
4             with ui.Container():
5                 _v1 = ui.InputVector(label="Input vector:")
6                 _v2 = ui.InputVector(label="Input vector:")
7
```

Widget: GroupBox

Signature / GroupBox

```
1 def GroupBox(  
2     title: str | None=None,  
3     *,  
4     add: bool=True,  
5     **  
6     kwargs: KwArgs  
7 ) -> 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)
```



```
1 def demo_GroupBox():
2     with ui.Dialog("GroupBox"):
3         with ui.GroupBox("General"):
4             _quant = ui.InputInt(label="Quantity:")
5             _round = ui.InputBoolean(label="Round:")
6         with ui.GroupBox("Details"):
7             _desc = ui.InputText(label="Description:")
8             _rate = ui.InputFloat(label="Rate:")
9
```

Widget: Container

Signature / Container

```
1 def Container(
2     *,
3     add: bool=True,
4     top: bool=False,
5     **
6     kwargs: KwArgs
7 ) -> Generator[QFrame, 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)

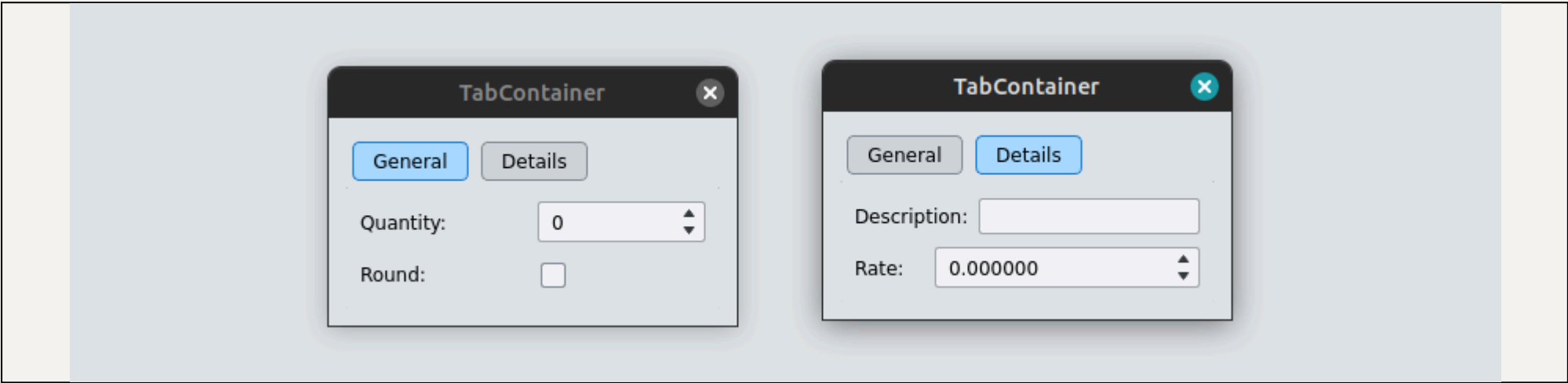
```
1 def demo_Scroll():
2     with ui.Dialog("Scroll"):
3         with ui.Scroll(widgetResizable=True):
4             with ui.Container():
5                 _v1 = ui.InputVector(label="Input vector:")
6                 _v2 = ui.InputVector(label="Input vector:")
7
```

Widget: TabContainer

Signature / TabContainer

```
1 def TabContainer(  
2     *,  
3     stretch: int=0,  
4     add: bool=True,  
5     **  
6     kwargs: KwArgs  
7 ) -> 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:

```
1 | ../examples/ui/widgets.py (TabContainer)
```

```
1 def demo_TabContainer():
2     with ui.Dialog("TabContainer"):
3         with ui.TabContainer():
4             with ui.Tab("General"):
5                 _quant = ui.InputInt(label="Quantity:")
6                 _round = ui.InputBoolean(label="Round:")
7             with ui.Tab("Details"):
8                 _desc = ui.InputText(label="Description:")
9                 _rate = ui.InputFloat(label="Rate:")
10
```

Function: Tab

Signature / Tab

```
1 def Tab(
2     title: str,
3     *,
4     icon: QIcon=None,
5     add: bool=True,
6     **
7     kwargs: KwArgs
8 ) -> 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)
```

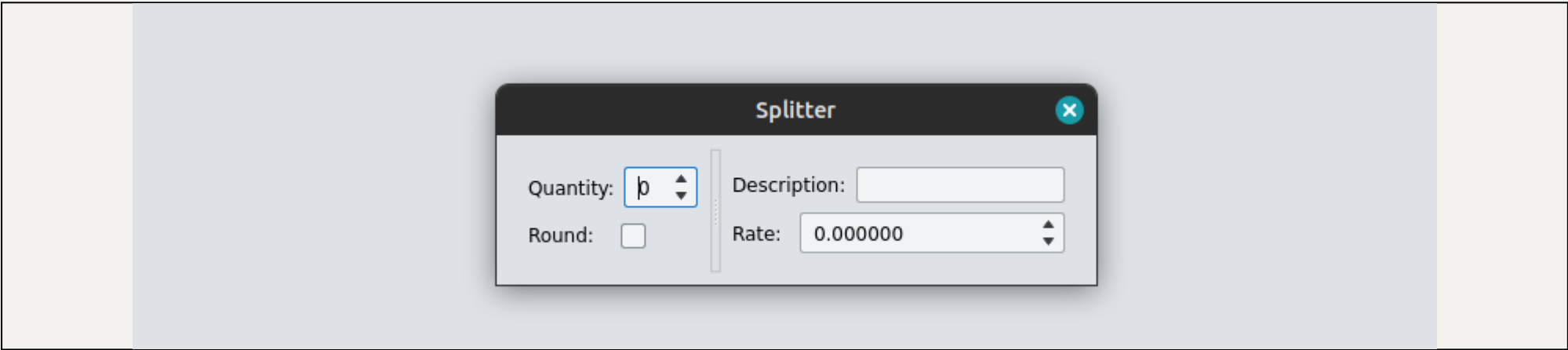
```
1 def demo_TabContainer():
2     with ui.Dialog("TabContainer"):
3         with ui.TabContainer():
4             with ui.Tab("General"):
5                 _quant = ui.InputInt(label="Quantity:")
6                 _round = ui.InputBoolean(label="Round:")
7             with ui.Tab("Details"):
8                 _desc = ui.InputText(label="Description:")
9                 _rate = ui.InputFloat(label="Rate:")
10
```

Widget: Splitter

Signature / Splitter

```
1 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)
```

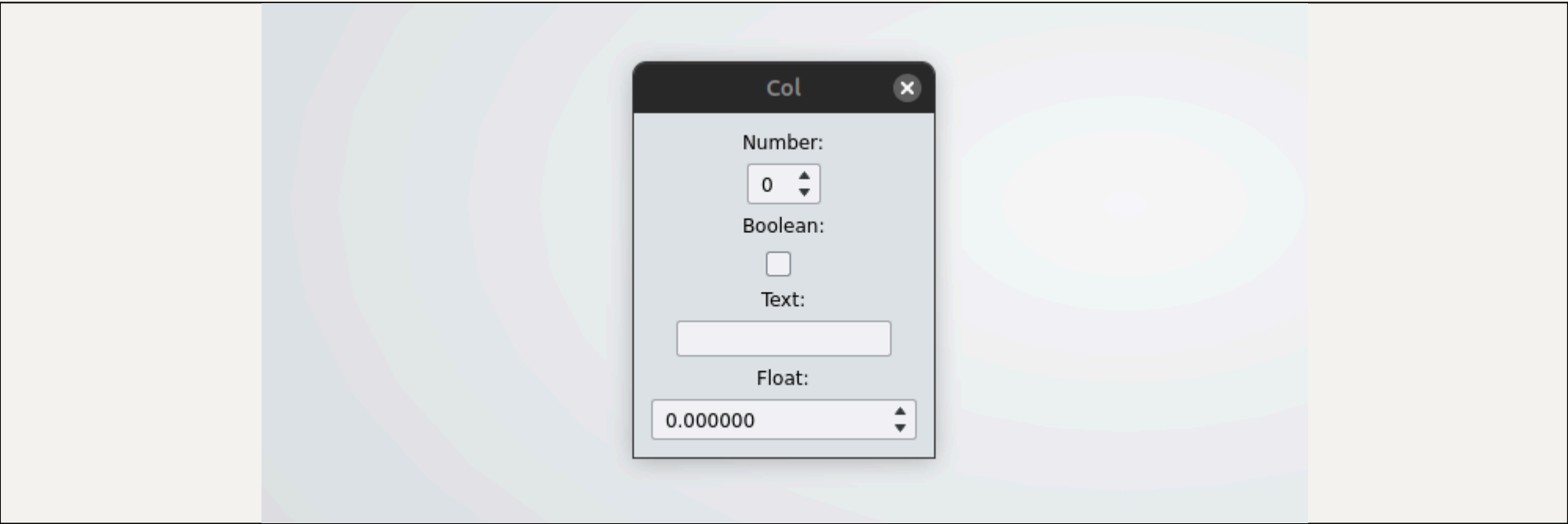
```
1 def demo_Splitter():
2     with ui.Dialog("Splitter"):
3         with ui.Splitter():
4             with ui.Container():
5                 _quant = ui.InputInt(label="Quantity:")
6                 _round = ui.InputBoolean(label="Round:")
7             with ui.Container():
8                 _desc = ui.InputText(label="Description:")
9                 _rate = ui.InputFloat(label="Rate:")
10
```

Widget: Col

Signature / Col

```
1 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)`

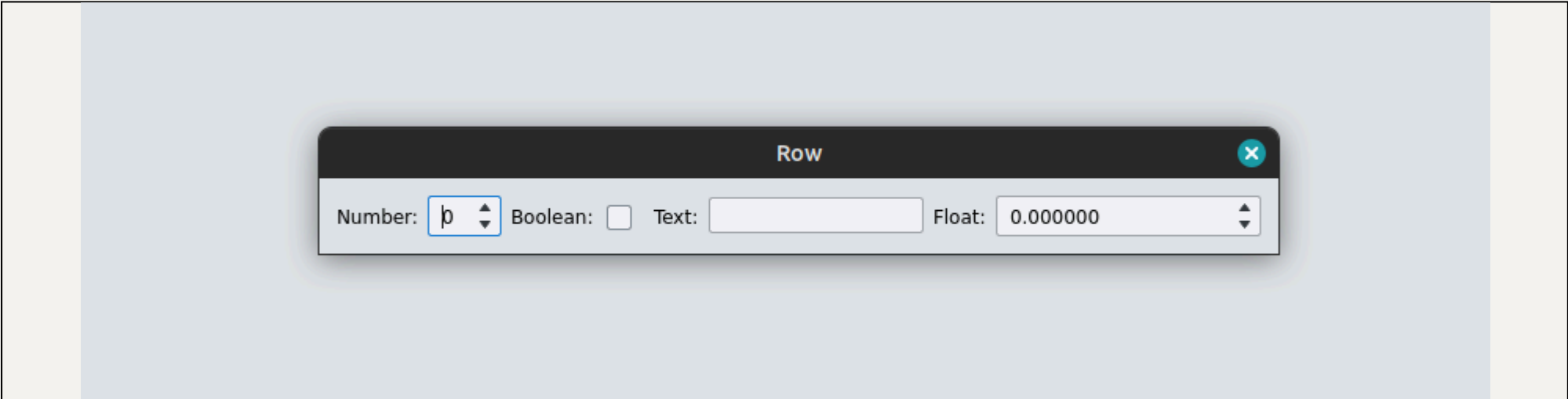
```
1 def demo_Col():
2     with ui.Dialog("Col"):
3         with ui.Col():
4             ui.TextLabel("Number:", alignment=Qt.AlignCenter)
5             ui.InputInt(alignment=Qt.AlignCenter)
6             ui.TextLabel("Boolean:", alignment=Qt.AlignCenter)
7             ui.InputBoolean(alignment=Qt.AlignCenter)
8             ui.TextLabel("Text:", alignment=Qt.AlignCenter)
9             ui.InputText(alignment=Qt.AlignCenter)
10            ui.TextLabel("Float:", alignment=Qt.AlignCenter)
11            ui.InputFloat(alignment=Qt.AlignCenter)
12
```

Widget: Row

Signature / Row

```
1 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

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

Horizontal context/layout

Example:

`../examples/ui/widgets.py (Row)`

```
1 def demo_Row():
2     with ui.Dialog("Row"):
3         with ui.Row():
4             ui.TextLabel("Number:")
5             ui.InputInt()
6             ui.TextLabel("Boolean:")
7             ui.InputBoolean()
8             ui.TextLabel("Text:")
9             ui.InputText()
10            ui.TextLabel("Float:")
11            ui.InputFloat()
12
```

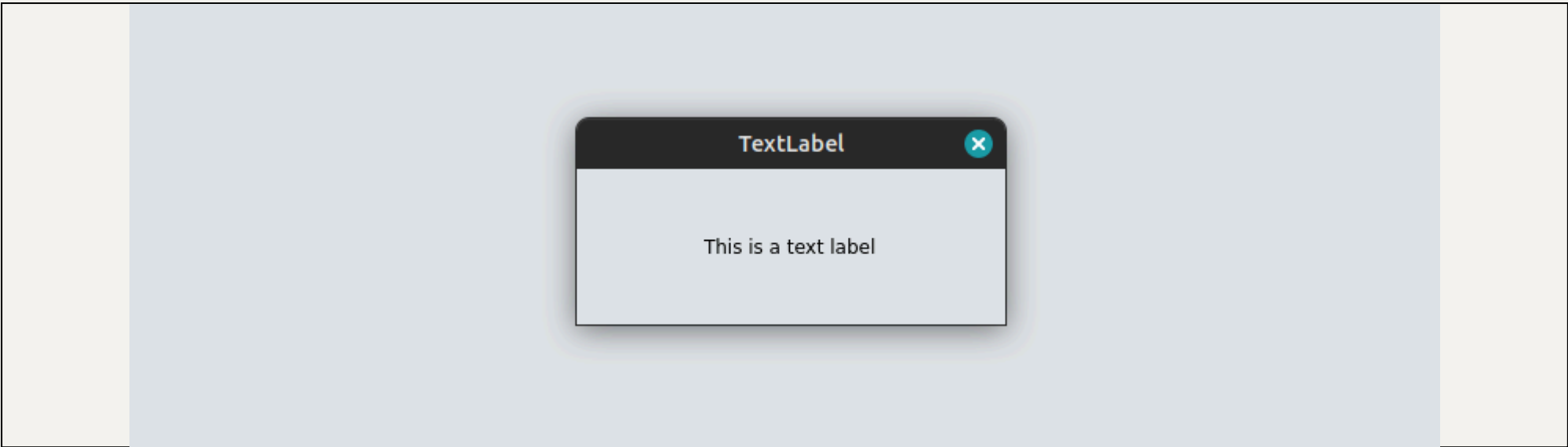
Output

Widget: TextLabel

Signature / TextLabel

```
1 def TextLabel(
2     text: str='',
3     *,
4     stretch: int=0,
5     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
6     add: bool=True,
7     **
8     kwargs: KwArgs
9 ) -> 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)
```

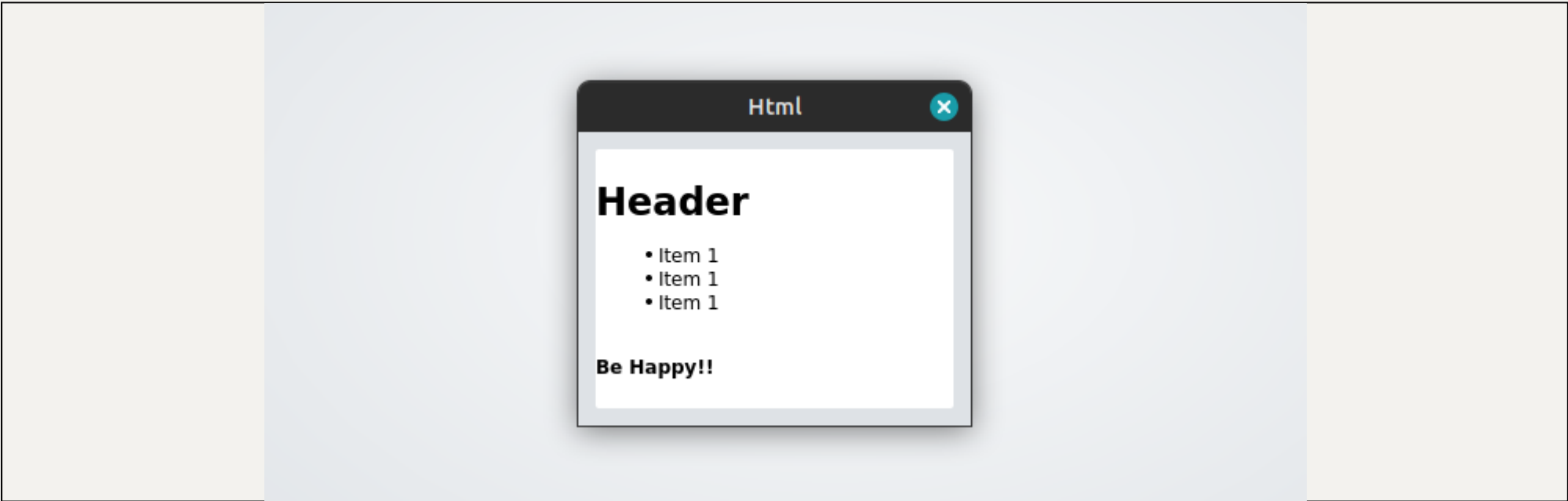
```
1 def demo_TextLabel():
2     with ui.Dialog("TextLabel"):
3         with ui.Col():
4             ui.TextLabel("This is a text label", alignment=Qt.AlignCenter)
5
```


Widget: Html

Signature / Html

```
1 def Html(  
2     *,  
3     html: str | None=None,  
4     file: str | None=None,  
5     css: str | None=None,  
6     css_file: str | None=None,  
7     base_path: str | None=None,  
8     background: str | None=None,  
9     stretch: int=0,  
10    alignment: Qt.Alignment=DEFAULT_ALIGNMENT,  
11    variables: dict[str, Any] | None=None,  
12    add: bool=True,  
13    **  
14    kwargs: KwArgs  
15 ) -> 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*)

```
1 def demo_Html():
2     with ui.Dialog("Html"):
3         with ui.Col():
4             ui.Html(
5                 background="#ffffff",
6                 html="""
7                 <h1>Header</h1>
8                 <ul>
9                     <li>Item 1</li>
10                    <li>Item 1</li>
11                    <li>Item 1</li>
12                </ul>
13                <br />
14                <p>
15                    <strong>Be Happy!!</strong>
16                </p>
17                """,
18             )
19
```

Widget: ImageView

Signature / ImageView

```
1 def ImageView(
2     uri: str,
3     *,
```

```
4         label: str | None=None,
5         name: str | None=None,
6         background: str | QColor | None=None,
7         add: bool=True,
8         **
9         kwargs: KwArgs
10    ) -> ImageViewWidget: ...
```

Gui / ImageView



Docs / ImageView

RETURN TYPE		DESCRIPTION
ImageViewWidget		<i>description</i>

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	<i>description</i> , defaults to True

Image render widget.

Example:

```
 | ../examples/ui/widgets.py (ImageView)
```

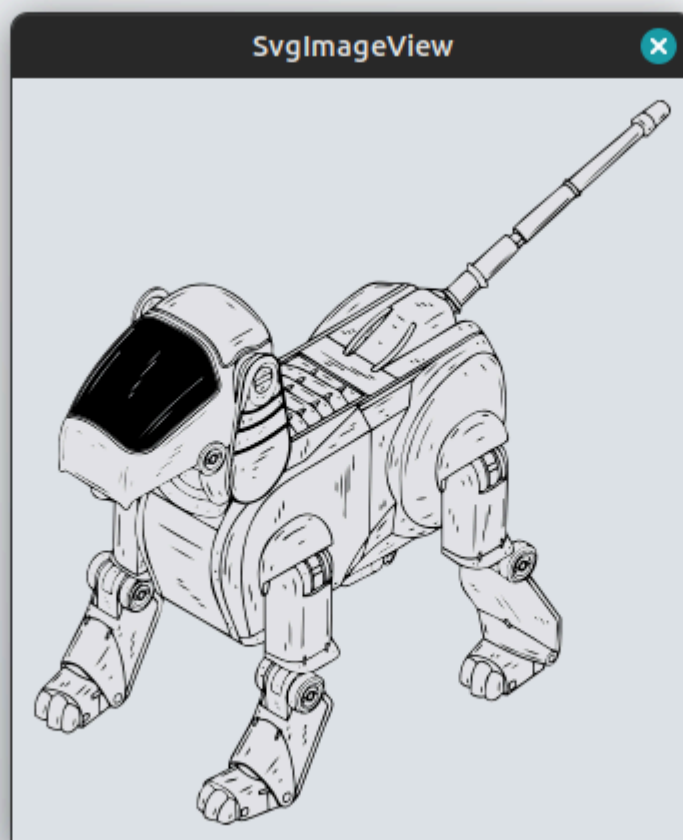
```
1 def demo_ImageView():
2     with ui.Dialog("ImageView", size=(400, 300)):
3         with ui.Col():
4             ui.ImageView(
5                 str(Path(__file__).parent / "image.png"),
6             )
7
```

Widget: SvgImageView

Signature / SvgImageView

```
1 def SvgImageView(
2     uri: str,
3     label: str | None=None,
4     *,
5     name: str | None=None,
6     stretch: int=0,
7     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
8     **
9     kwargs: KwArgs
10 ) -> 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)
```

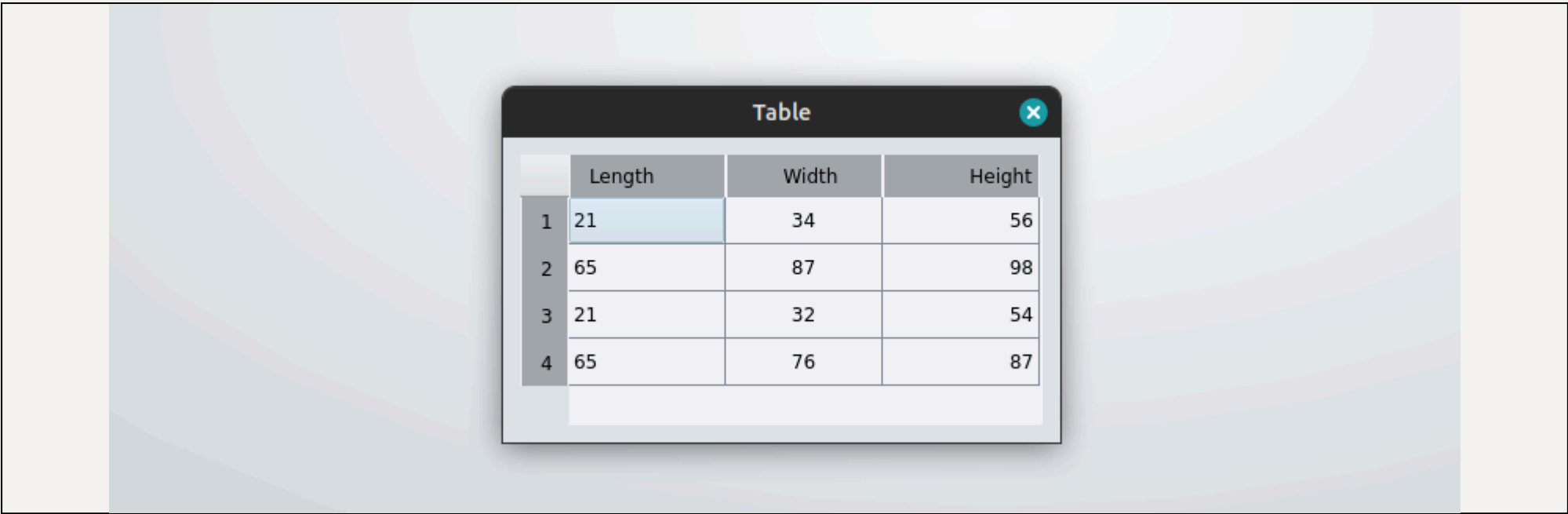
```
1 def demo_SvgImageView():
2     with ui.Dialog("SvgImageView"):
3         with ui.Col():
4             ui.SvgImageView(
5                 str(Path(__file__).parent / "vector.svg"),
6             )
7
```

Widget: Table

Signature / Table

```
1 def Table(
2     headers: list[str],
3     rows: list[list[Any]],
4     *,
5     name: str | None=None,
6     stretch: int=0,
7     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
8     add: bool=True,
9     **
10     kwargs: KwArgs
11 ) -> 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)
```

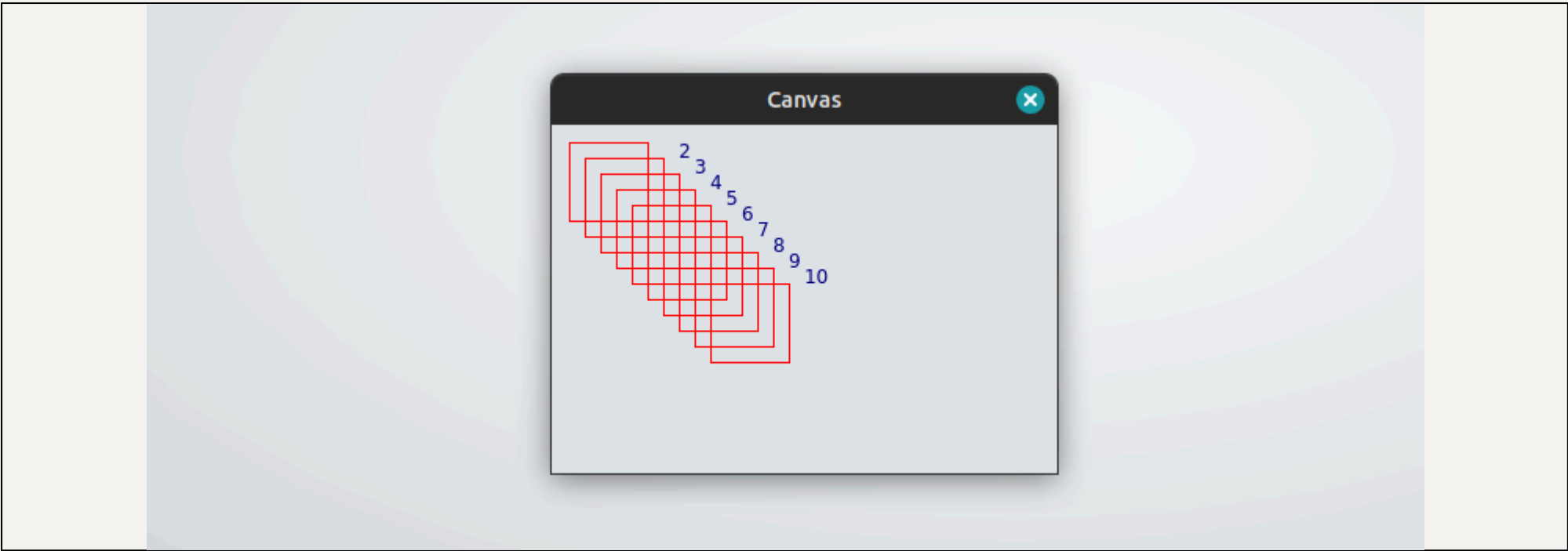
```
1 def demo_Table():
2     with ui.Dialog("Table"):
3         with ui.Col():
4             ui.Table(
5                 headers=("Length", "^Width", ">Height"),
6                 rows=[
7                     [21, 34, 56],
8                     [65, 87, 98],
9                     [21, 32, 54],
10                    [65, 76, 87],
11                ],
12            )
13
```

Widget: Canvas

Signature / Canvas

```
1 def Canvas(
2     paint: Callable[[QWidget, QPainter, QPaintEvent], None],
3     *,
4     setup: Callable[[QWidget, QPainter, QPaintEvent], None] | None=None,
5     name: str | None=None,
6     stretch: int=0,
7     width: int=0,
8     height: int=0,
9     **
10     kwargs: KwArgs
11 ) -> 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)
```



```
1 def demo_Canvas():
2
3     def render(_widget: QWidget, qp: QPainter, ch: ui.CanvasHelper):
4         for i in range(10):
5             with ch.pen(color=Qt.red, width=1):
6                 qp.drawRect(QRect(i*10, i*10, 50, 50))
7             with ch.pen(color=Qt.darkBlue, width=2):
8                 qp.drawText(i*10+60, i*10, str(i+1))
9
10    with ui.Dialog("Canvas"):
11        ui.Canvas(render, width=300, height=200)
12
13
```

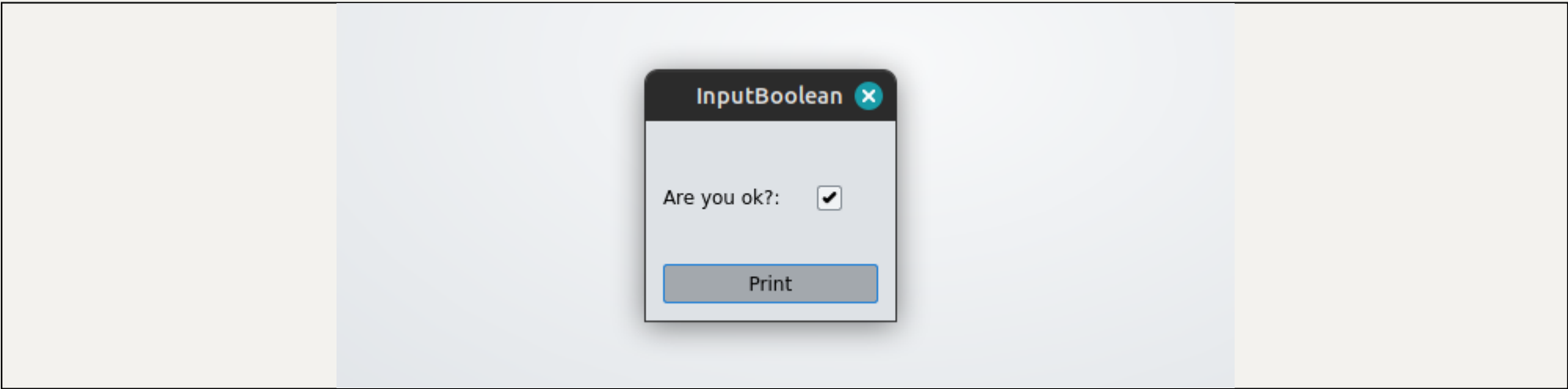
Inputs

Widget: InputBoolean

Signature / InputBoolean

```
1 def InputBoolean(
2     value: bool=False,
3     *,
4     name: str | None=None,
5     label: QWidget | str=None,
6     stretch: int=0,
7     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
8     add: bool=True,
9     **
10     kwargs: KwArgs
11 ) -> 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*)

```
1 def demo_InputBoolean():
2     with ui.Dialog("InputBoolean"):
3         val = ui.InputBoolean(label="Are you ok?:", alignment=Qt.AlignCenter)
4
5         @ui.button("Print")
6         def btn_print():
7             ui.print_log(val.value())
8
```

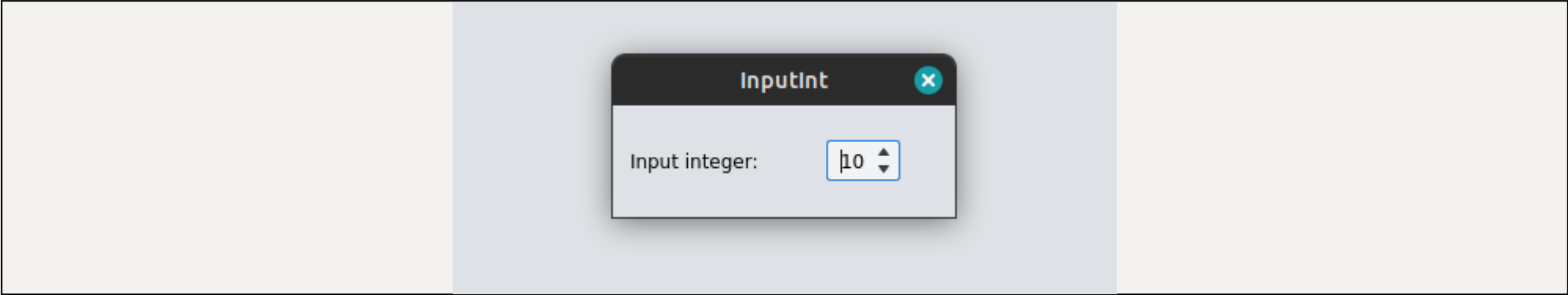
Widget: InputInt

Signature / InputInt

```
1 def InputInt(
2     value: int=0,
3     *,
4     name: str | None=None,
5     min: int=0,
6     max: int | None=None,
7     step: int=1,
8     label: QWidget | str=None,
9     stretch: int=0,
10    alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
11    add: bool=True,
```

```
12         **
13         kwargs: KwArgs
14     ) -> 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)
```

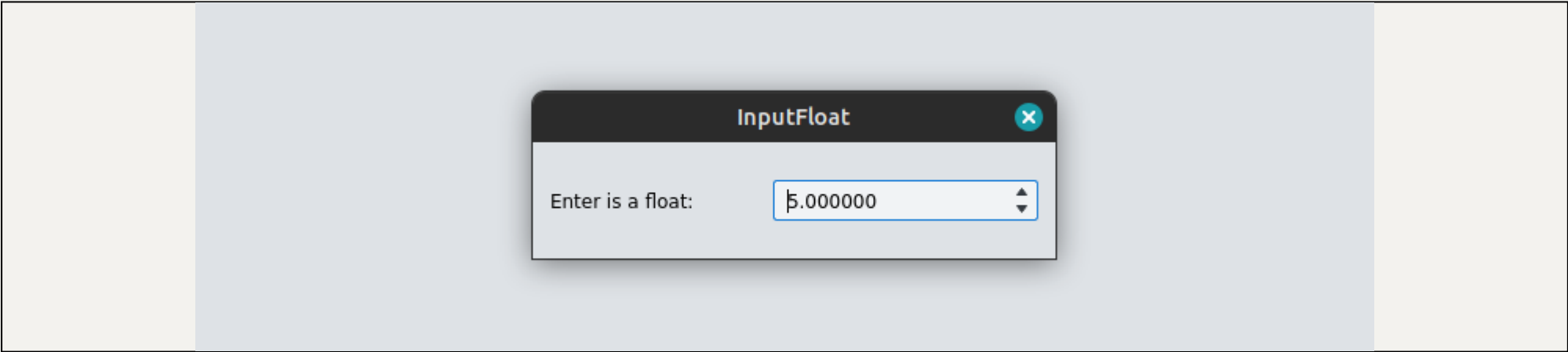
```
1 def demo_InputInt():
2     with ui.Dialog("InputInt"):
3         val = ui.InputInt(value=5, label="Enter is an int:", alignment=Qt.AlignCenter)
4
5         @ui.button("Print")
6         def print_int():
7             ui.print_log(val.value())
8
```

Widget: InputFloat

Signature / InputFloat

```
1 def InputFloat(
2     value: float=0.0,
3     *,
4     name: str | None=None,
5     min: float=0.0,
6     max: float=sys.float_info.max,
7     decimals: int=6,
8     step: float=0.01,
9     label: QWidget | str | None=None,
10    stretch: int=0,
11    alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
12    add: bool=True,
13    **
14    kwargs: KwArgs
15) -> 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*)

```
1 def demo_InputFloat():
2     with ui.Dialog("InputFloat"):
3         val = ui.InputFloat(value=5.0, label="Enter is a float:", alignment=Qt.AlignCenter)
4
5         @ui.button("Print")
6         def print_float():
7             ui.print_log(val.value())
8
```

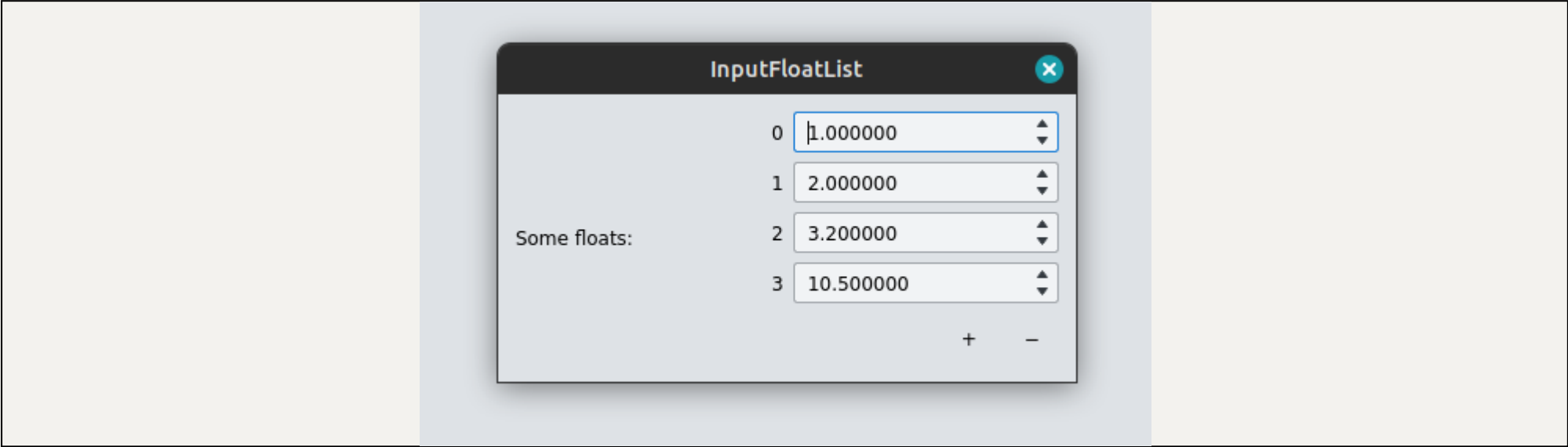
Widget: InputFloatList

Signature / InputFloatList

```
1 def InputFloatList(
2     values: list[float] | None=None,
3     label: QWidget | str | None=None,
4     *,
5     name: str | None=None,
6     label_fn: Callable[[int], str] | None=None,
```

```
7         count: int=0,
8         resizable: bool=False,
9         min_count: int=0,
10        add: bool=True,
11        **
12        kwargs: KwArgs
13    ) -> 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)
```

```

1  def demo_InputFloatList():
2      with ui.Dialog("InputFloatList"):
3          with ui.Col():
4              val = ui.InputFloatList(
5                  values=[1.0, 2.0, 3.2, 10.5],
6                  label="Some floats:",
7                  resizable=True,
8              )
9
10         @ui.button("Print")
11         def btn_print():
12             ui.print_log(val.value())
13

```

Widget: InputQuantity

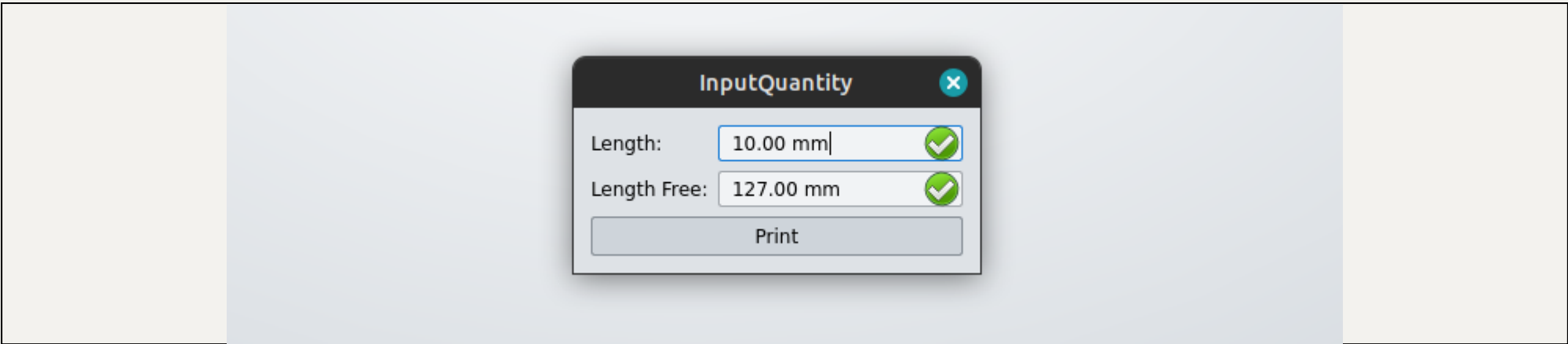
Signature / InputQuantity

```

1  def InputQuantity(
2      value: Numeric=None,
3      *,
4      name: str | None=None,
5      min: Numeric=None,
6      max: Numeric=None,
7      step: Numeric=1.0,
8      label: QWidget | str=None,
9      stretch: int=0,
10     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
11     unit: str | None=None,
12     obj: object | None=None,
13     property: str | None=None,
14     add: bool=True,
15     **
16     kwargs: KwArgs
17 ) -> 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)
```

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



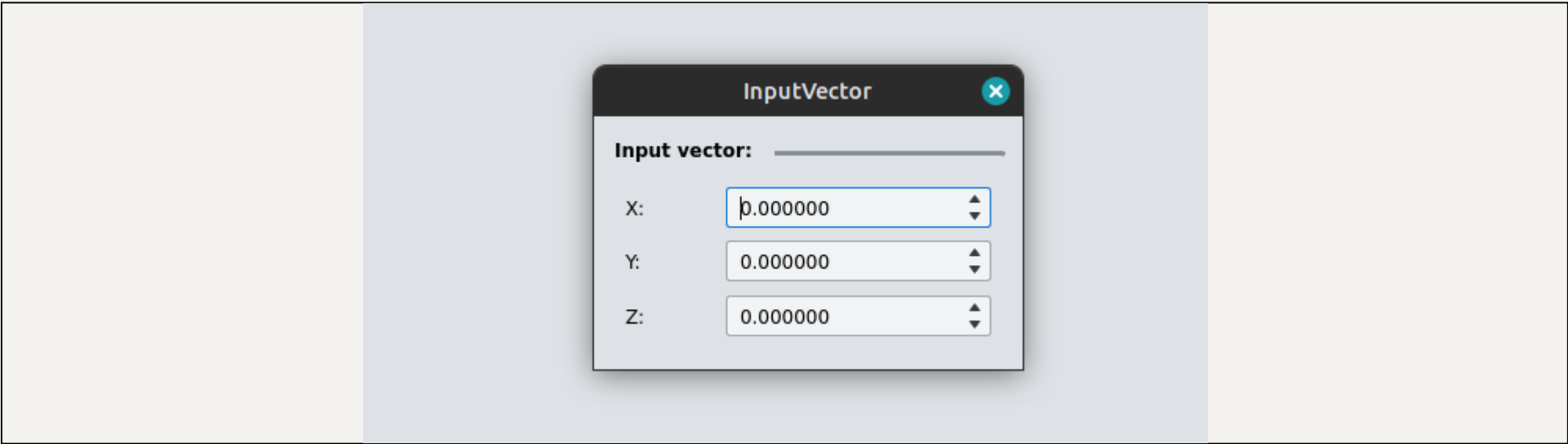
```
6         obj=App.activeDocument().getObject("Box1"),
7         property="Length",
8         unit="in",
9         alignment=Qt.AlignCenter,
10    )
11
12    quantity2 = ui.InputQuantity(
13        value=5.0, label="Length Free:", unit="in", alignment=Qt.AlignCenter,
14    )
15
16    @ui.button("Print")
17    def print_q():
18        ui.print_log(quantity.value(), quantity2.value())
19
```

Widget: InputVector

Signature / InputVector

```
1 def InputVector(
2     label: QWidget | str | None=None,
3     value: tuple | Vector=(0.0, 0.0, 0.0)
4 ) -> 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*)

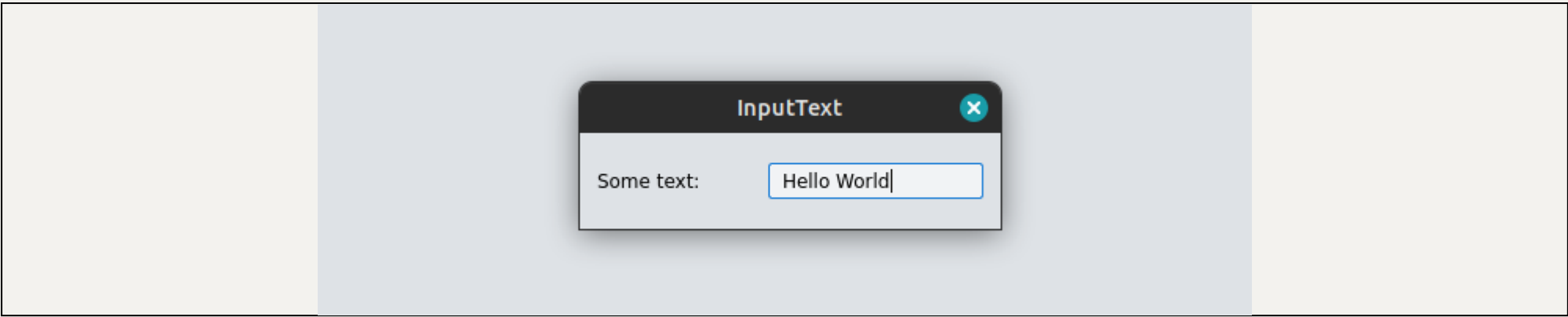
```
1 def demo_InputVector():
2     with ui.Dialog("InputVector"):
3         with ui.Col():
4             val = ui.InputVector(label="Input vector:")
5
6             @ui.button("Print")
7             def btn_print():
8                 ui.print_log(val.value())
9
```

Widget: InputText

Signature / InputText

```
1 def InputText(
2     value: str='',
3     *,
4     name: str | None=None,
5     label: QWidget | str=None,
6     stretch: int=0,
7     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
8     add: bool=True,
9     **
10     kwargs: KwArgs
11 ) -> 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:

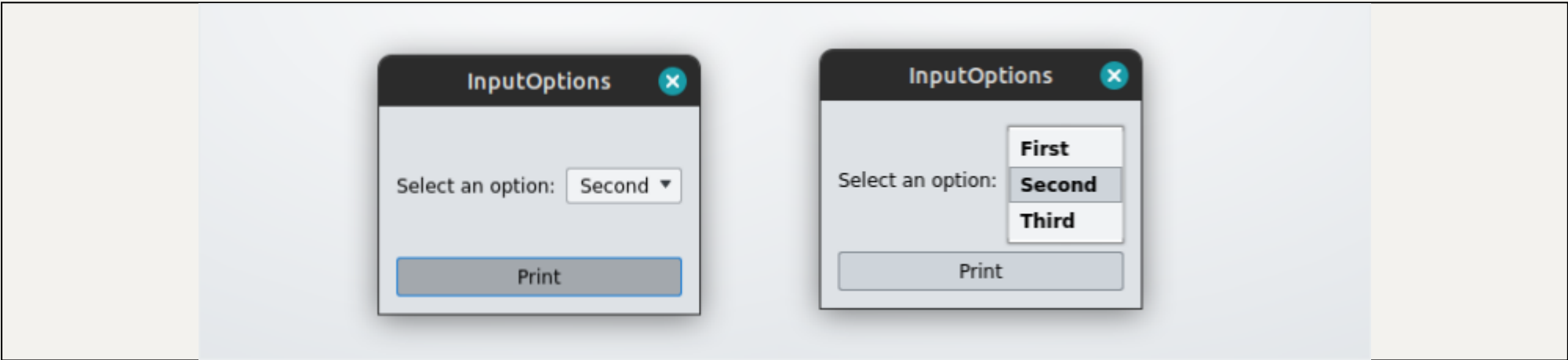
```
1 def demo_InputText():
2     with ui.Dialog("InputText"):
3         text = ui.InputText(value="Hello World", label="Some text:", alignment=Qt.AlignCenter)
4
5         @ui.button("Print")
6         def print_text():
7             ui.print_log(text.value())
8
```

Widget: InputOptions

Signature / InputOptions

```
1 def InputOptions(  
2     options: dict[str, Hashable],  
3     value: Hashable=None,  
4     label: str | None=None,  
5     *,  
6     name: str | None=None,  
7     stretch: int=0,  
8     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,  
9     add: bool=True,  
10    **  
11    kwargs: KwArgs  
12) -> 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()
**kwargs	dict[str, Any]	Qt properties

ComboBox widget.

Example:

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

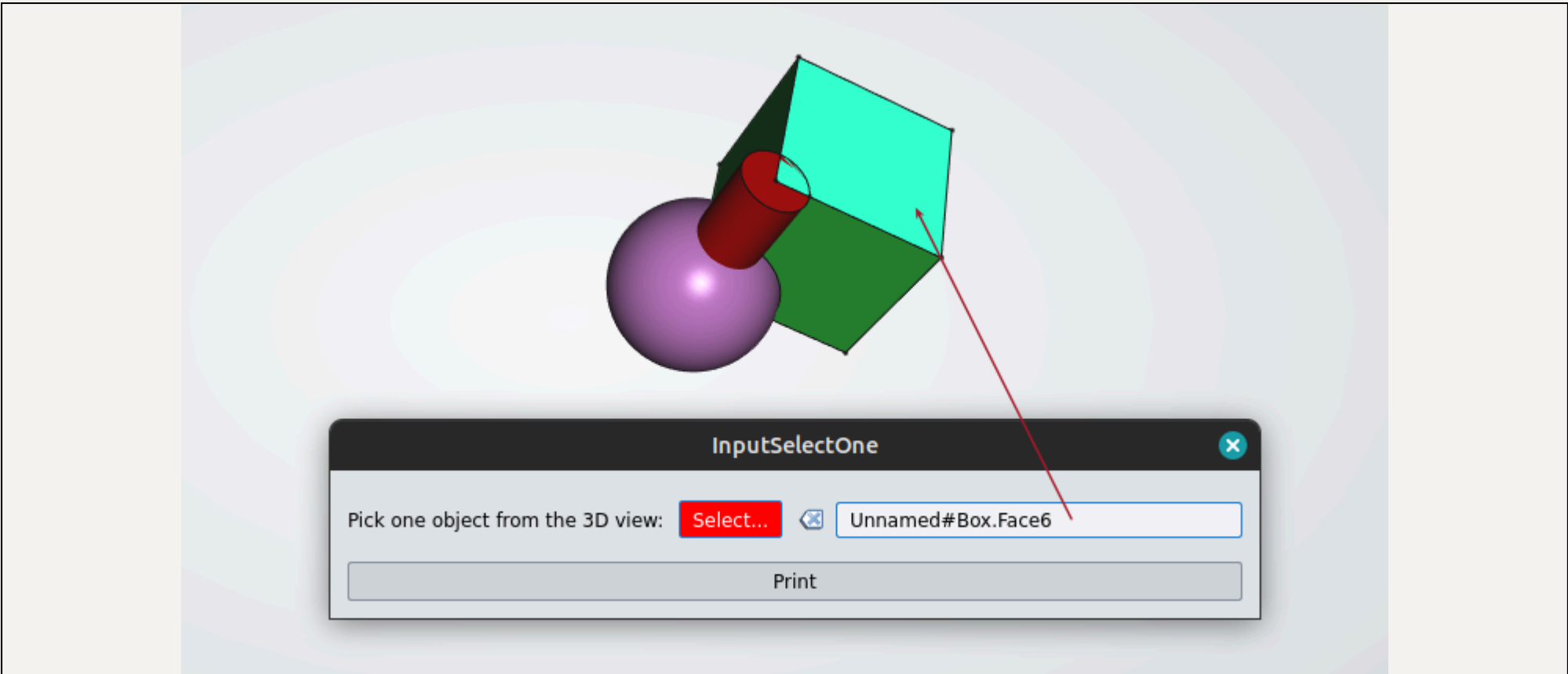
```
1 def demo_InputOptions():
2     with ui.Dialog("InputOptions"):
3         val = ui.InputOptions(
4             options={
5                 "First": 10,
6                 "Second": 20,
7                 "Third": 30,
8             },
9             value=20,
10            label="Select an option:",
11            alignment=Qt.AlignCenter,
12        )
13
14        @ui.button("Print")
15        def print_opt():
16            ui.print_log(val.value())
17
```

Widget: InputSelectOne

Signature / InputSelectOne

```
1 def InputSelectOne(
2     label: str | None=None,
3     *,
4     name: str | None=None,
5     active: bool=False,
6     auto_deactivate: bool=True
7 ) -> 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	<i>description</i> , defaults to True

3D Selection widget. Allows to pick an object.

Example:

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

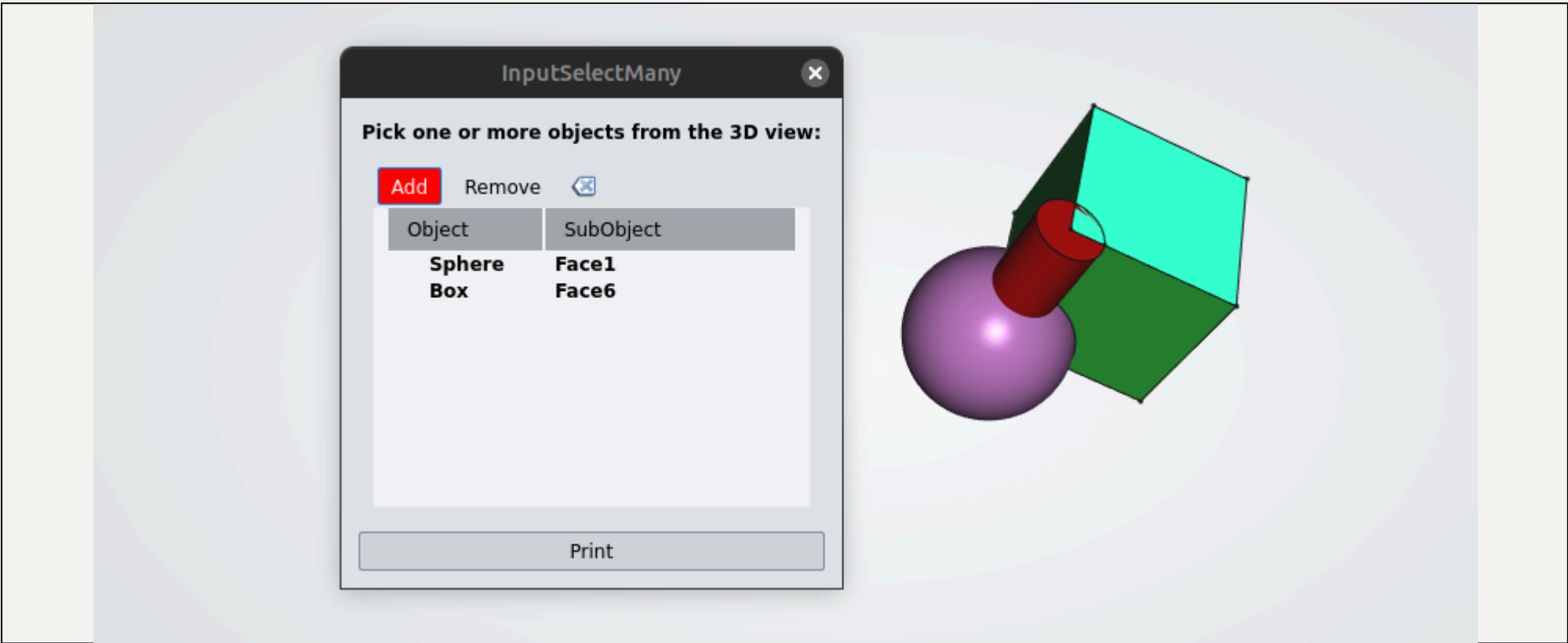
```
1 def demo_InputSelectOne():
2     with ui.Dialog("InputSelectOne", modal=False):
3         val = ui.InputSelectOne(label="Pick one object from the 3D view:")
4
5         @ui.button("Print")
6         def print_sel():
7             ui.print_log(val.value())
8
```

Widget: InputSelectMany

Signature / InputSelectMany

```
1 def InputSelectMany(  
2     label: str | None=None,  
3     *,  
4     name: str | None=None,  
5     active: bool=False  
6 ) -> 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)
```

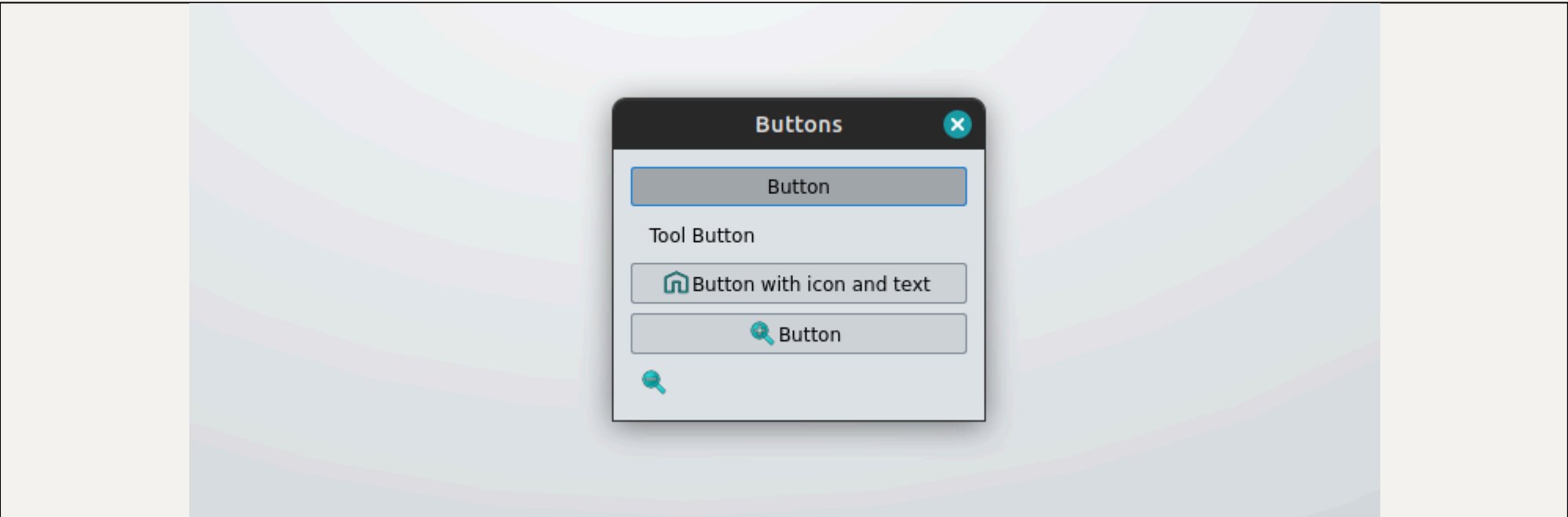
```
1 def demo_InputSelectMany():
2     with ui.Dialog("InputSelectMany", modal=False):
3         val = ui.InputSelectMany(label="Pick one or more objects from the 3D view:")
4
5         @ui.button("Print")
6         def print_sel():
7             ui.print_log(val.value())
8
```

Widget: button

Signature / button

```
1 def button(
2     label: str | None=None,
3     *,
4     tool: bool=False,
5     icon: QIcon | str | None=None,
6     stretch: int=0,
7     alignment: Qt.Alignment=DEFAULT_ALIGNMENT,
8     add: bool=True,
9     **
10    kwargs: KwArgs
11) -> 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)`

```
1 def demo_buttons():
2     with ui.Dialog("Buttons", modal=False):
3
4         @ui.button("Button")
5         def btn1():
6             ui.print_log("Hello btn1")
7
8         @ui.button("Tool Button", tool=True)
9         def btn2():
10             ui.print_log("Hello btn2")
11
12         @ui.button("Button with icon and text", icon=QIcon(":/icons/Std_ViewHome.svg"))
13         def btn3():
14             ui.print_log("Hello btn3")
15
16         @ui.button(icon=QIcon(":/icons/zoom-in.svg"))
17         def btn4():
18             ui.print_log("Hello btn4")
19
20         @ui.button(icon=QIcon(":/icons/zoom-out.svg"), tool=True)
21         def btn5():
22             ui.print_log("Hello btn4")
23
```

Layout tools

Function: Stretch

Signature / Stretch

```
1 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

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

Docs / Spacing

ARGUMENT	TYPE	DESCRIPTION
size	int	spacing

Adds spacing ro the current layout.