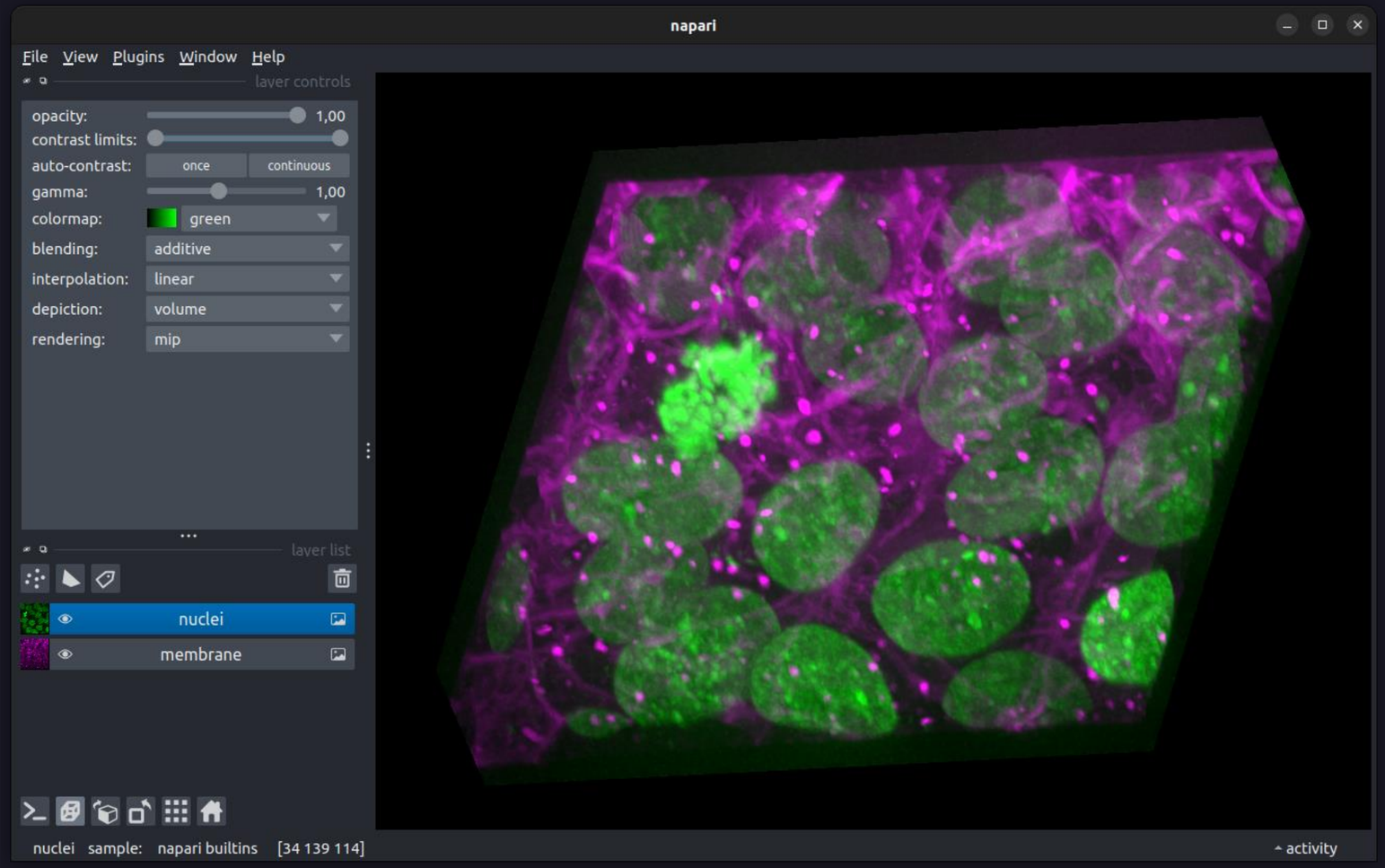
# Napari - multi-dimensional image viewer for python

Grzegorz Bokota

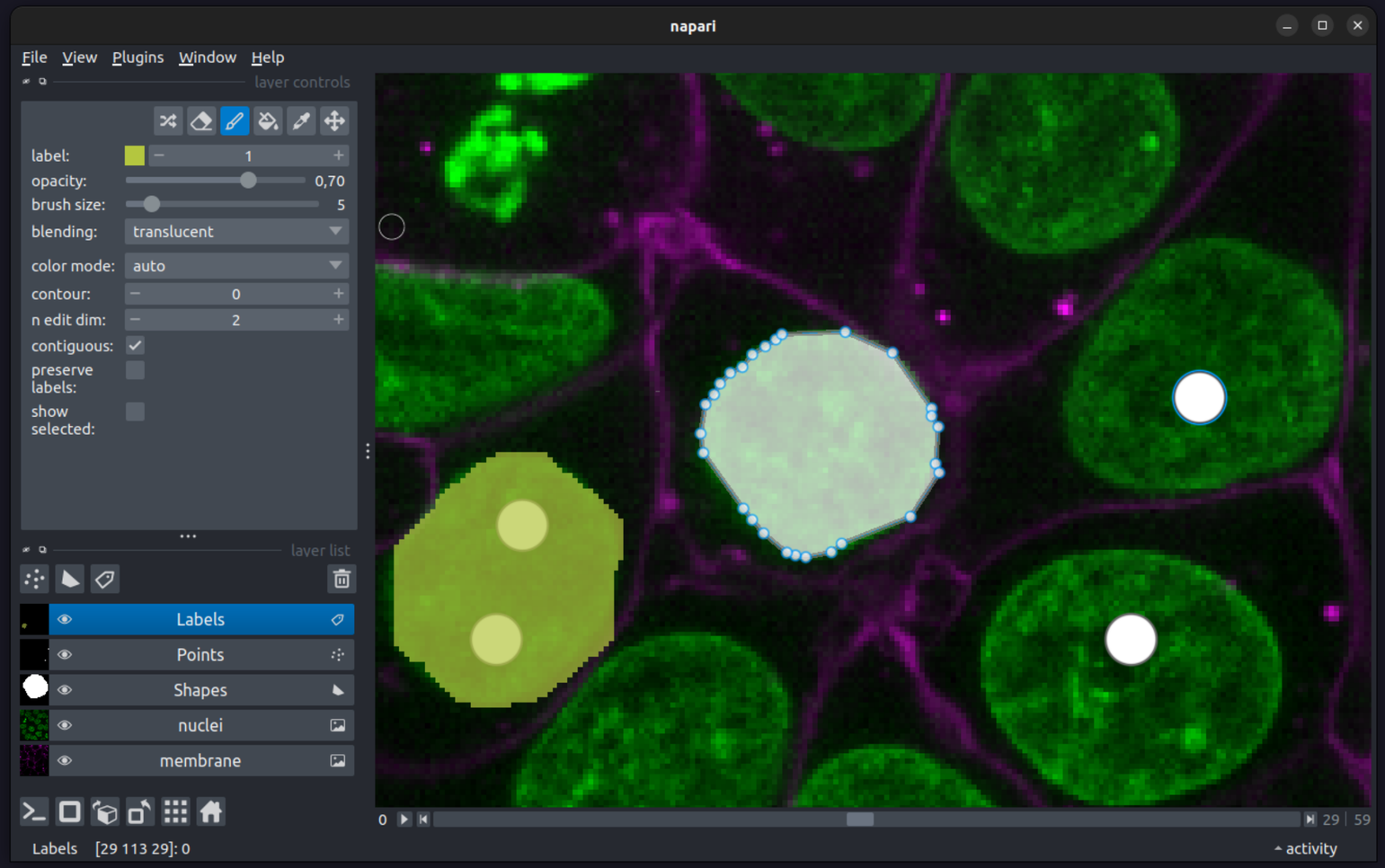
# Napari

- Native
- Pluggable
- Embeddable



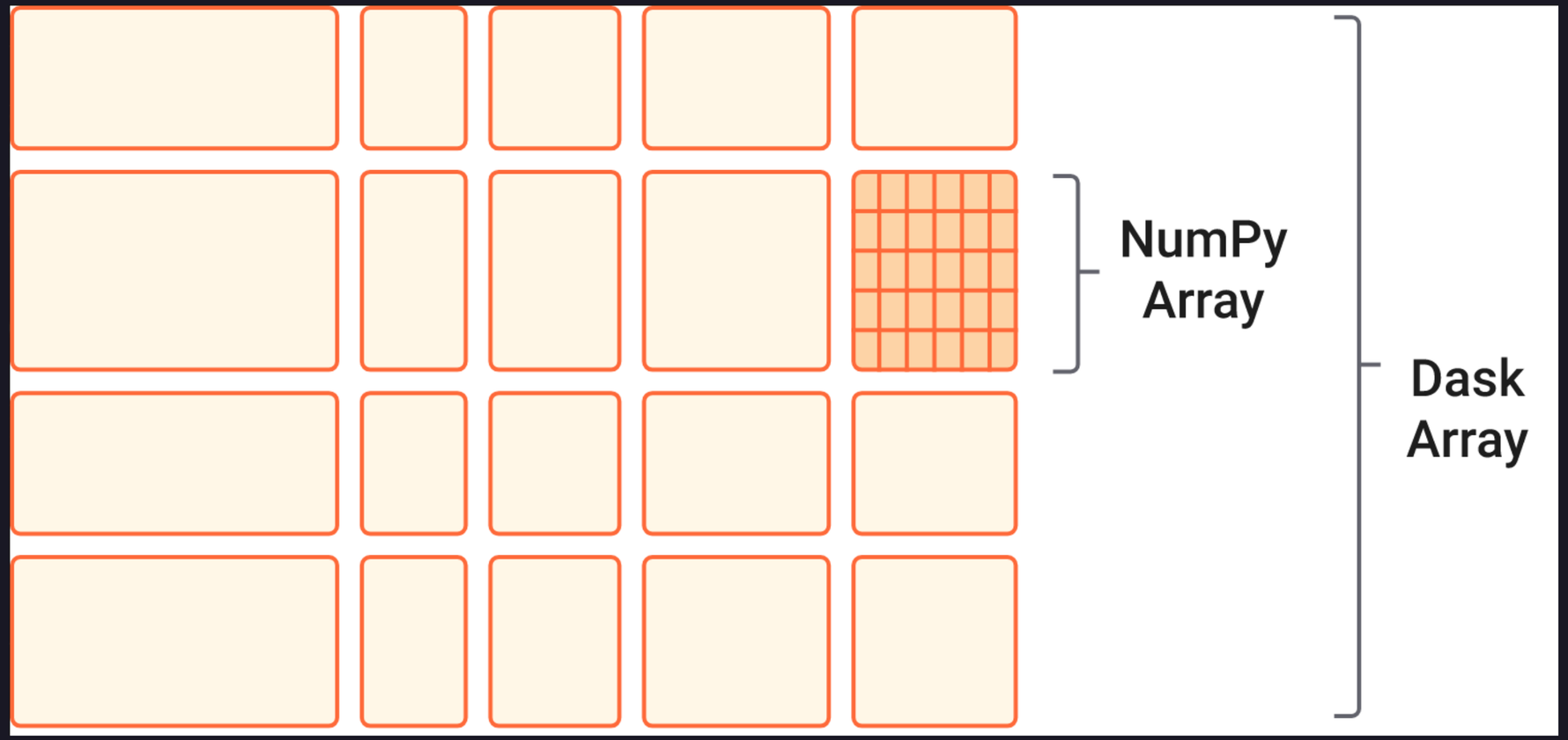
# Supported data

- Image
- Labels
- Points
- Shapes
- Surfaces
- Tracks
- Vectors
- (WIP) Graph



# Supported array type

- Numpy arrays
- Dask arrays
- Zarr arrays



# Team

Core developers currently are Scientists from:

- Australia
- England
- Germany
- Poland
- Switzerland
- USA

Contributors are scientists and data analysists all around the world

# Logo

# For whom?

For anyone who works with spatial data and need to visualize them

- Bioimage analysis
- Materials science
- geoJSON data
- ?

# License

BSD-3 clause

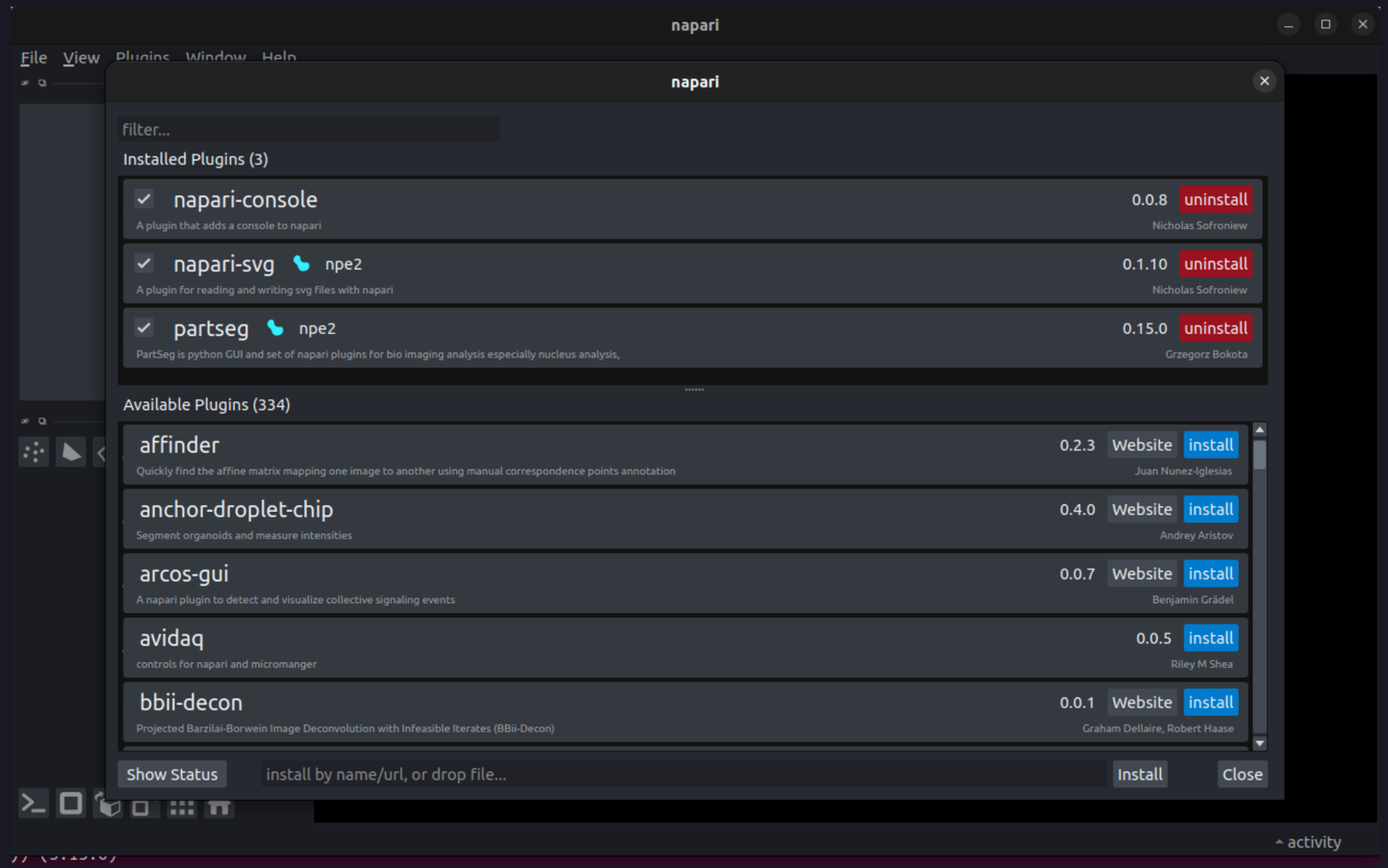
But depends on Qt that is dual licensed LGPL and paid commercial

# How to use napari

- As application
- As the viewer opened from the script/notebook
- As a viewer embedded in your own application

# Napari as application

- Limited functionality without plugins
- Plugins are Python packages

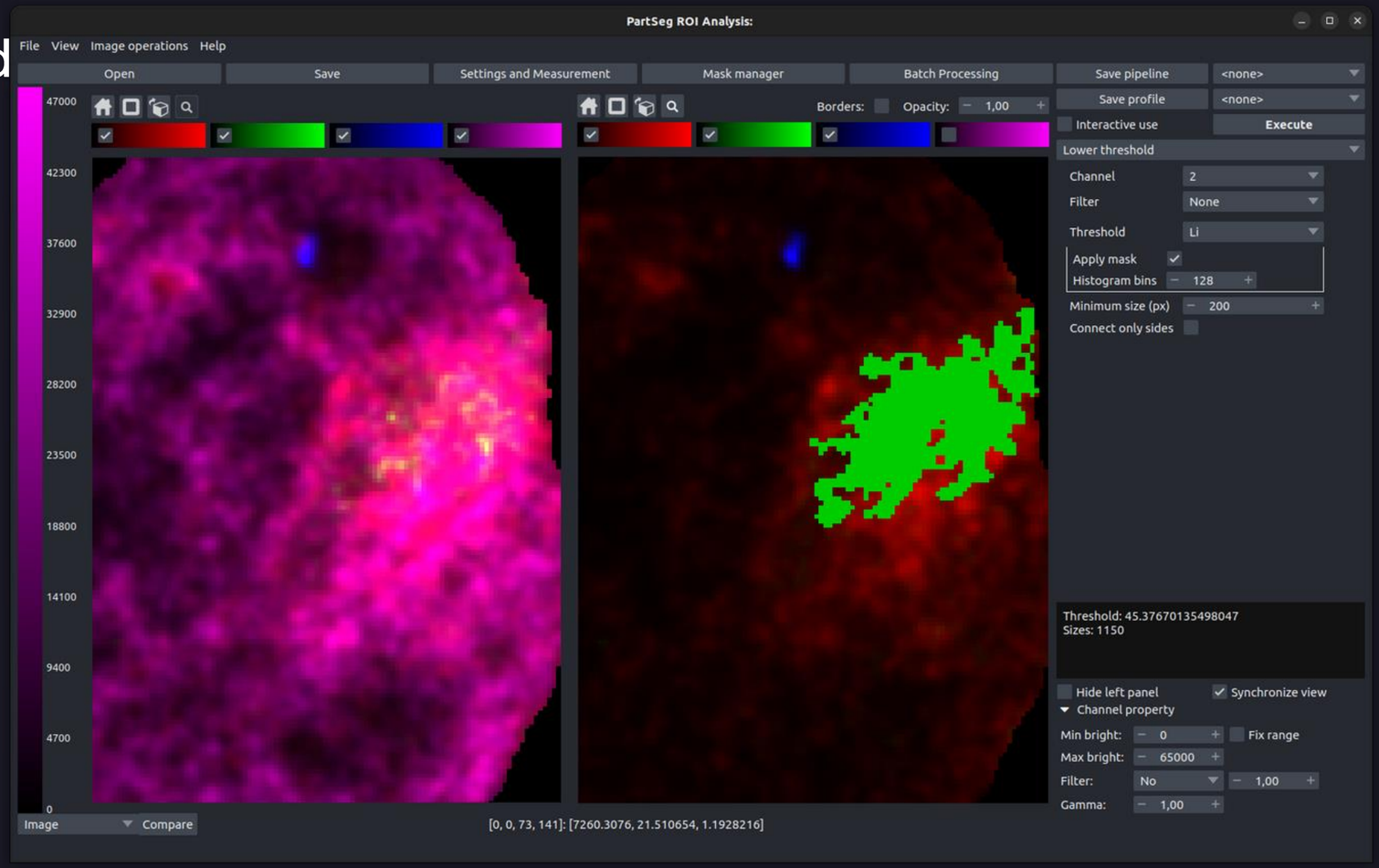


# Napari  from notebook

On a locally executed notebook, napari could be used  as a viewer for comfortable view or annotate data

# Napari embeded

It could be easily embedded in any Qt (PyQt or PySide) application



# Resources

- https://napari.org/
- https://github.com/napari/napari/
- https://forum.image.sc/tag/napari

# Nice napari releated libraries

- **magicgui**
- **superqt**
- **psygnal**

# magicgui

create ipywidget from function

In [1]:
```
from magicgui import magicgui, use_app
```

In [2]:
```
use_app("ipynb")
# %gui qt
```

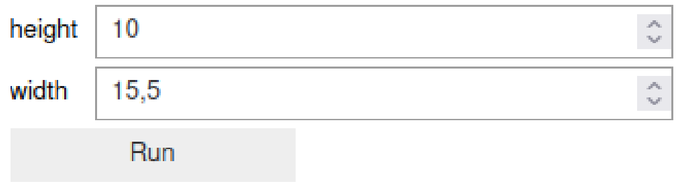
Out[2]:
```
<magicgui app, wrapping the ipynb GUI toolkit>
```

In [4]:
```
@magicgui
def square_calculator(height: float, width: float = 15.5) -> float:
    return width * height
```

In [5]:
```
@square_calculator.called.connect
def callback(a=None):
    print("Square is: ", a)
```

In [7]:
```
square_calculator
```

Out[7]:



# magicgui

create custom
ipywidget

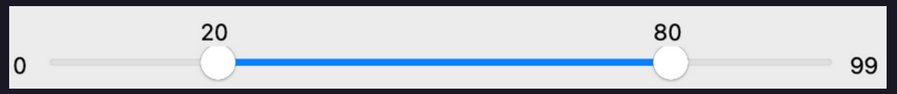
```
In [11]: from magicgui.widgets import Container, Button, FloatSlider

In [14]: c = Container()
         slider_1 = FloatSlider(min=1, max=10, label="a")
         slider_2 = FloatSlider(min=5, max=20, label="b")

         c.append(slider_1)
         c.append(slider_2)

         @c.changed.connect
         def power():
             print("power", slider_1.value ** slider_2.value)

In [15]: c
Out[15]: a 5.00
         b 10.00

         power 32.0
         power 243.0
         power 1023.9999999999994
         power 3125.0
         power 15625.0
         power 78125.0
```

https://pyapp-kit.github.io/magicgui/

# superqt

Set of useful Qt extensions that are mainly developed for napari  but could be reused in other projects:

- ensure_main_thread/ensure_object_thread - for merge qt and non-qt code
- thread_worker - fun function in thread with decorator
- Throttling utilities
- QEnumComboBox - combo box from Enum
- QLabeledRangeSlider




And many others

https://pyapp-kit.github.io/superqt/

# psygnal

Pure python (compiled with mypyc) library for implements Qt like Singnals in non-qt code part/base

- Signals using weakrefs if possible to store callbacks
- Evented dataclass / EventedModel(pydantic.BaseModel)
- Support passing events between threads

https://pyapp-kit.github.io/superqt/

# Community meeting

Every Wednesday

- Every 2-weeks 9:00 with Asia
- Every 2 weeks 17:30 with USA

https://napari.zulipchat.com/

# Thanks for your attention

https://github.com/Czaki/PyConPL2023