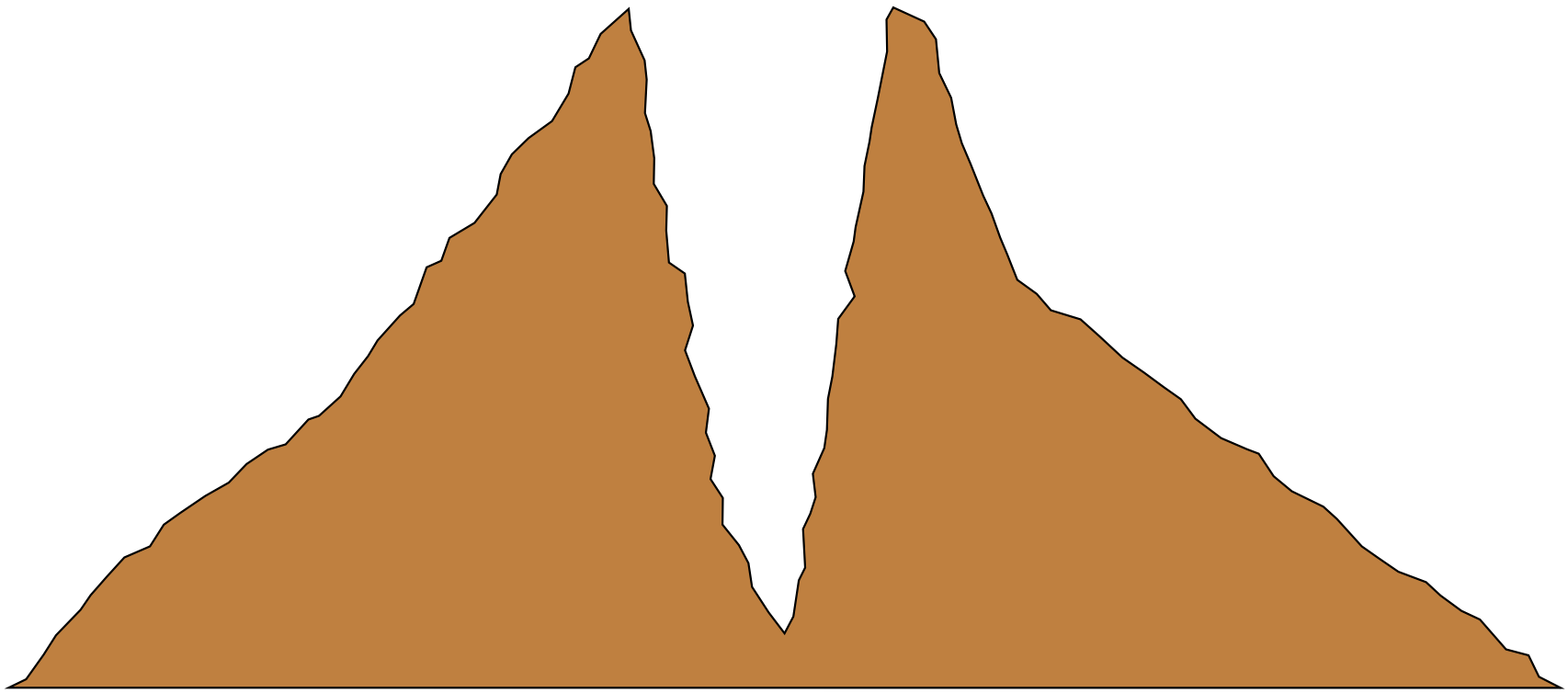
hhmi | janelia
Research Campus

Philipp Hanslovsky

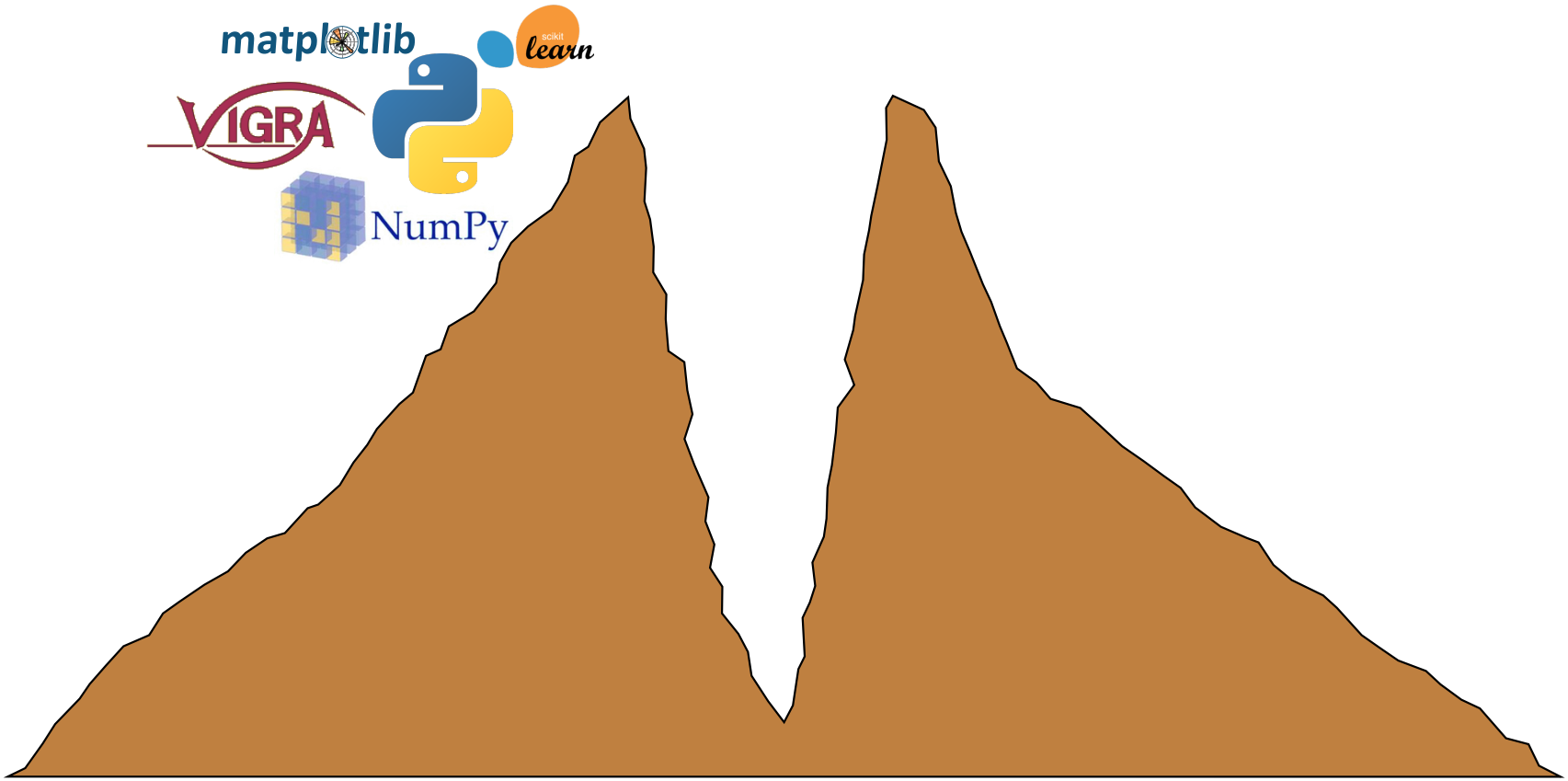
# imglyb
# Bridging The Chasm Between ImageJ and NumPy

May 11, 2017

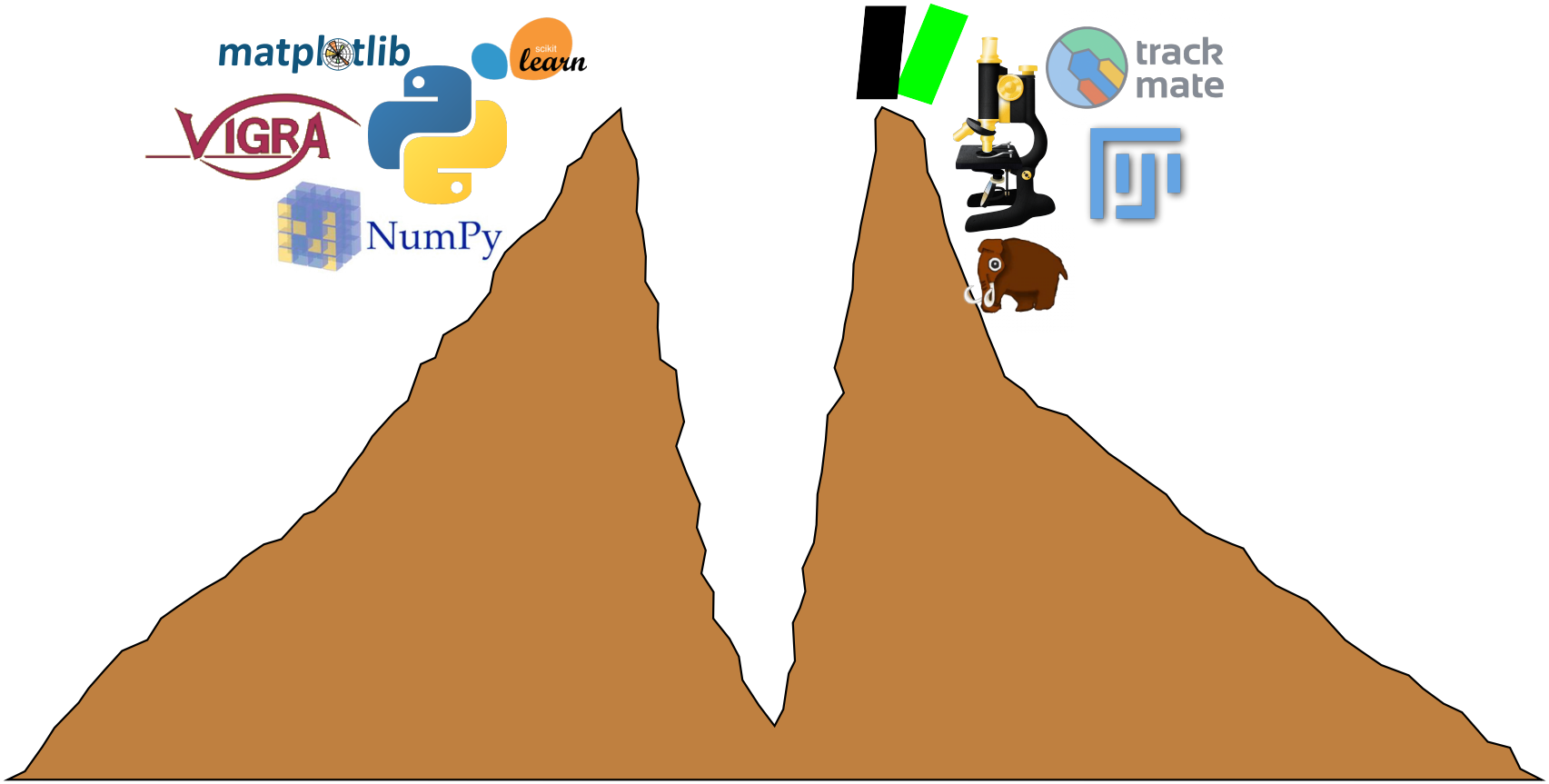
# Image Analysis in Bioinformatics



# Image Analysis in Bioinformatics



# Image Analysis in Bioinformatics



# What is Python?

- Interpreted language with dynamic typing.

# What is Python?

- Interpreted language with dynamic typing.
- Access to native memory.

# What is Python?

- Interpreted language with dynamic typing.
- Access to native memory.
- Efficient software through C/C++ extensions.

# What is Python?

- Interpreted language with dynamic typing.
- Access to native memory.
- Efficient software through C/C++ extensions.
- Interactive shell and notebooks.

# What is Java?

- Statically typed language that compiles into byte code.

# What is Java?

- Statically typed language that compiles into byte code.
- Byte code is executed within a virtual machine (JVM).

# What is Java?

- Statically typed language that compiles into byte code.
- Byte code is executed within a virtual machine (JVM).
- No access to native memory through Java language API.

# Ways out of the dilemma

- Inter-process communication:

# Ways out of the dilemma

- Inter-process communication:
  - ✓ Comparatively easy.
  - ✓ Can be extended to any language.
  - ✗ No shared memory.

# Ways out of the dilemma

- Inter-process communication:
  - ✓ Comparatively easy.
  - ✓ Can be extended to any language.
  - ✗ No shared memory.
- Python and JVM in the same process:

# Ways out of the dilemma

- Inter-process communication:
  - ✓ Comparatively easy.
  - ✓ Can be extended to any language.
  - ✗ No shared memory.
- Python and JVM in the same process:
  - ✓ Shared memory possible.
  - ✓ Avoids unnecessary copies of data.

# Ways out of the dilemma

- Inter-process communication:
  - ✓ Comparatively easy.
  - ✓ Can be extended to any language.
  - ✗ No shared memory.

- Python and JVM in the same process:
  - ✓ Shared memory possible.
  - ✓ Avoids unnecessary copies of data.

# What do we need to do?

- ✓ Start JVM from Python process.

  `https://github.com/kivy/pyjnius`

# What do we need to do?

- ✓ Start JVM from Python process.

  `https://github.com/kivy/pyjnius`

  Make ImgLib2 and NumPy understand each other.

# What do we need to do?

- ✓ Start JVM from Python process.

  `https://github.com/kivy/pyjnius`

- ✓ Make ImgLib2 and NumPy understand each other.

  `https://github.com/imglib/imglib2-unsafe`

  `https://github.com/hanslovsky/imglib2-imglyb`

# Features

- NumPy and ImgLib2 share memory.

# Features

- NumPy and ImgLib2 share memory.
- Convenience layer for easy access to Java classes and conversions between NumPy and ImgLib2.

# Features

- NumPy and ImgLib2 share memory.
- Convenience layer for easy access to Java classes and conversions between NumPy and ImgLib2.
- Use BigDataViewer to visualize dense 3D+t data with arbitrary slicing from within Python.

# Features

- NumPy and ImgLib2 share memory.
- Convenience layer for easy access to Java classes and conversions between NumPy and ImgLib2.
- Use BigDataViewer to visualize dense 3D+t data with arbitrary slicing from within Python.
- Use a **real** IPython console in ImageJ.

# Features

- NumPy and ImgLib2 share memory.
- Convenience layer for easy access to Java classes and conversions between NumPy and ImgLib2.
- Use BigDataViewer to visualize dense 3D+t data with arbitrary slicing from within Python.
- Use a **real** IPython console in ImageJ.
- Populate ImgLib2 caches with NumPy arrays.

# Features

- NumPy and ImgLib2 share memory.
- Convenience layer for easy access to Java classes and conversions between NumPy and ImgLib2.
- Use BigDataViewer to visualize dense 3D+t data with arbitrary slicing from within Python.
- Use a **real** IPython console in ImageJ.
- Populate ImgLib2 caches with NumPy arrays.
- Unrelated trickery: Embed Java UIs in native Qt applications.

# Where to get?

| | |
|---|---|
| imglyb | `https://github.com/hanslovsky/imglib2-imglyb` |
| examples | `https://github.com/hanslovsky/imglyb-examples` |
| imagey | `https://github.com/hanslovsky/imagey` |
| imglib2-unsafe | `https://github.com/imglib/imglib2-unsafe` |
| PyJNIus | `https://github.com/kivy/pyjnius` |
| slides | `https://gist.github.com/hanslovsky/39aa3a29cb49270e2c3b7750bdfe15c8` |

```
$ conda create -n imglyb -c hanslovsky -c ukoethe imglyb-examples
```

# How to use

Set up your environment (not necessary if loaded from conda):

```
# bash
export JAVA_HOME=/path/to/JAVA_HOME
export PYJNIUS_JAR=/path/to/pyjnius.jar
export IMGLYB_JAR=/path/to/imglib2-imglyb-<VERSION>.jar
```

This goes at the top of your Python file:

```
# import imglyb before jnius
import imglyb
from imglyb import util
# import from jnius what you need
from jnius import autoclass, cast, PythonJavaClass, java_method
```

# Wrap NumPy arrays in ImgLib2

```
import imglyb
from imglyb import util

import numpy as np

img = np.random.rand( 300, 200, 100 ) * 2**16
wrapped = util.to_imglib( img )
util.BdvFunctions.show( wrapped, "wrapped image" )

rgba = np.random.randint( 2**32, size = ( 300, 200, 100 ), dtype=np.uint32 )
wrapped_rgba = util.to_imglib_argb( rgba )
util.BdvFunctions.show( wrapped_rgba, "wrapped rgba image" )
```

# Java interface in Python

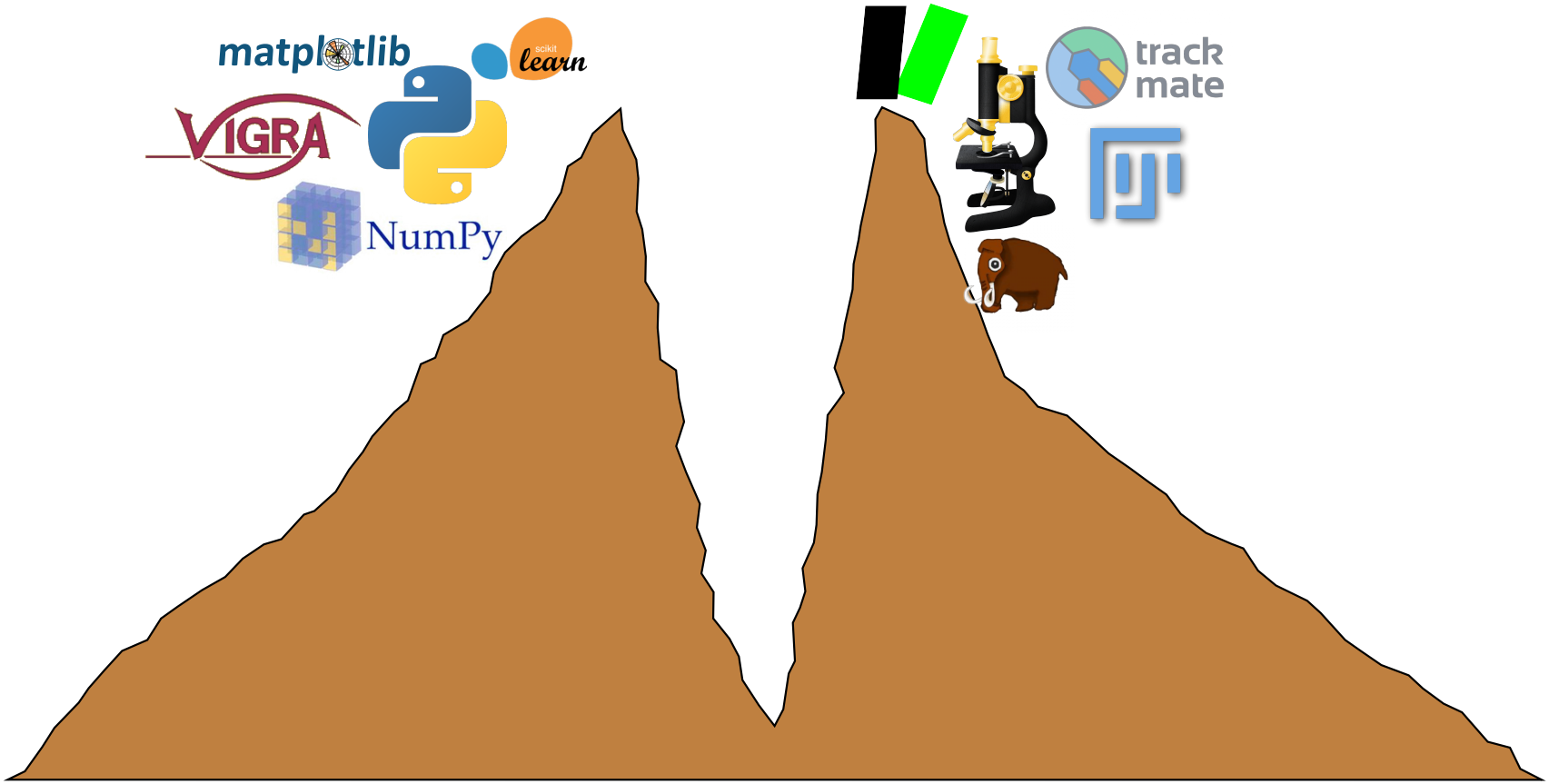
```
import imglyb
from jnius import PythonJavaClass, java_method
class GenericOverlayRenderer( PythonJavaClass ):
    __javainterfaces__ = ["net/imglib2/ui/OverlayRenderer"]

    def __init__( self, draw_overlays, set_canvas_size ):
        super( GenericOverlayRenderer, self ).__init__()
        self.draw_overlays = draw_overlays
        self.set_canvas_size = set_canvas_size

    @java_method( "(Ljava/awt/Graphics;)V" )
    def drawOverlays( self, g ):
        try:
            self.draw_overlays( g )
        except Exception as e:
            print( e )
            raise e

    @java_method( "(II)V" )
    def setCanvasSize( self, width, height ):
        try:
            self.set_canvas_size( width, height )
        except Exception as e:
            print( e )
            raise e
```

# Examples

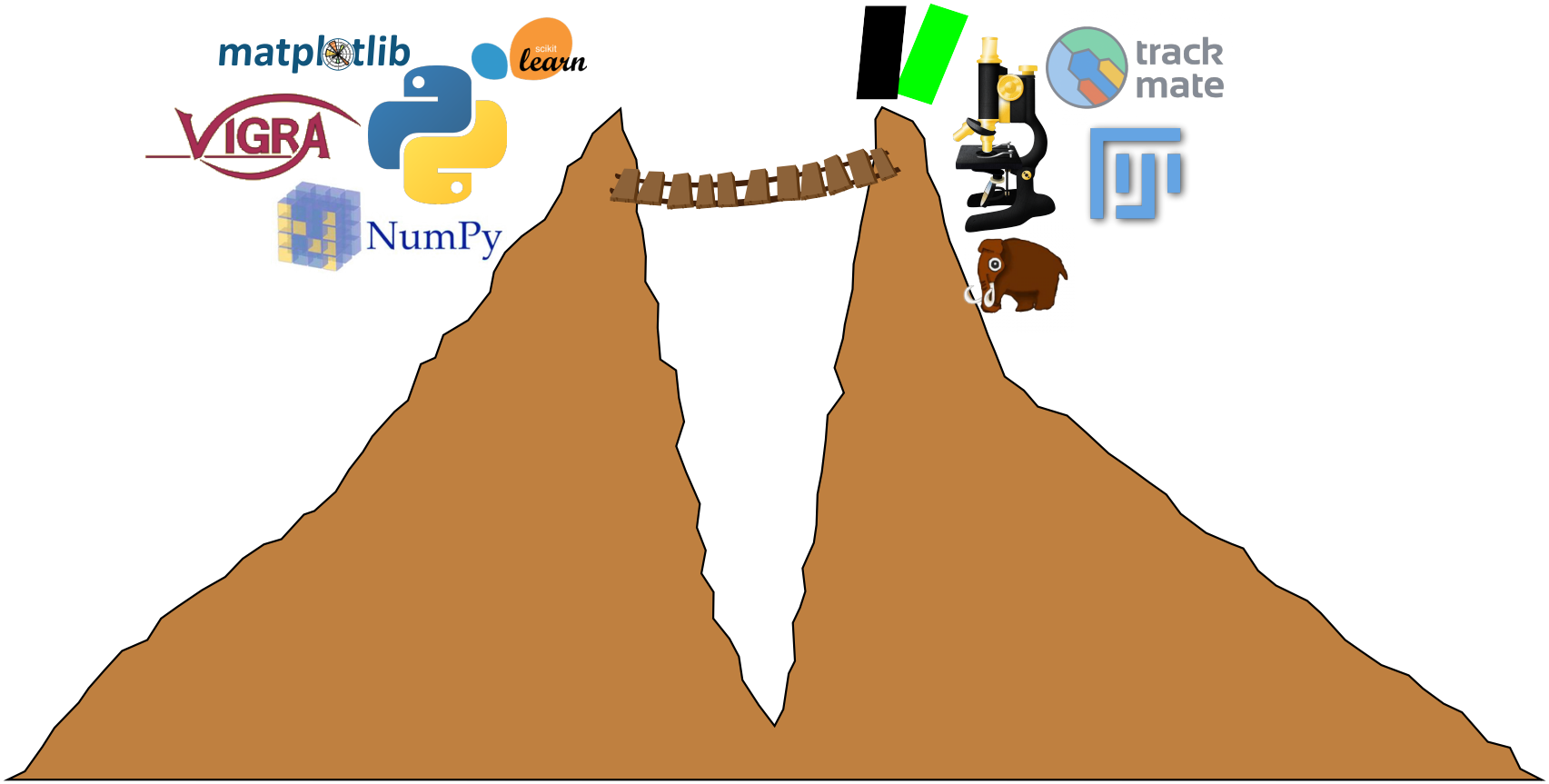
Run imglyb examples:

```
# bash
# Basic visualization and application of Python and ImgLib2 filters
python -m imglyb-examples.butterfly
# Python implemented overlays for BigDataViewer
python -m imglyb-examples.bdv-hello-world
# Python implementation of a painting tool for BigDataViewer
python -m imglyb-examples.bdv-painter
# Embed BigDataViewer in Qt application
python -m imglyb-examples.qt-awt
# Basic visualization of series of 2D NumPy arrays
python -m imglyb-examples.views-stack
```

# Image Analysis in Bioinformatics



# Image Analysis in Bioinformatics



# To-Do

- Get it to work on OSX and Windows

# To-Do

- Get it to work on OSX and Windows
- Provide conda packages with latest changes

# Resources

https://www.python.org


http://www.numpy.org


https://matplotlib.org


https://ukoethe.github.io/vigra


http://scikit-learn.org

https://imagej.net/ImageJ2


https://imagej.net/TrackMate

https://fiji.sc/

https://imagej.net/MaMuT

https://imagej.net/ImgLib2

https://openclipart.org/detail/209218/firebog-bridge