Module 3:
OMERO and Jupyter Notebooks

Use Python API to interact with OMERO.server

Workshop in 4 Modules

May 13th, 2024

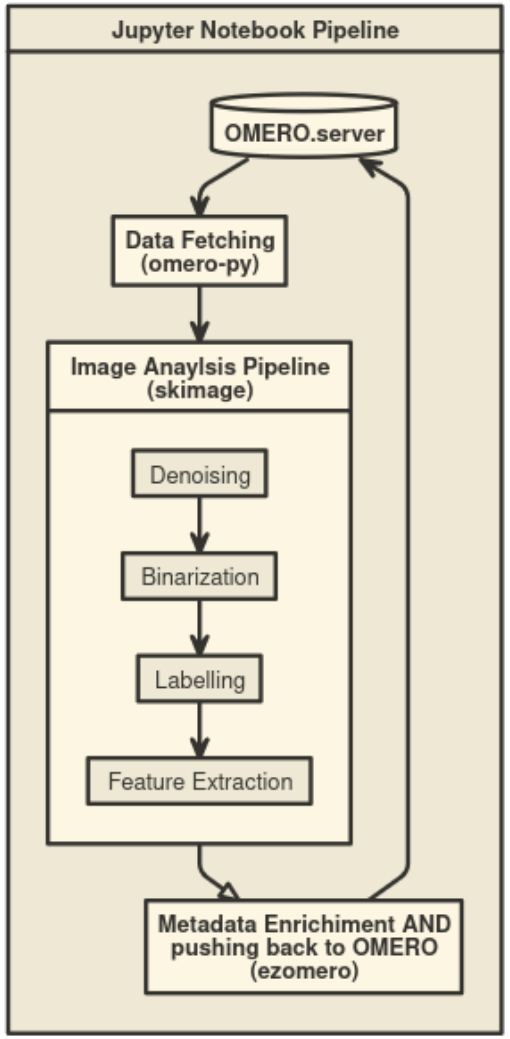
Trainers: Michele Bortolomeazzi, **Riccardo Massei**, Christian Schmidt

Support: Lena Krämer & Tom Boissonnet

Program for today…

Time	Activity
13:45 – 13:55	**Introduction and general set-up**
14:00 -14:20	**Interacting with OMERO using python** *Short presentation on omero-py and existing python libraries (focus on ezomero)*
14:20 - 15:00	**All together step-by-step pipeline** Writing together a pipeline using Jupyter Notebook for image analysis
15:00 - 15:40	**Group Exercises**
15:40 - 15:45	**Wrap-up and feedback**

What are we gonna do today?



THE TRAINERS will:

- Introduce you the the OMERO Python API
- Show different open source python libraries usefull to interact with OMERO
- Support with technical problems and questions

THE TRAINERS and THE TRAINEEs will:

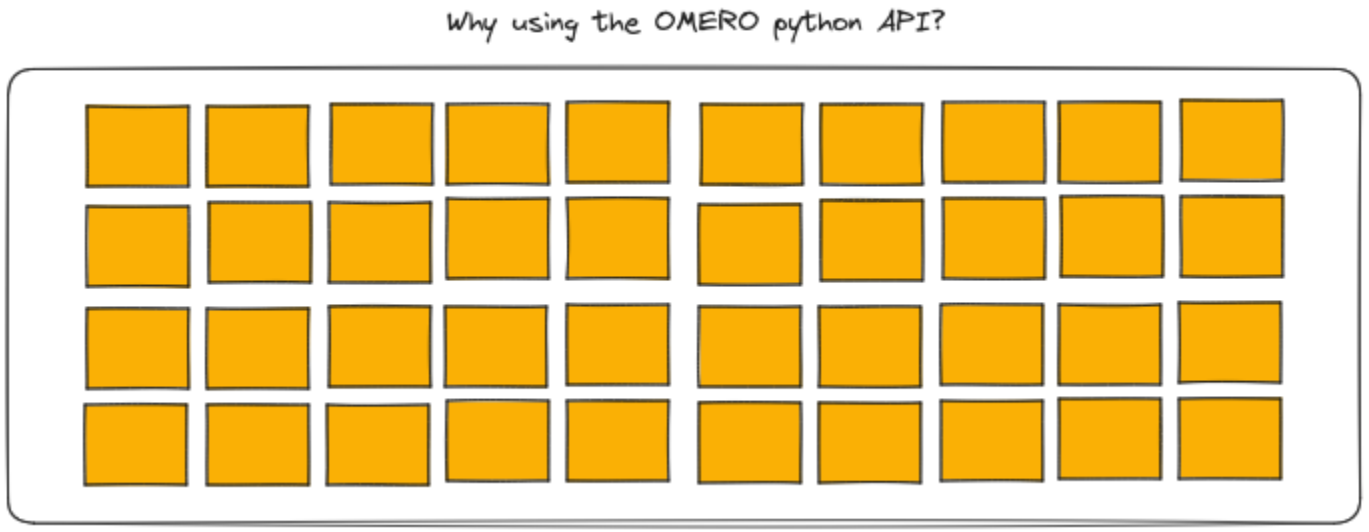
- Build together a simple pipeline for data transfer, analysis and annotation using Jupyter Notebook
- Support and help each other in the learning and teaching

THE TRAINEEs will:

- Try to build an own pipeline based on same example
- Do not be scare to ask questions
- Feel free to decide your own learning path

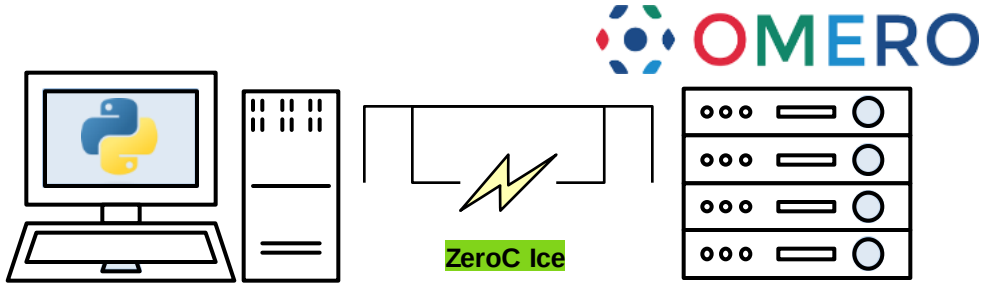
https://github.com/rmassei/2024-jn-omero-pipeline

You start!



Go to **Excalidraw** and write your own opinion!

OMERO Python API – Basics and overview



The OMERO Python API allows users and developers to interact with an OMERO server using python

Four „simple“ steps to say hello to OMERO trough python:

1. Create a connection object

connection_object = BlitzGateway(„username“, „password“, host=“server“, port = 4064)

BlitzGateway (component of omero-py library) is a user-friendly gateway between Python and OMERO server functionalities, which are based on the Internet Communications Engine (ICE)

2. Establish a connection

3. Interact with the OMERO.server

image = **connection_object**.getObject(„Image“, 256)

What can you do with python API

- Transfer images with metadata and ROIs
- Create structures into OMERO (Project/Dataset)
- Fetch Data and push them back
- Annotate data in OMERO
- Organize data in OMERO
- Connect to different instances and even IDR
- Connect to an S3 storage

omero-py documentation (getting started)
https://docs.openmicroscopy.org/omero/5.6.0/developers/Python.html

ezomero

https://github.com/TheJacksonLaboratory/ezomero

A module with convenience functions for writing Python code that interacts with OMERO.

In general, you will need to create a BlitzGateway object using **ezomero.connect()**, then pass the conn object to most of these helper functions along with function-specific parameters.

connection_object = ezomero.connect(user: „username“ , password: „PASSWORD“, host: „host“, port: 4064, secure: True)

ezomero

Really easy to handle and **GREAT** documentation

https://thejacksonlaboratory.github.io/ezomero/index.html

Easy approaches to

- Import your data
- Deal with the file structure
- Filter
- Annotate your data
- Import annotated data

ezomero.ezimport(conn: _BlitzGateway, target: str, project: str | int | None = None, dataset: str | int | None = None, screen: str | int | None = None, ln_s: bool | None = False, ann: dict | None = None, ns: str | None = None)

Arguments:

- conn (omero.gateway.BlitzGateway object.) – OMERO connection.
- target (string) – Path to the import target to be imported into OMERO.
- project (str or int, optional) – The name or ID of the Project data will be imported into.
- dataset (str or int, optional) – The name or ID of the Dataset data will be imported into.
- screen (str or int, optional) – The name or ID of the Screen data will be imported into.
- ln_s (boolean, optional) – Whether to use ln_s softlinking during imports or not.
- ann (dict, optional) – Dictionary with key-value pairs to be added to imported images.
- ns (str, optional) – Namespace for the added key-value pairs.

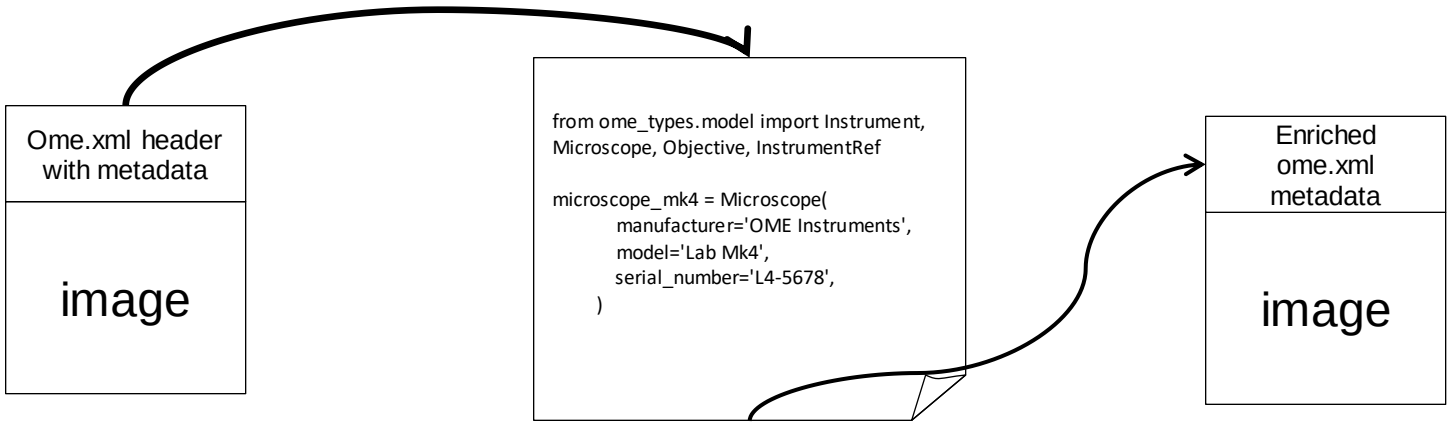
ome-types

https://github.com/tlambert03/ome-types

A pure-python implementation of the OME data model

ome_types provides a set of python dataclasses and utility functions for parsing the OME-XML format into fully-typed python objects for interactive or programmatic access in python.

It can also take these python objects and output them into valid OME-XML. ome_types is a pure python library and does not require a Java virtual machine.



The library allows also to create ome.xml from scratch or add new elements

omero-rois

https://github.com/ome/omero-rois

OMERO Python utilities for handling regions of interest (ROIs)

https://omero-rois.readthedocs.io/en/stable/#

Two interesting functions:

omero_rois.mask_from_binary_image(binim, rgba=None, z=None, c=None, t=None, text=None,raise_on_no_mask=True)

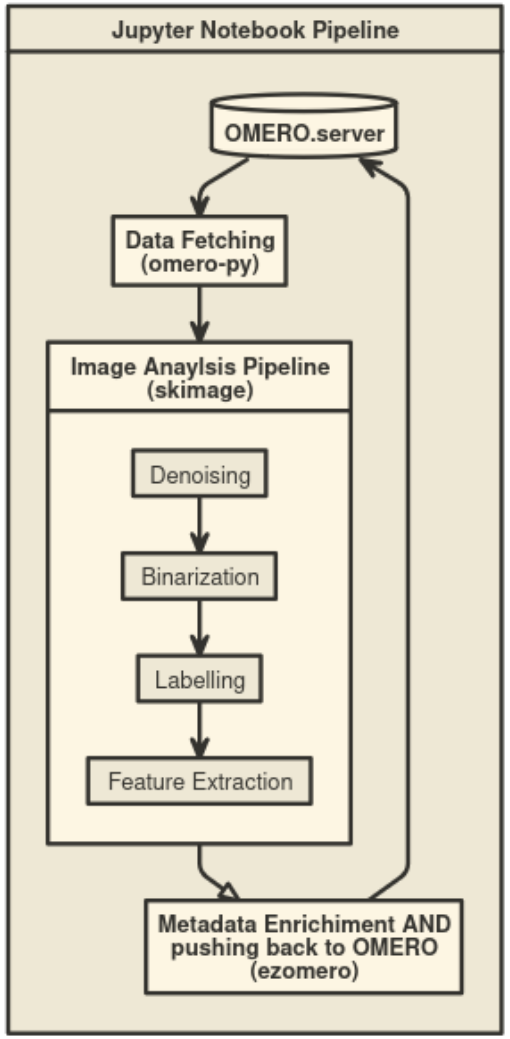
omero_rois.masks_from_label_image(labelim, rgba=None, z=None, c=None, t=None, text=None, raise_on_no_mask=True)

Advantages of Jupyter Notebooks

1. **Programming Practice**
2. **Collaborating Across Projects and Tools**
3. **Allows workflow control**
4. **Data Organization and Cleaning**
5. **Data Visualization and Sharing**
6. **Teaching Data Science Skills**

https://www.nobledesktop.com/classes-near-me/blog/top-uses-for-jupyter-noteboook

Let’s create a pipeline together!



How does it works?

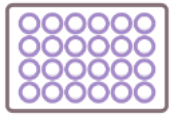
- We code together in three different groups
- Open the JN located in „exercises“
- We will use a mix of ezomero and omero-py to interact with OMERO
- Skimage to perform image analysis

Create your own pipeline

Scenario 1
Monday morning as DaSt

Just a normal monday morning as DaSt. **Upload** your images with correct K/V pairs. **Organize** data in project/dataset. **Add tags** to image A06 and H11 and **run an analysis** on these two. **Push them back**!

Scenario 2
HCS - A full plate analysis



Looks like you are working with a proper HCS experiment. **Loop trough the plate a repeat the analysis for each well**. **Push the results back** and **attach the result to the plate**.

Scenario 3
A messy experiment...

Oh man, there were problems experimental planning… The scientist just upload them in OMERO by adding the tags „Condition1“ and „Condition2“. **Could your run the 2 analysis separately**?

Scenario 4
Riccardo, just let me be

Hej, just code what you want and have fun testing around :)

1) dast_morning.ipynb

2) HCS_plate.ipynb

3) messy_scenario.ipynb

No fix rules, just suggestions..
try to be creative

Before asking:
- Check image.sc
- Check stackoverflow
- AI help is allowed
- Ask the trainer