Introduction to Jupyter notebooks, data organization and plotting

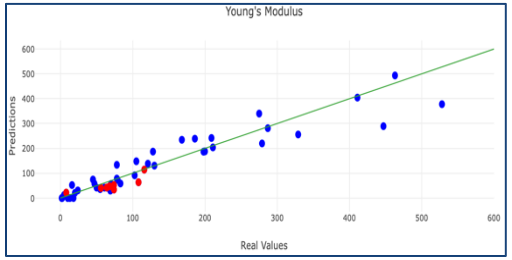
In this tutorial:
- Using a Jupyter notebook in nanoHUB
- Organizing and filtering data using Pandas
- Creating a simple scatter plot using Plotly

Juan C. Verduzco and Alejandro Strachan
jverduzc@purdue.edu || strachan@purdue.edu
School of Materials Engineering & Birck Nanotechnology Center
Purdue University
West Lafayette, Indiana USA

Data Science & Machine Learning in Science & Engineering

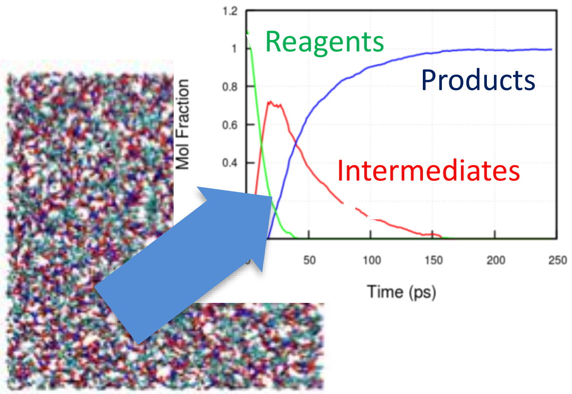
Learning from data

Predictive models

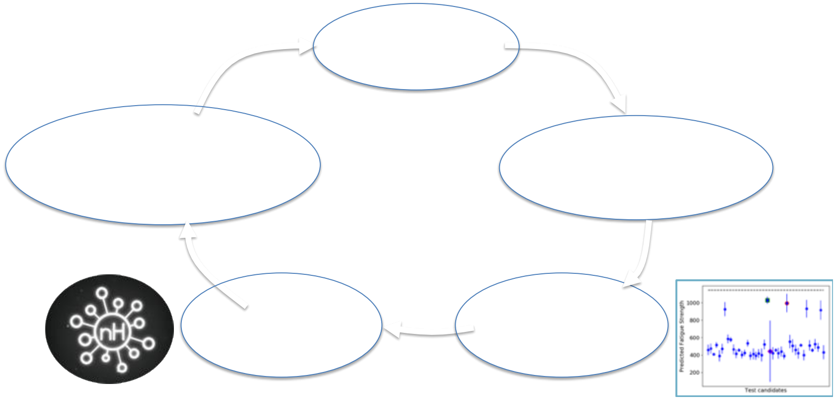


Classification

Dimensionality reduction (Coarse graining & multiscale modeling)



Design of experiments



Cyber-infrastructure

nanoHUB

CITRINE INFORMATICS

The Materials Project

WolframAlpha

KIM

MATERIALS DATA FACILITY

Tutorial outline

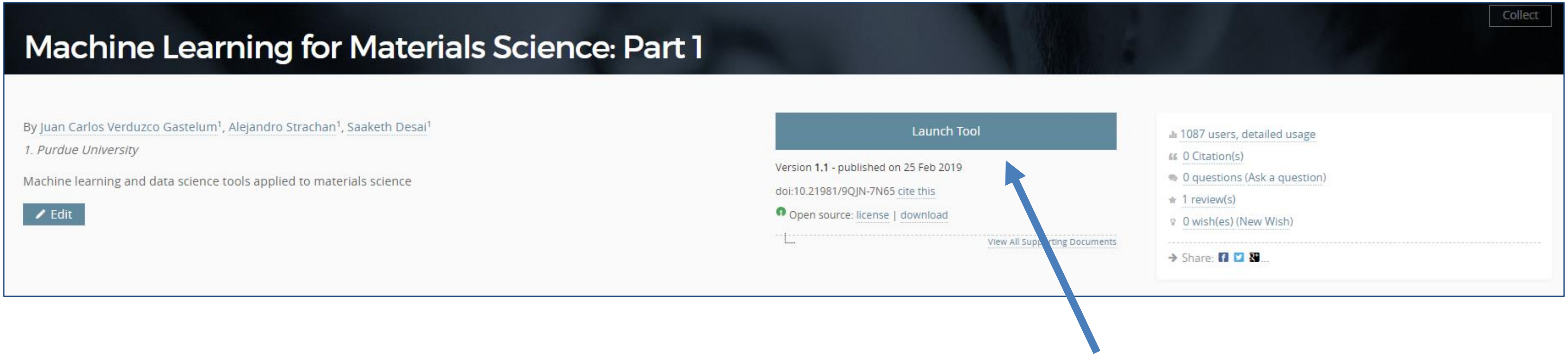
1. Launch a Jupyter notebook in nanoHUB
2. Python 101 using Jupyter
3. Organizing data using Pandas
4. Visualizing data using Plotly
5. Working with Jupyter in nanoHUB
6. Q&A

Let's get our hands dirty – use the subsequent slides to follow along

Launching a Jupyter tool in nanoHUB

Machine Learning for Materials Science: Part 1

From your browser go to link: https://nanohub.org/tools/mseml/



Click on Launch Tool to begin

Landing Page – Notebook: Querying

Navigate to the first link in the landing page, to access the notebook we will be working on during this workshop.

Introduction to Machine Learning for Materials Science

The tutorials here will give you an insight into the usage of Machine Learning to approach problems related to materials science.

- **Get started** Click on the links below to begin each tutorial.
- **Important** To exit individual tutorials and return to this page, use File -> Close and Halt. "Terminate Session" (top right) will kill your entire Jupyter session.

Querying databases, Organizing and Plotting Data:

- Query Pymatgen and Mendeleev for properties like Young's modulus and melting temperature
- Organize data into Pandas dataframes and python dictionaries and plot using Plotly

Linear Regression to predict material properties:

- Perform linear regression using the scikit learn package and predict Young's modulus
- Visualize trends in data and 'goodness of fit' of linear model

Neural Network Regression to predict material properties:

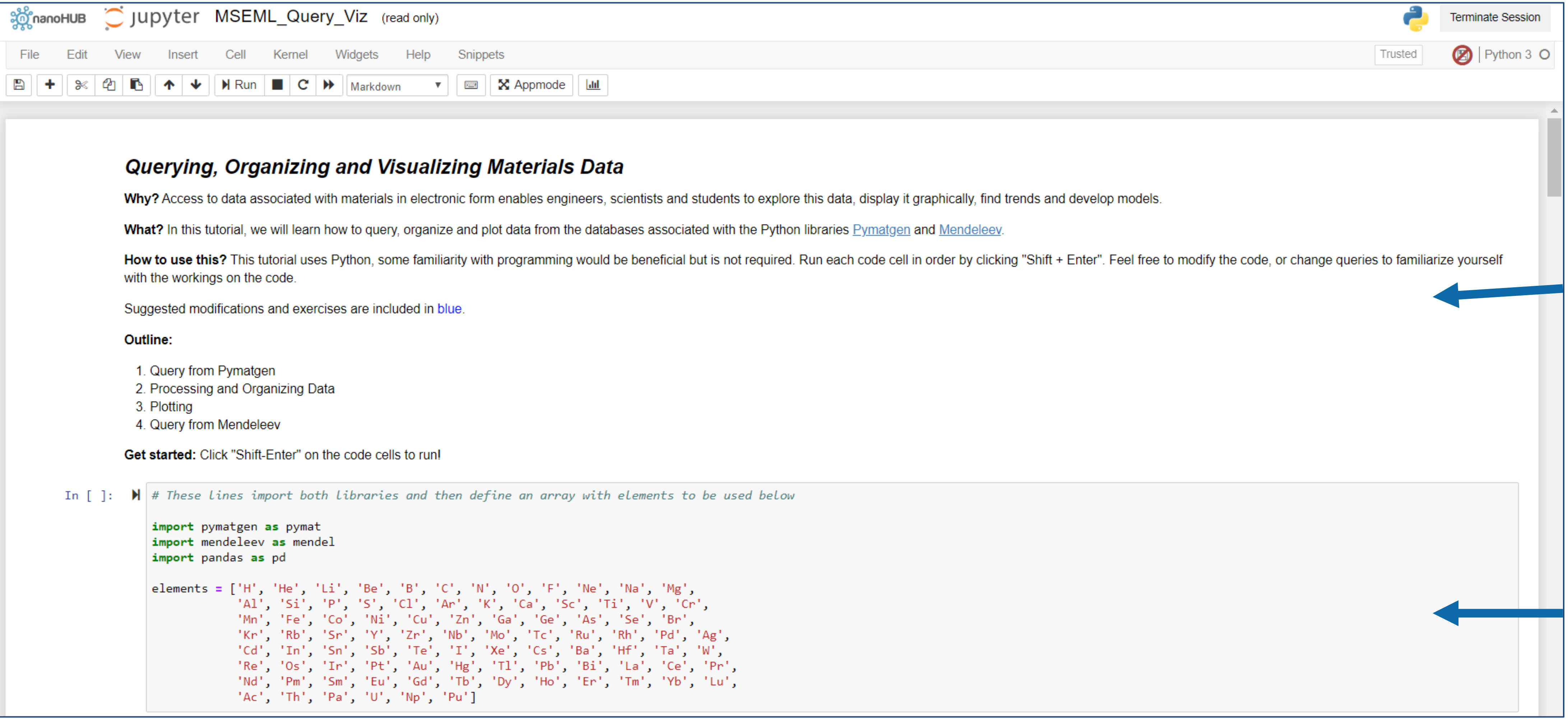
- Use neural networks to perform non-linear, higher order regression
- Visualize trends and compare non-linear model to linear regression

Neural Network Classification to predict crystal structures:

- Use neural networks to classify elements according to their crystal structures

Introduction to Jupyter

Jupyter notebooks combine text using markdown, live code and powerful visualization



Querying, Organizing and Visualizing Materials Data

Why? Access to data associated with materials in electronic form enables engineers, scientists and students to explore this data, display it graphically, find trends and develop models.

What? In this tutorial, we will learn how to query, organize and plot data from the databases associated with the Python libraries Pymatgen and Mendeleev.

How to use this? This tutorial uses Python, some familiarity with programming would be beneficial but is not required. Run each code cell in order by clicking "Shift + Enter". Feel free to modify the code, or change queries to familiarize yourself with the workings on the code.

Suggested modifications and exercises are included in blue.

Outline:

1. Query from Pymatgen
2. Processing and Organizing Data
3. Plotting
4. Query from Mendeleev

Get started: Click "Shift-Enter" on the code cells to run!

In [ ]:

```
# These lines import both libraries and then define an array with elements to be used below

import pymatgen as pymat
import mendeleev as mendel
import pandas as pd

elements = ['H', 'He', 'Li', 'Be', 'B', 'C', 'N', 'O', 'F', 'Ne', 'Na', 'Mg',
            'Al', 'Si', 'P', 'S', 'Cl', 'Ar', 'K', 'Ca', 'Sc', 'Ti', 'V', 'Cr',
            'Mn', 'Fe', 'Co', 'Ni', 'Cu', 'Zn', 'Ga', 'Ge', 'As', 'Se', 'Br',
            'Kr', 'Rb', 'Sr', 'Y', 'Zr', 'Nb', 'Mo', 'Tc', 'Ru', 'Rh', 'Pd', 'Ag',
            'Cd', 'In', 'Sn', 'Sb', 'Te', 'I', 'Xe', 'Cs', 'Ba', 'Hf', 'Ta', 'W',
            'Re', 'Os', 'Ir', 'Pt', 'Au', 'Hg', 'Tl', 'Pb', 'Bi', 'La', 'Ce', 'Pr',
            'Nd', 'Pm', 'Sm', 'Eu', 'Gd', 'Tb', 'Dy', 'Ho', 'Er', 'Tm', 'Yb', 'Lu',
            'Ac', 'Th', 'Pa', 'U', 'Np', 'Pu']
```

Markdown cells

"Shift + Enter" to run cell

Code cells

Let's get some data

Learn more about repositories and queries:

Data can be queried or uploaded

2. **Repositories and data management**
   1. Leader: Zachary McClure
   2. Date/Time–Friday, 10th April 2020 / 11 AM – 12 PM EDT
   3. Topics covered:
      - Introduction to repository APIs
      - Querying and advanced plotting

Query Pymatgen and Mendeleev, two libraries with materials properties

Run the cells sequentially for this exercise.

```
In [ ]:  querable_pymatgen = ["atomic_mass", "poissons_ratio","atomic_radius", "electrical_resistivity",
                              "molar_volume","thermal_conductivity", "bulk_modulus", "youngs_modulus",
                             "brinell_hardness", "average_ionic_radius", "melting_point", "rigidity_modulus",
                              "density_of_solid","coefficient_of_linear_thermal_expansion"]

         sample = ['Fe', 'Co', 'Ni', 'Cu', 'Zn']

         for item in sample:
             element_object = pymat.Element(item)
             print(item, element_object.youngs_modulus) # You can change "youngs_modulus" to any of the
                                                        # properties in the querable_pymatgen list

         #for item in sample:
         #    for i in querable_pymatgen:
         #        element_object = pymat.Element(item)
         #        print(item, i, getattr(element_object,i))
```

Change ”youngs_modulus” to “atomic_mass” and re-run

“Shift + Enter” to run cell

Organizing data: python dictionaries

Dictionaries =

key-value pairs

Access to the entire entry, or specific attributes.

```
In [ ]:  Fe_data = {} # Initializing a dictionary

         # Each of the following lines is making a single entry

         Fe_data["atomic_number"] = mendel.element("Fe").atomic_number
         Fe_data["coefficient_of_linear_thermal_expansion"] = pymat.Element("Fe").coefficient_of_linear_thermal_expansion
         Fe_data["youngs_modulus"] = pymat.Element("Fe").youngs_modulus
         Fe_data["specific_heat"] = mendel.element("Fe").specific_heat

         #Print the entire entry for Fe
         print(Fe_data)

         #Print a specific attribute:
         print(Fe_data["specific_heat"])

         # This line is to delete an entry
             # del Fe_data["atomic_number"]
```

Learn more at: https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Organizing data : python lists

Access elements by index number instead of their properties.

Add new information using the .append() method.

```
In [ ]:
sample = elements.copy()

CTE = [] # In this list we will store the Coefficients of Thermal Expansion
youngs_modulus = [] # In this list we will store the Young's Moduli
melting_temperature = [] # In this list we will store the Melting Temperatures

for item in sample:
    CTE.append(pymat.Element(item).coefficient_of_linear_thermal_expansion)
    youngs_modulus.append(pymat.Element(item).youngs_modulus)
    melting_temperature.append(pymat.Element(item).melting_point)

# You can visualize the lists by uncommenting these print statements
#print(CTE)
#print(youngs_modulus)
#print(melting_temperature)
```

Learn more at: https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

Organizing data: Pandas dataframes

Created for data analysis in data science.

Advantages:

- Performing operations, sorts and filters.
- Working with non-numeric data
- Handling of large datasets

```
In [ ]:
all_values = [] # Values for Attributes

for item in fcc_elements:
    element_values = []

    element_object = pymat.Element(item)
    for i in querable_pymatgen:
        element_values.append(getattr(element_object,i))

    all_values.append(element_values) # All lists are appended to another list, creating a list of lists

# Pandas Dataframe
df = pd.DataFrame(all_values, columns=querable_pymatgen)
display(df)
```

	atomic_mass	poissons_ratio	atomic_radius	electrical_resistivity	molar_volume	thermal_conductivity	bulk_modulus
Ag	107.868200	0.37	1.60	1.630000e-08	10.27	430.0	100.0
Al	26.981539	0.35	1.25	2.700000e-08	10.00	235.0	76.0
Au	196.966569	0.44	1.35	2.200000e-08	10.21	320.0	220.0
Cu	63.546000	0.34	1.35	1.720000e-08	7.11	400.0	140.0
Ir	192.217000	0.26	1.35	4.700000e-08	8.52	150.0	320.0
Ni	58.693400	0.31	1.35	7.200000e-08	6.59	91.0	180.0
Pb	207.200000	0.44	1.80	2.100000e-07	18.26	35.0	46.0
Pd	106.420000	0.39	1.40	1.080000e-07	8.56	72.0	180.0
Pt	195.084000	0.38	1.35	1.060000e-07	9.09	72.0	230.0
Rh	102.905500	0.26	1.35	4.300000e-08	8.28	150.0	380.0
Sr	87.620000	0.28	2.00	1.350000e-07	33.94	35.0	NaN
Th	232.038060	0.27	1.80	1.500000e-07	19.80	54.0	54.0
Yb	173.040000	0.21	1.75	2.500000e-07	24.84	39.0	31.0

Learn more at: https://pandas.pydata.org/docs/getting_started/index.html

Organizing data: Pandas operations

One-line operations:

- Reindexing
- Filtering

```
In [ ]:  df.index = fcc_elements
         display(df)

In [ ]:  df_big_atoms = df[df.atomic_mass.ge(150)]
         display(df_big_atoms)

In [ ]:  df_poisson = df[df.poissons_ratio.eq(0.26)]
         display(df_poisson)

In [ ]:  df_condition = df[(df['youngs_modulus'] < 120) & (df["poissons_ratio"] > 0.25)]
         display(df_condition)
```

Binary filters and custom conditions filter the dataframe.

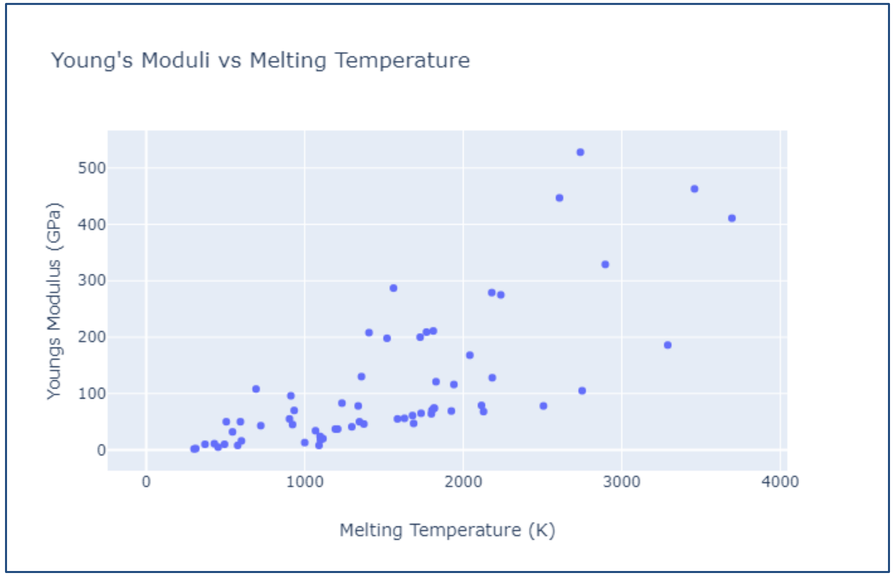
Standard binary operators are:

.eq() .neq() .ge() .le()

Try filtering our data to get elements with a Poisson's ratio different than 1.35.

Learn more at: https://pandas.pydata.org/pandas-docs/version/0.24.2/reference/frame.html

Simple plot using Plotly



Create interactive plots using Plotly.

Interactive and publication quality plots.

Make a plot from your data with as little as 5 lines of code.

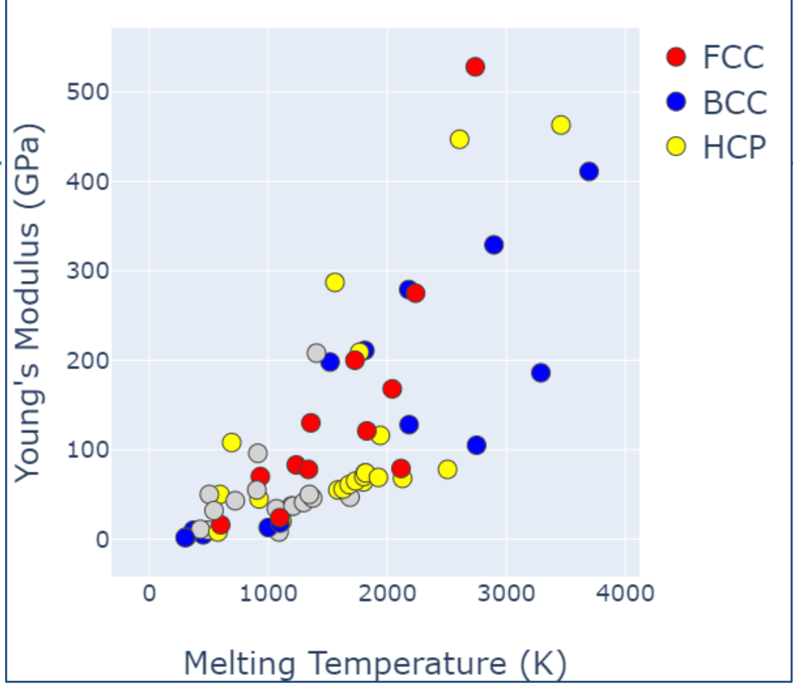
```
In [ ]:
import plotly #This is the library import
import plotly.graph_objs as go # This is the graphical object (Think "plt" in Matplotlib if you have used that before)

from plotly.offline import iplot # These lines are necessary to run Plotly in Jupyter Notebooks, but not in a dedicated environment
plotly.offline.init_notebook_mode(connected=True)

# To create a plot, you need a layout and a trace

# The layout gives Plotly the instructions on the background grids, tiles in the plot,
# axes names, axes ticks, legends, labels, colors on the figure and general formatting.

layout = go.Layout(title = "Young's Moduli vs Melting Temperature",xaxis= dict(title= 'Melting Temperature (K)'),
                   yaxis= dict(title= 'Youngs Modulus (GPa)'))

# The trace contains a type of plot (In this case, Scatter, but it can be "Bars, Lines, Pie Charts", etc.),
# the data we want to visualize and the way ("Mode") we want to represent it.

trace = go.Scatter(x = melting_temperature, y = youngs_modulus, mode = 'markers')

# To plot, we create a figure and implement our components in the following way:

data = [trace] # We could include more than just one trace here

fig= go.Figure(data, layout=layout)
iplot(fig)
```

Learn more at: https://plotly.com/python/

Custom plot using Plotly

Create a custom plot, colored dynamically from the data.

- Modify the label sizes and fonts
- Add legends and grids
- Change the figure dimensions



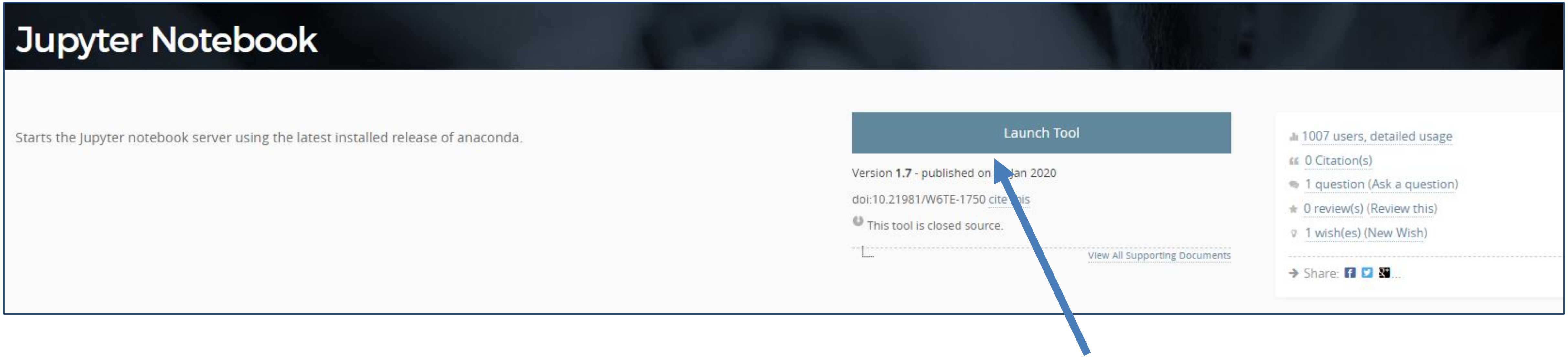
Change a color or a marker symbol

```
In [ ]:
layout0= go.Layout(hovermode= 'closest', width = 600, height=600, showlegend=True,
    xaxis= dict(title=go.layout.xaxis.Title(text='Melting Temperature (K)', font=dict(size=24)), zeroline= False, gridwidth= 1, tickfont=dict(size=18)),
    yaxis= dict(title=go.layout.yaxis.Title(text="Young's Modulus (GPa)", font=dict(size=24)), zeroline= False, gridwidth= 1, tickfont=dict(size=18)),
    legend=dict(font=dict(size=24))) # Adding a legend

# Trace

trace0 = go.Scatter(x = melting_temperature,y = youngs_modulus, mode = 'markers',
    marker= dict(size= 14, line= dict(width=1), color=colors), # We add a size, a border and our custom colors to the markers
    text= sample, # This attribute (Text) labels each point to this list, which contains our elements in the same indexes as our properties
showlegend = False)


# Empty Traces for Legend
legend_plot_FCC = go.Scatter(x=[None], y=[None], mode='markers', marker=dict(size=14,  line= dict(width=1),color='red'), name = 'FCC')
legend_plot_BCC = go.Scatter(x=[None], y=[None], mode='markers', marker=dict(size=14,  line= dict(width=1),color='blue'), name = 'BCC')
legend_plot_HCP = go.Scatter(x=[None], y=[None], mode='markers', marker=dict(size=14,  line= dict(width=1),color='yellow'), name = 'HCP')


data = [trace0, legend_plot_FCC, legend_plot_BCC, legend_plot_HCP]

fig= go.Figure(data, layout=layout0)
iplot(fig)
```

Learn more at: https://plotly.com/python/

Start your own Jupyter notebook

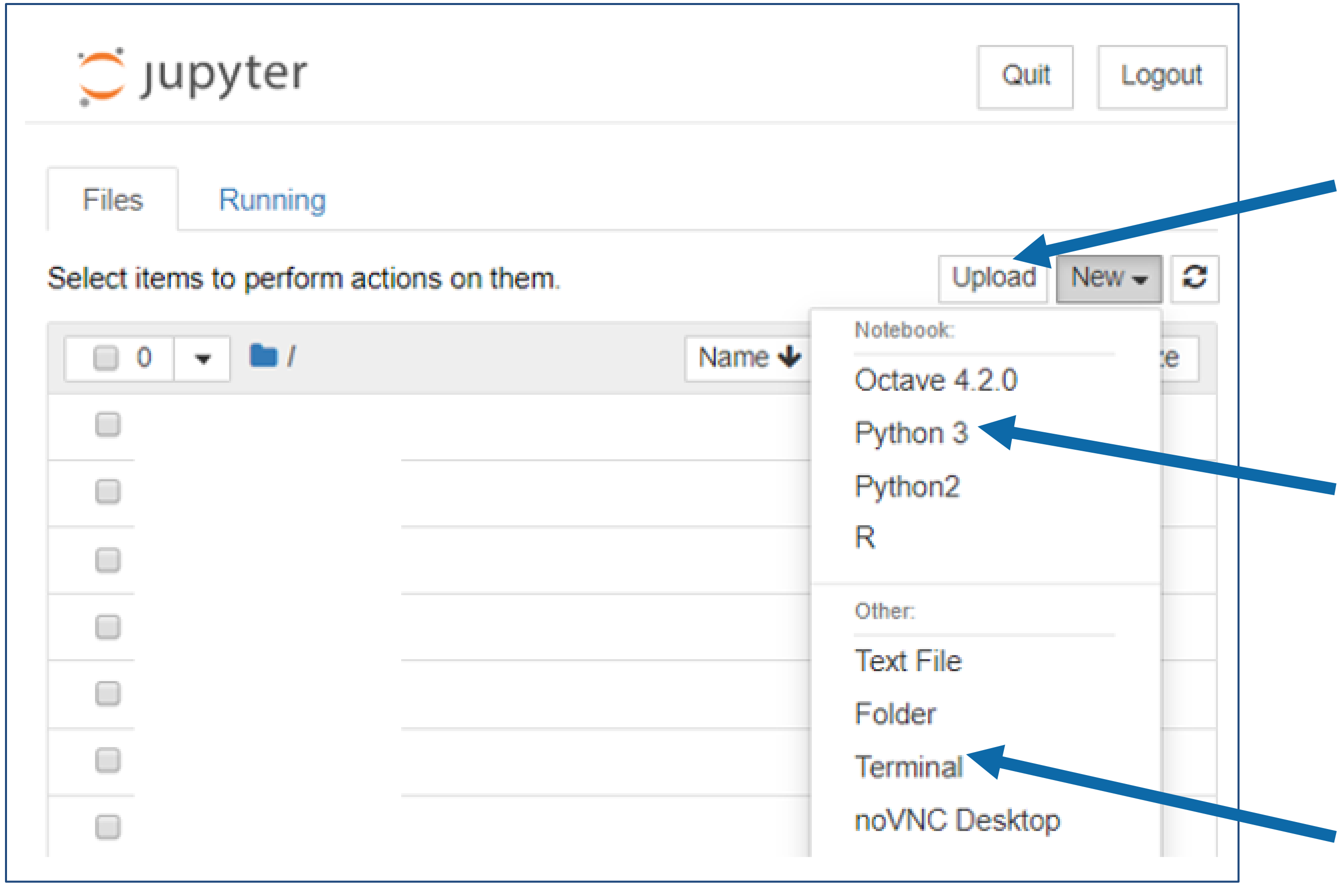
From your browser go to link: https://nanohub.org/tools/jupyter



Click on Launch Tool to begin

Home directory

All Jupyter notebooks created will be stored locally in your home directory.

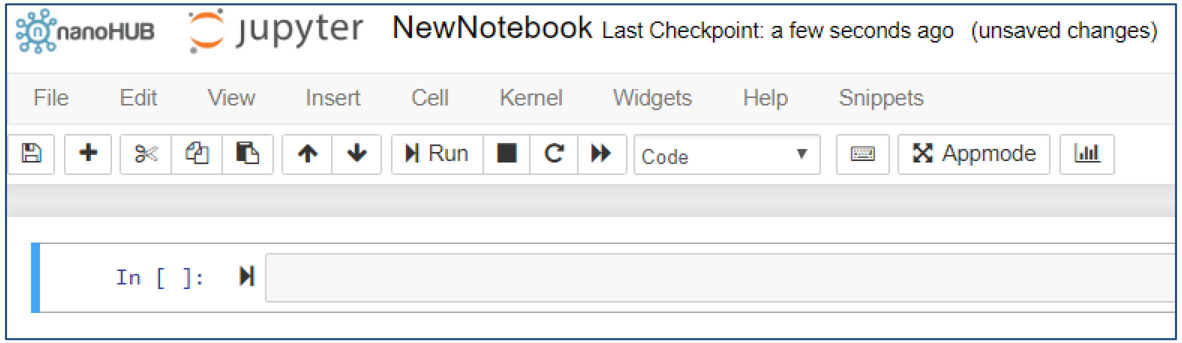


Upload files with data from your computer to use in your notebooks.

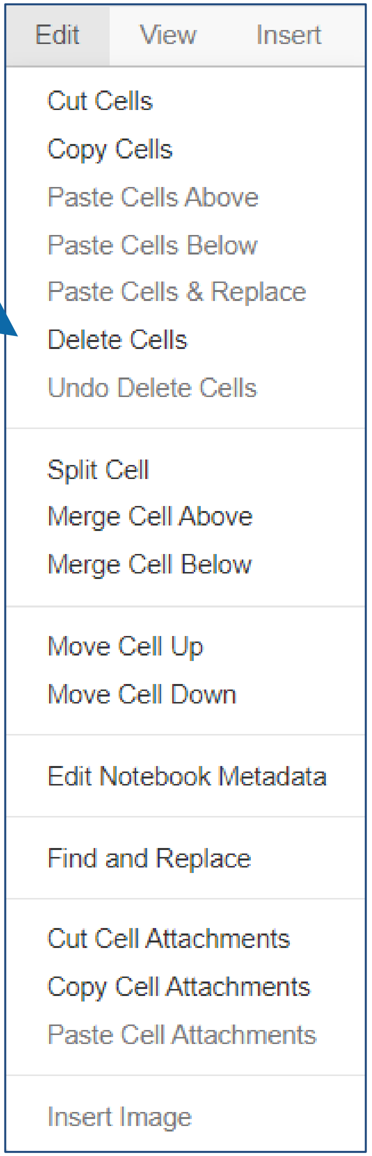
From New, get a new notebook in Python 3.

For UNIX environments, launch a terminal from here.

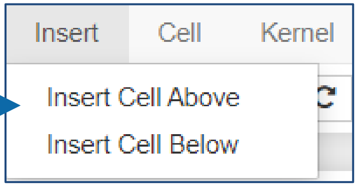
Working on a New Notebook



Delete a cell. Also undo the deletion.

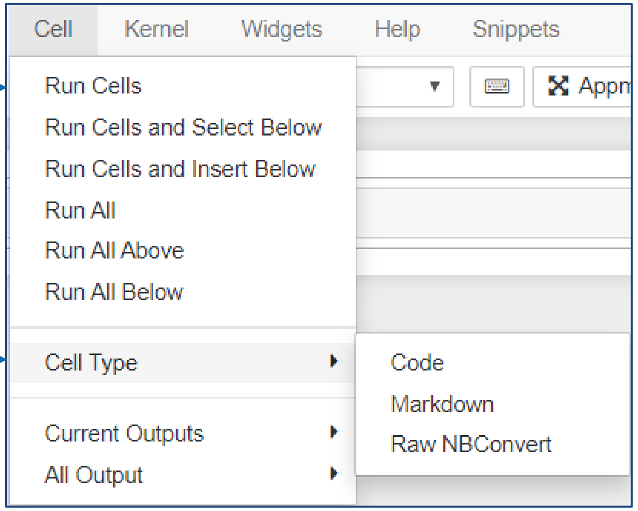


Create a new cell.



Run an individual cell.

Change the type of your cells from Markdowns (texts) to Code cells.



Markdown

```
# Heading 1
## Heading 2
### Heading 3
#### Heading 4

[Hyperlinks](website)

A numbered list:
1. One
2. Two
3. Three

An unordered list:
- One
- Two
- Three

HTML Codings for formatting

<b> bold </b><br>
<i> italics </i><br>
<font color='red' size = 6> Different colors and sizes </font><br>
```

Heading 1

Heading 2

Heading 3

Heading 4

Hyperlinks

A numbered list:

1. One
2. Two
3. Three

An unordered list:

- One
- Two
- Three

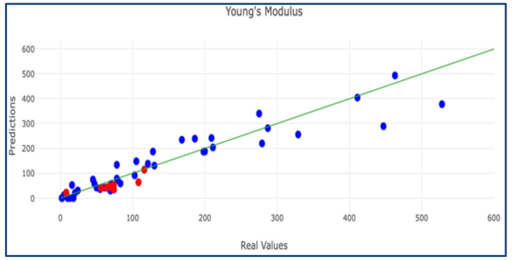
HTML Codings for formatting

bold
italics
Different colors and sizes

Data Science & Machine Learning in Science & Engineering

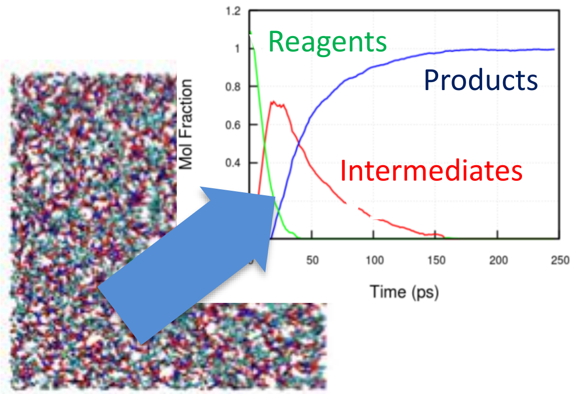
Learning from data

Predictive models

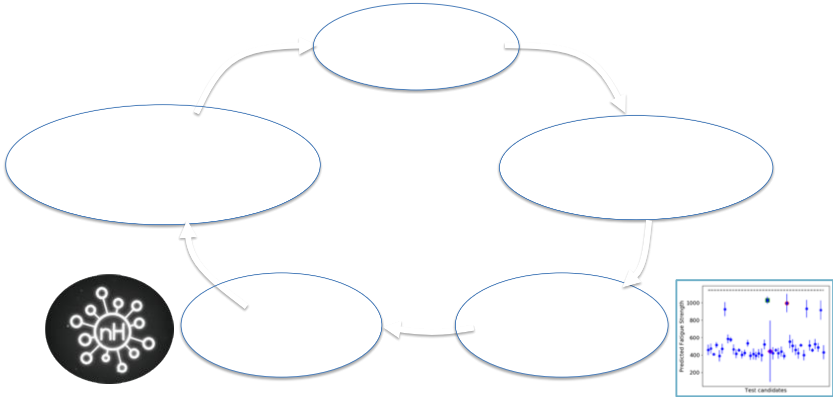


Classification

Dimensionality reduction (Coarse graining & multiscale modeling)



Design of experiments



Cyber-infrastructure

nanoHUB

CITRINE INFORMATICS

The Materials Project

WolframAlpha

KIM

MATERIALS DATA FACILITY

Q&A