Python in Science and Engineering

Vasyl Vovk, Consultant, Globallogic

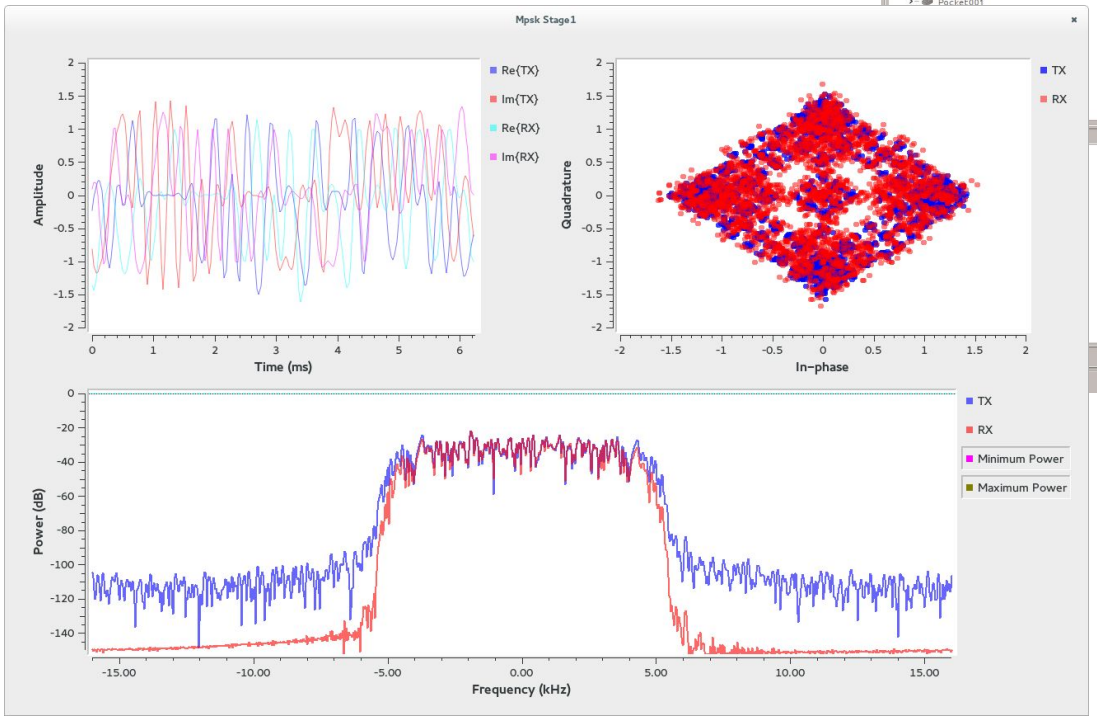
Agenda

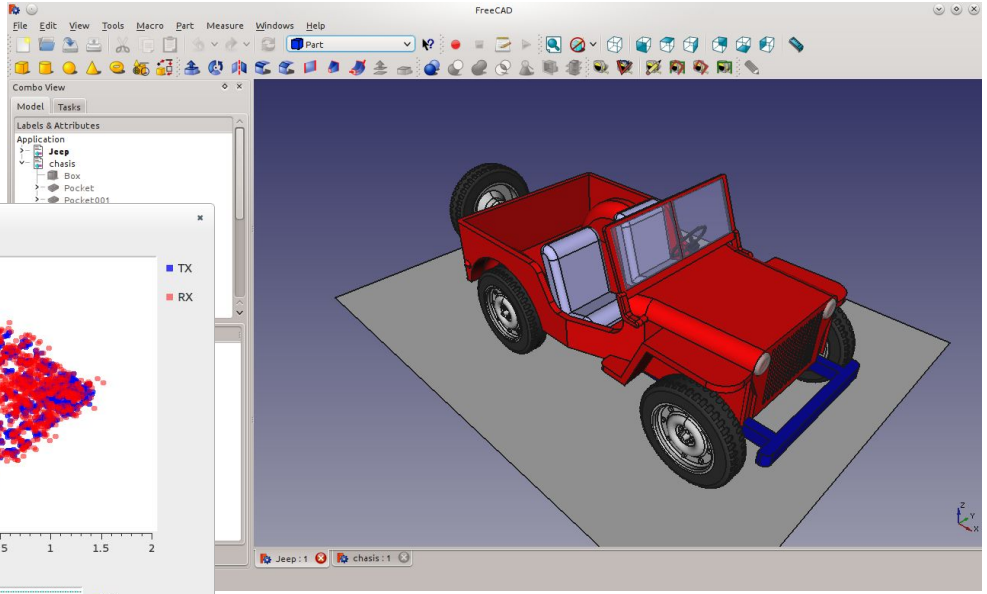
1. Topic overview;
2. Super powerful Calculator;
3. Python Libraries for Science and Engineering:
    - 3.1. Sympy;
    - 3.2. Numpy + Matplotlib;
    - 3.3. SciPy;
    - 3.4. ObsPy;
    - 3.5. AstroPy;
    - 3.6. PyEphem;
    - 3.7. PyRAF.

GlobalLogic

Topic overview

GNU Radio



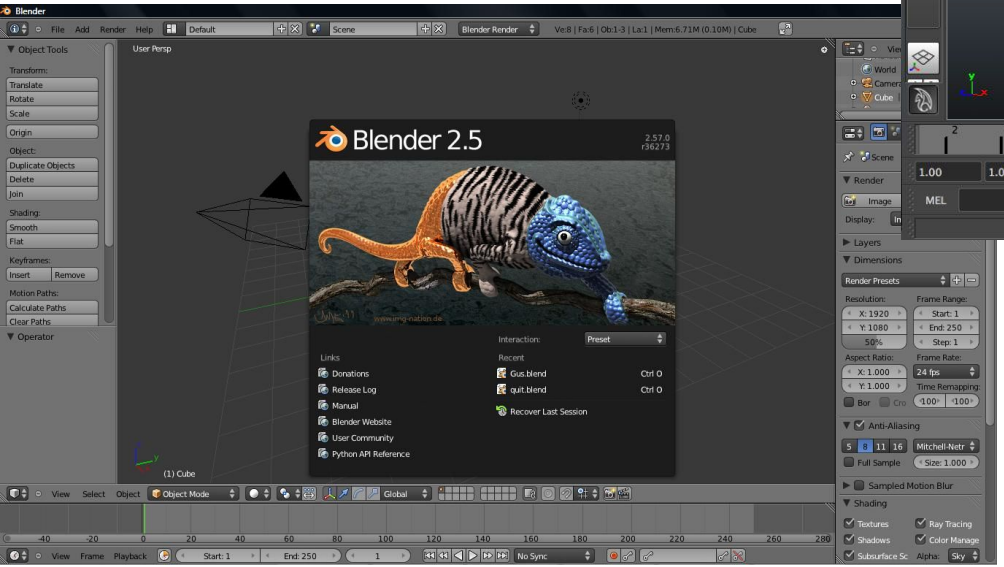


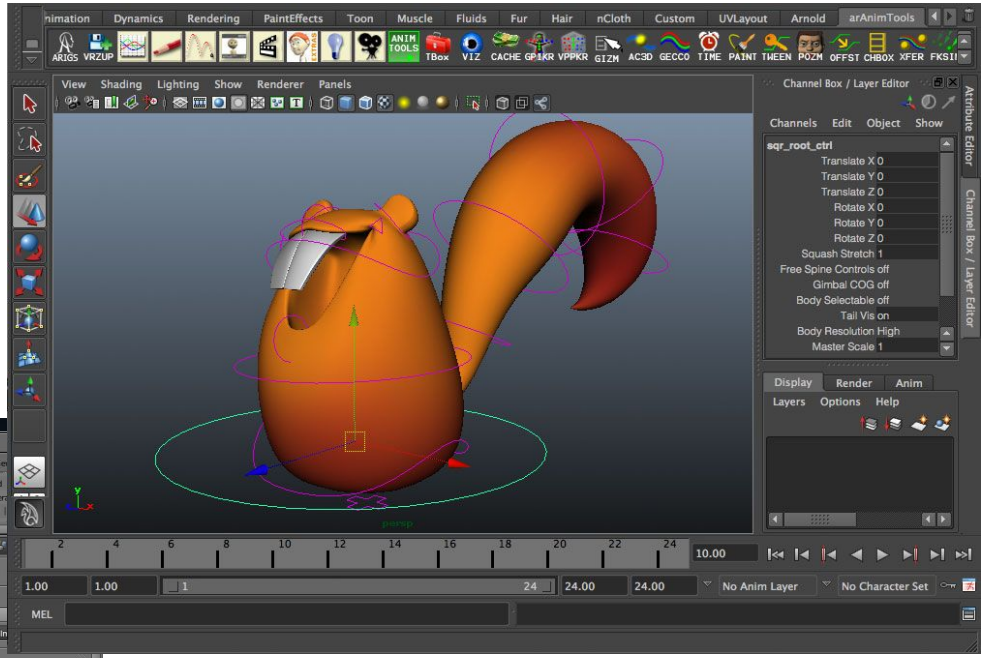
FreeCAD

GlobalLogic

Computer Graphics

Blender

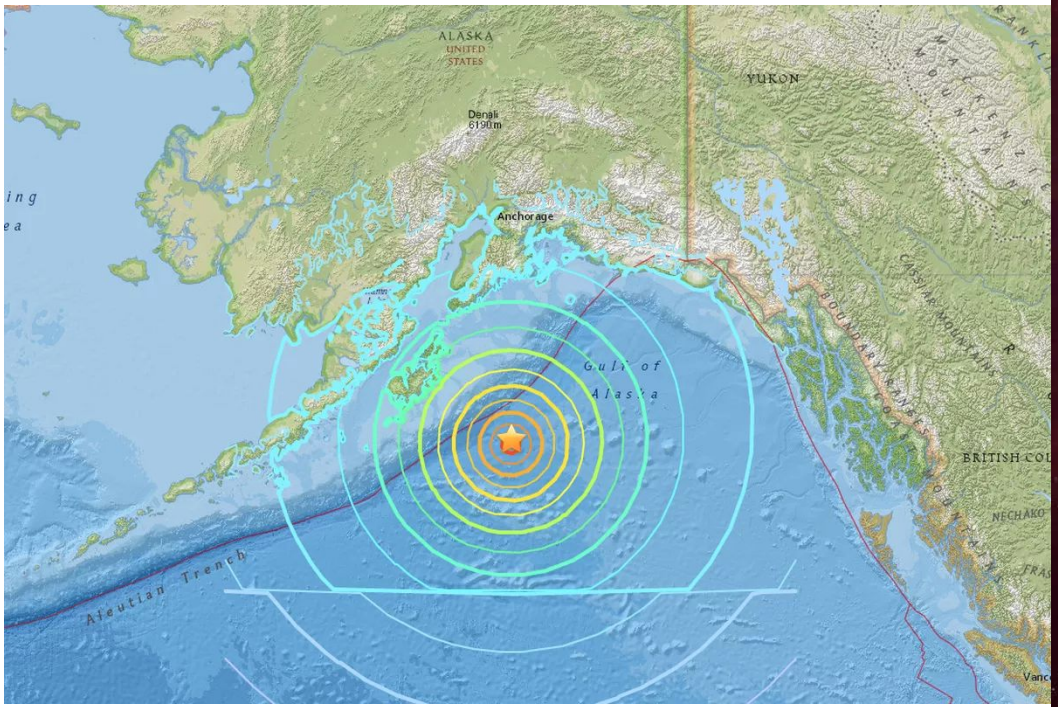




Autodesk Maya

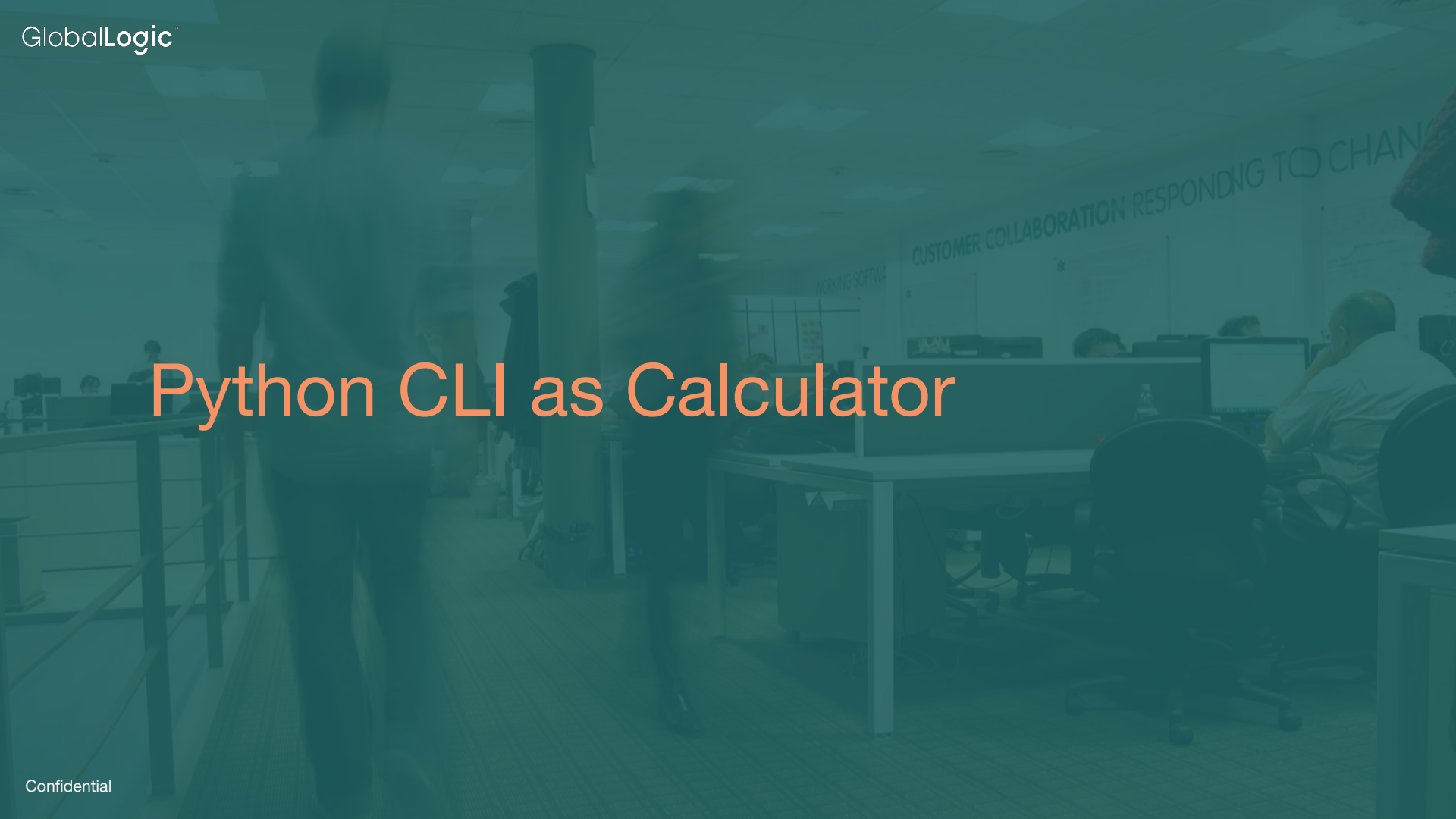
GlobalLogic

Science



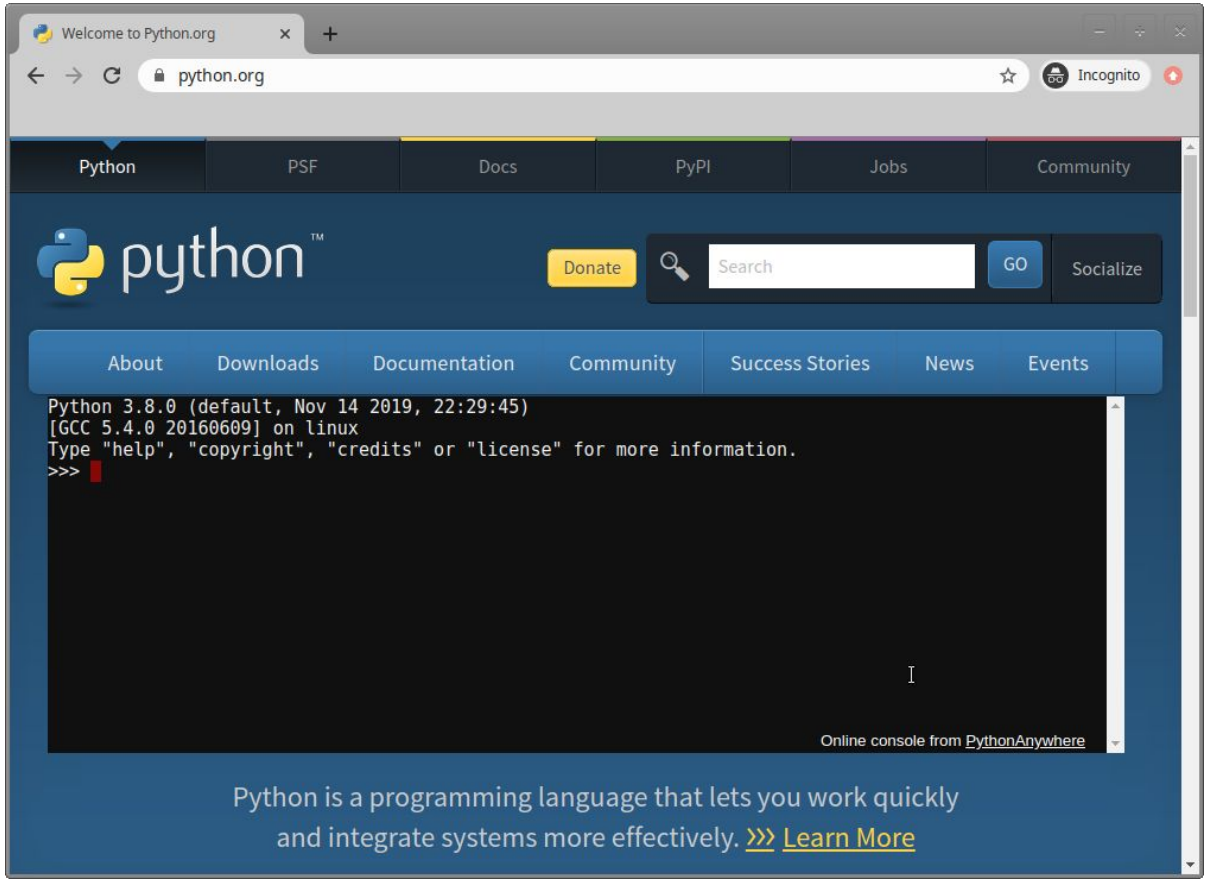
GlobalLogic

Python CLI as Calculator



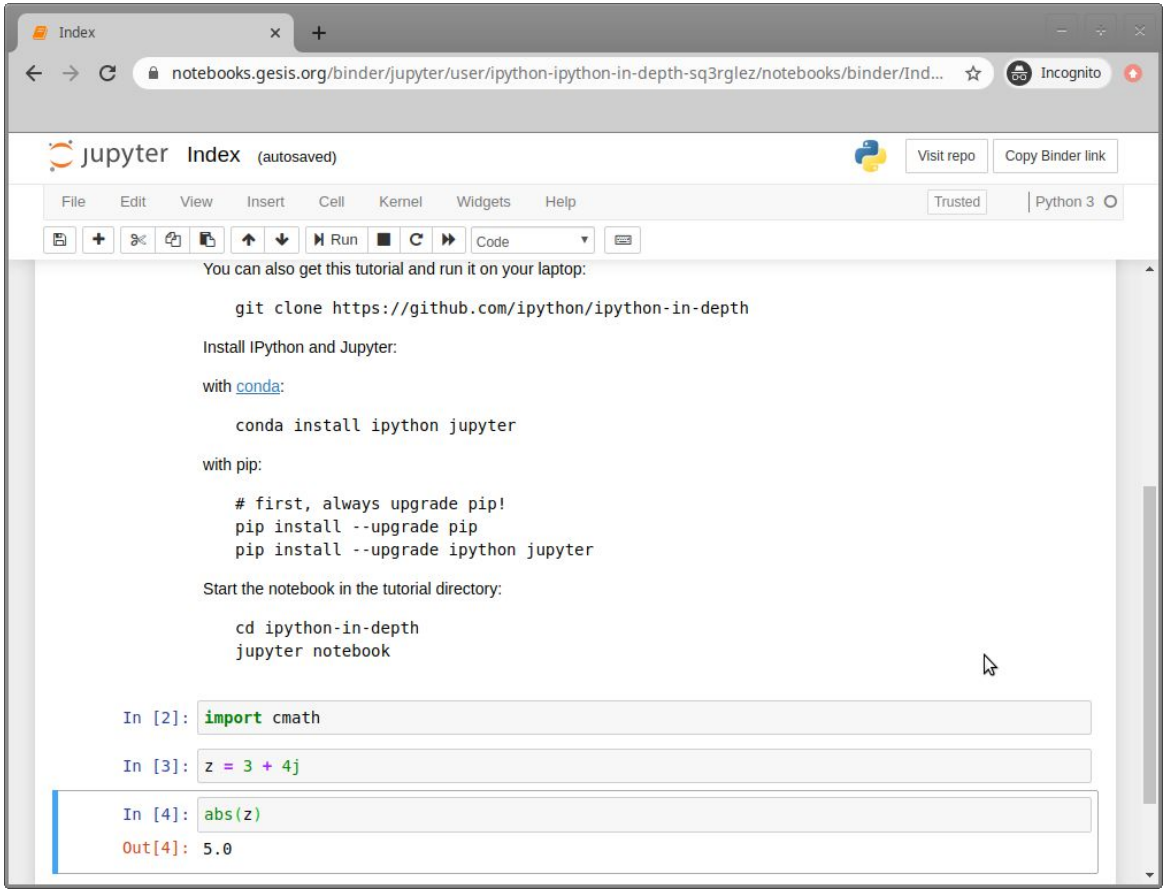
Python CLI as Calculator

https://python.org



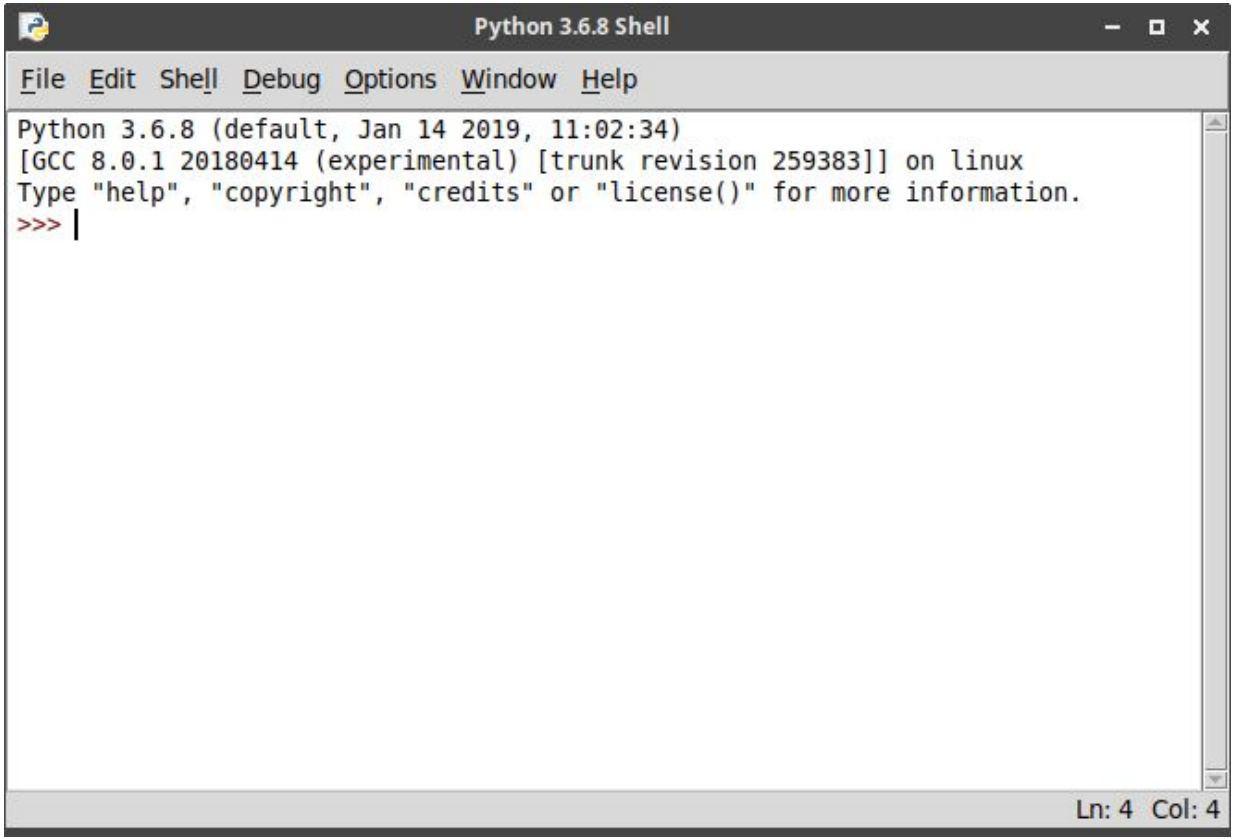
Python CLI as Calculator

https://jupyter.org/



Python CLI as Calculator

Python idle



GlobalLogic

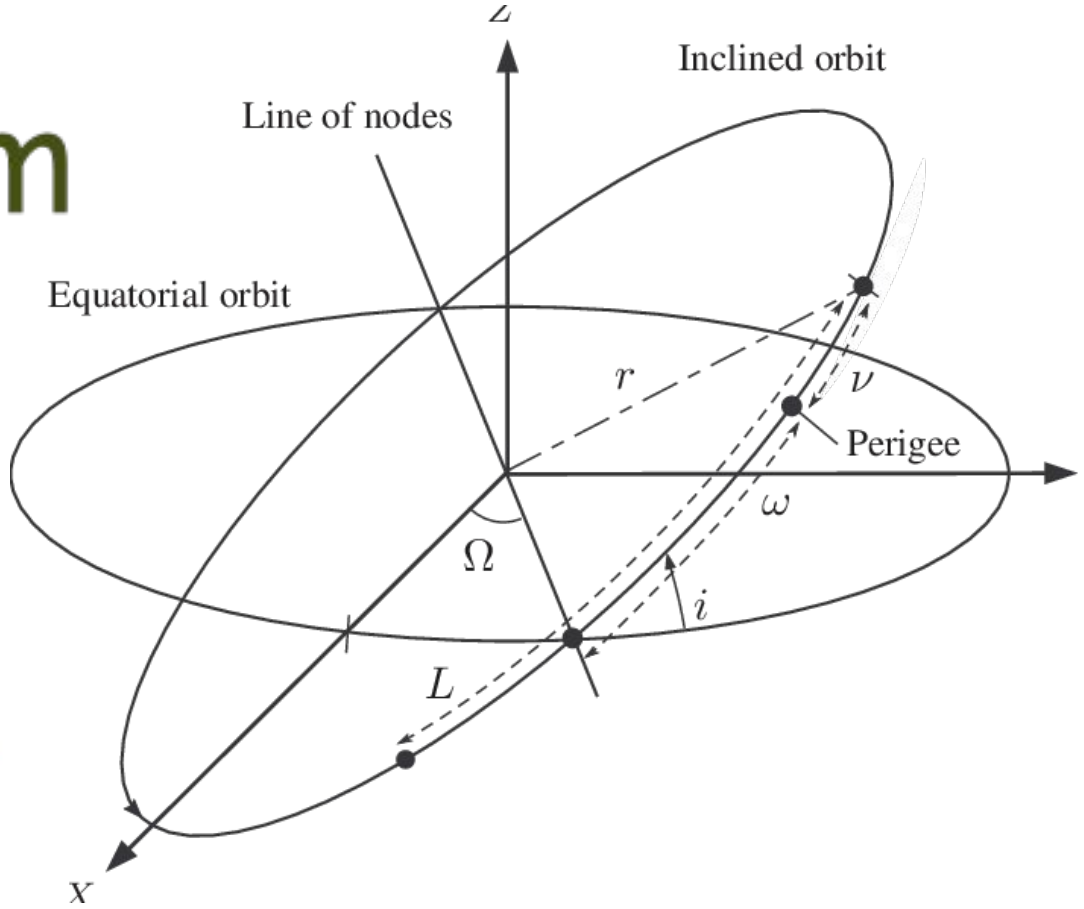
Python Libraries for Science and Engineering


I R A F
Image Reduction and Analysis Facility

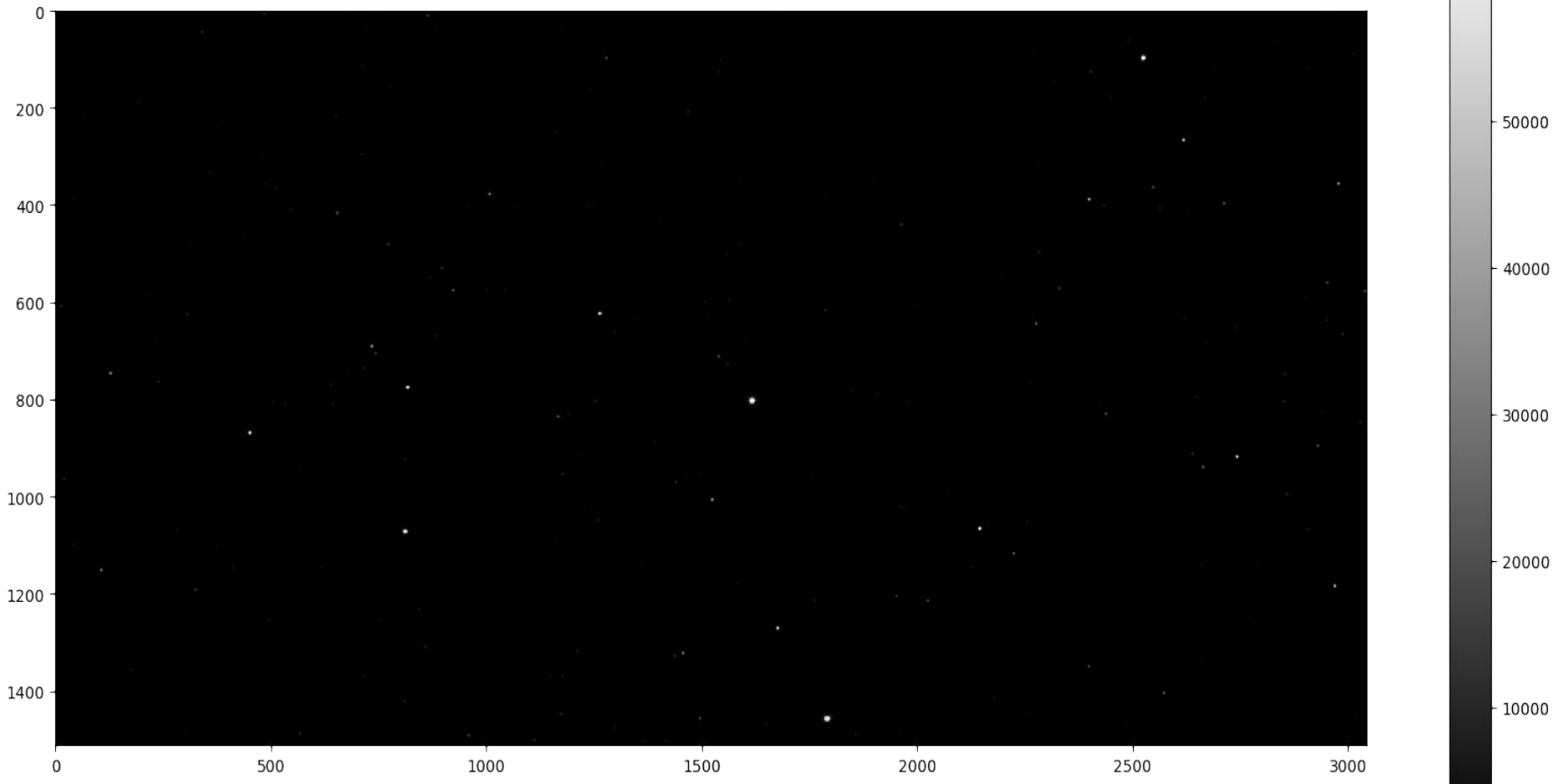

IRAF


PyRAF

PyEphem

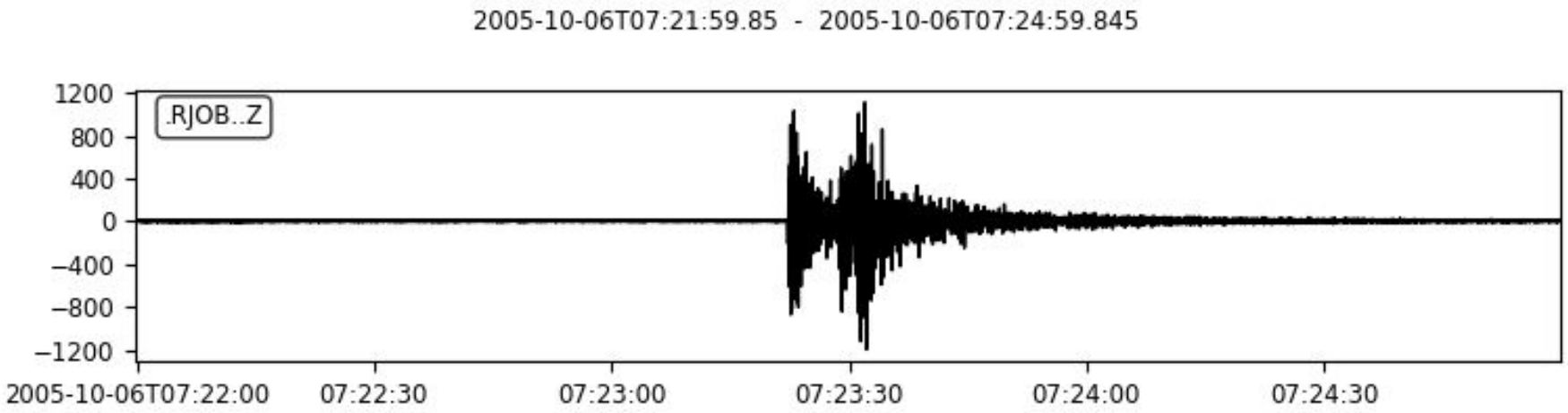

Z
Inclined orbit
Line of nodes
Equatorial orbit
r
ν
Perigee
ω
Ω
i
L
X

```
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from astropy.utils.data import download_file
#   Take a shot which was taken in Mykolaiv Astrom=nomical Observatory
# from open DataBase UKR Virtual Observatory
fit_url = 'http://nao.db.ukr-vo.org/MBT_F/2013/0213/R2015253K.fits.gz'
image_file = download_file(fit_url, cache=True)

hdu_list = fits.open(image_file)
print(hdu_list.info())

image_data = hdu_list[0].data

print(type(image_data))
print(image_data.shape)

hdu_list.close()

image_data = fits.getdata(image_file)
print(type(image_data))
print(image_data.shape)

plt.imshow(image_data, cmap='gray')
plt.colorbar()
plt.show()
```



```
from obspy import read

st = read(

'http://examples.obspy.org/RJOB_061005_072159.ehz.new'
)
print(st)

tr = st[0]  # assign first and only trace to new
variable
print(tr)
print(tr.stats)
tr.stats.station
tr.stats.gse2.datatype

tr.data
tr.data[0:3]

st.plot()
```

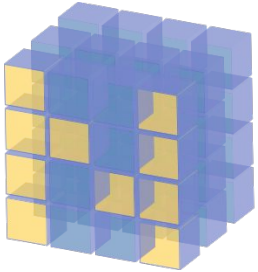
```
1 Trace(s) in Stream:
.RJOB..Z | 2005-10-06T07:21:59.850000Z -
2005-10-06T07:24:59.845000Z | 200.0 Hz, 36000 samples
.RJOB..Z | 2005-10-06T07:21:59.850000Z -
2005-10-06T07:24:59.845000Z | 200.0 Hz, 36000 samples
         network:
         station: RJOB
        location:
         channel: Z
       starttime: 2005-10-06T07:21:59.850000Z
         endtime: 2005-10-06T07:24:59.845000Z
   sampling_rate: 200.0
           delta: 0.005
            npts: 36000
           calib: 0.0949
         _format: GSE2
            gse2: AttribDict({'auxid': 'RJOB',
'datatype': 'CM6', 'calper': 1.0, 'instype': '', 'hang':
-1.0, 'vang': -1.0, 'lat': -999.0, 'lon': -999.0,
'coordsys': '', 'elev': -0.999, 'edepth': -0.999})
```



SciPy

All needed Numerical Methods for Physics research

SciPy =

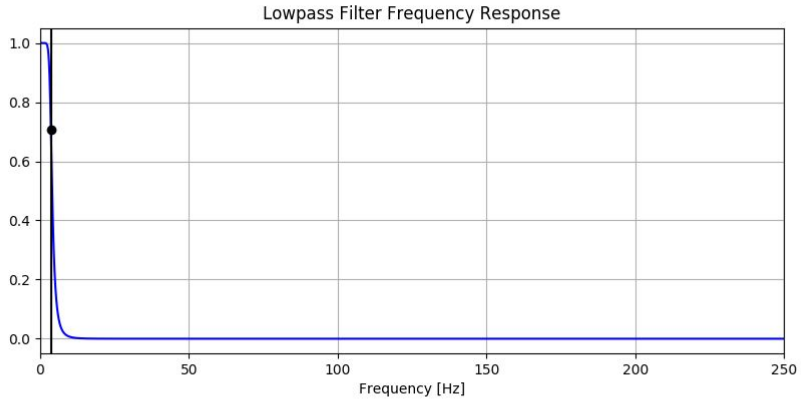
=  NumPy +

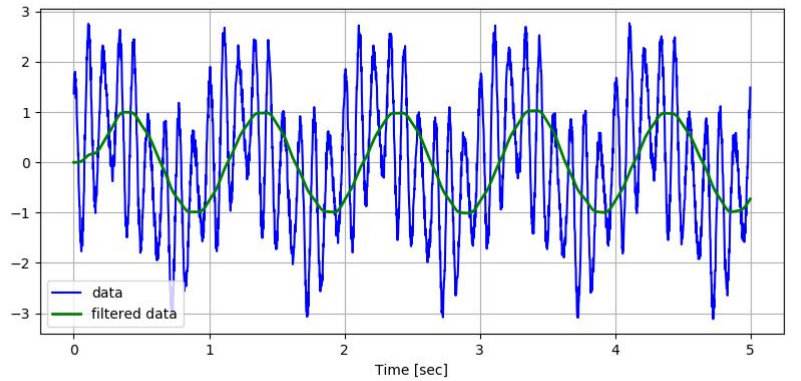
+

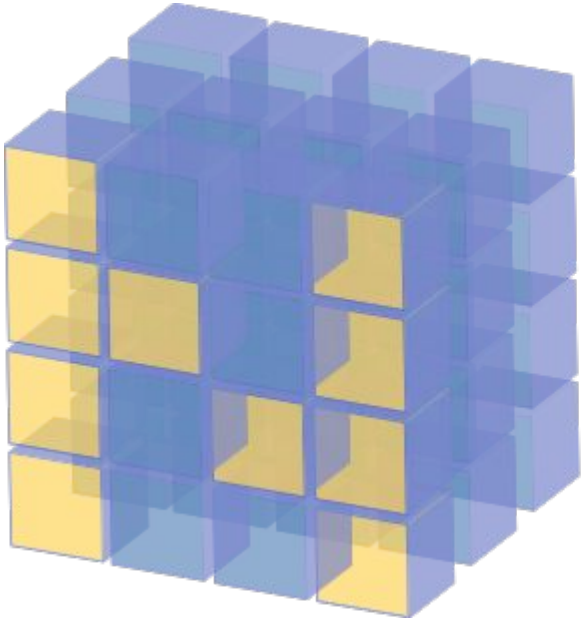
Physical Consts and extended algorithms for Physics research

Butterworth filter

```
import numpy as np
from scipy.signal import butter, lfilter, freqz, firwin
import matplotlib.pyplot as plt


def butter_lowpass(cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff,btype='low',
analog=False)
    return b, a

def butter_lowpass_filter(data, cutoff, fs, order=5):
    b, a = butter_lowpass(cutoff, fs,order=order)
    y = lfilter(b, a, data)
    return y
```







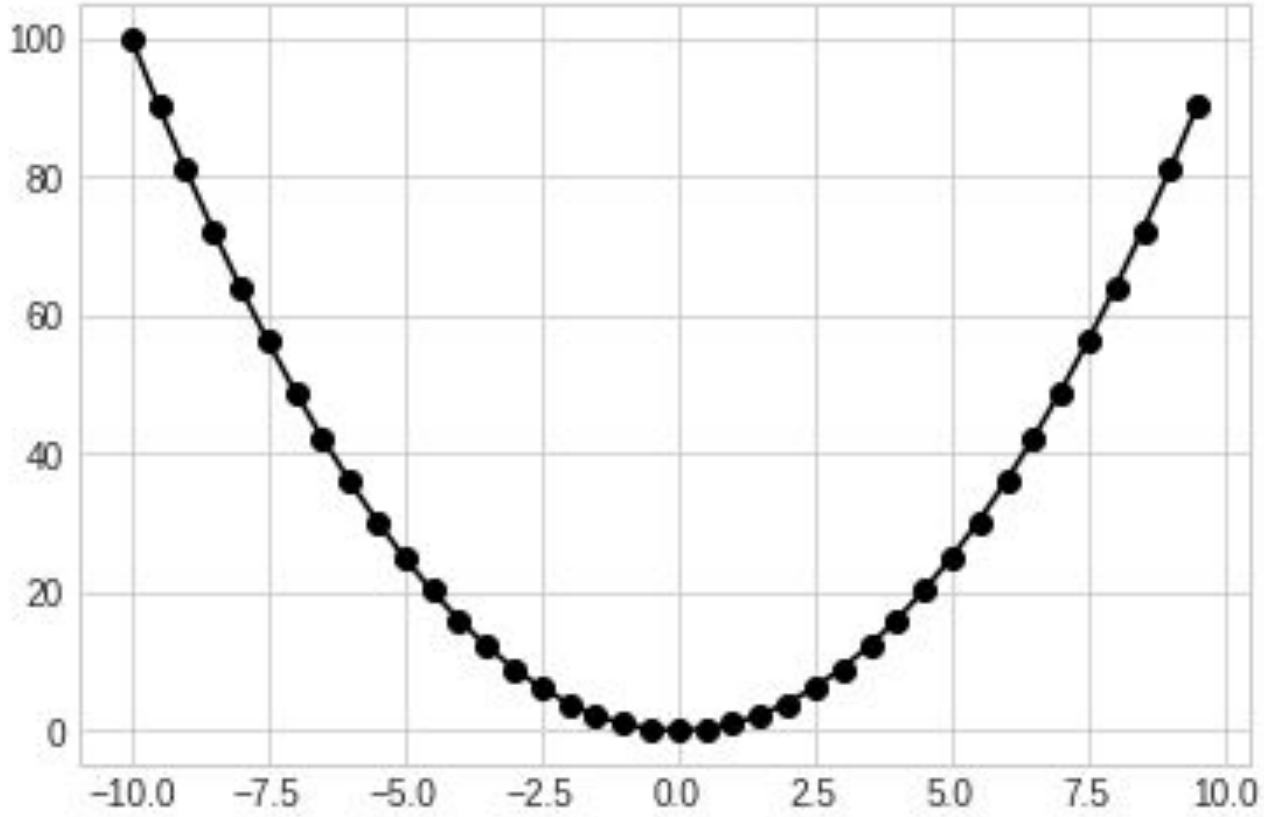
NumPy

All needed Numerical Methods

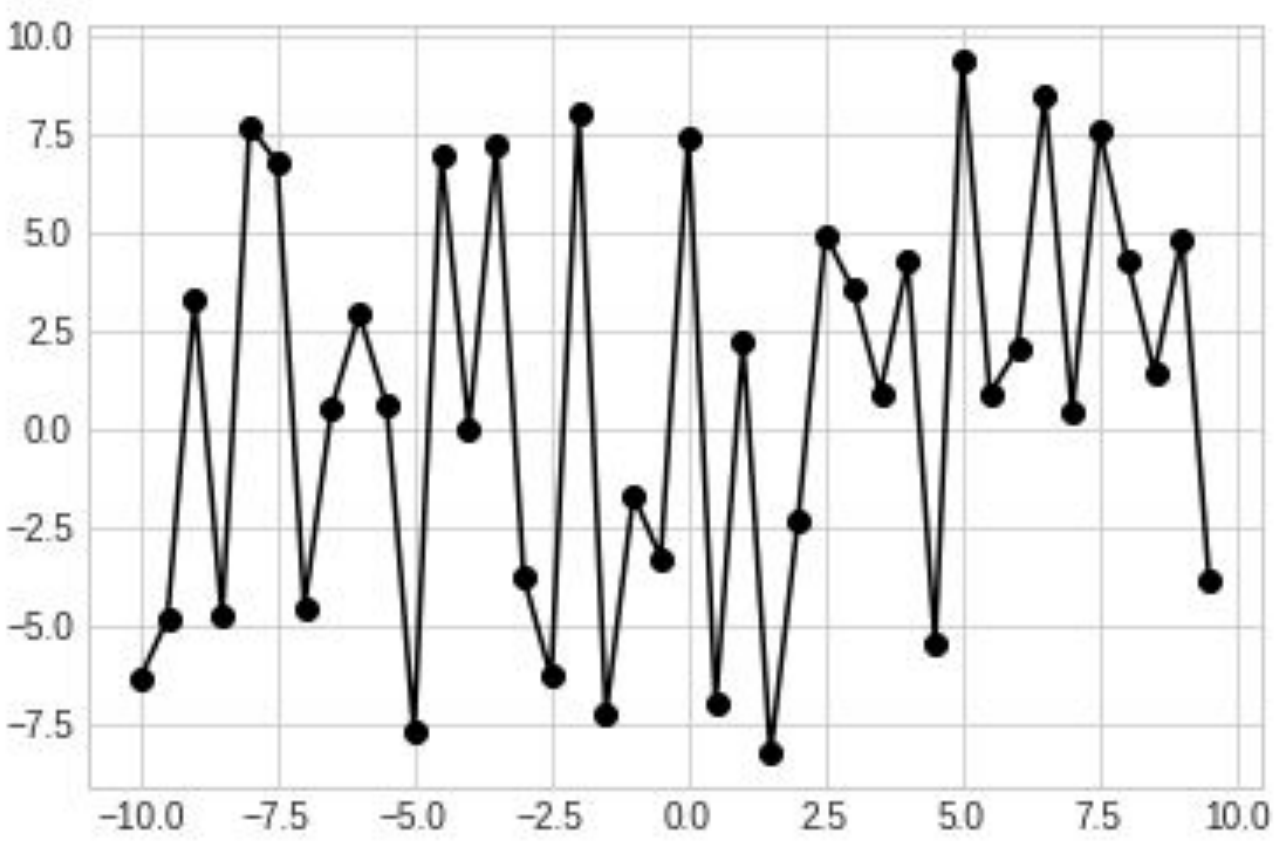
Mathematics consts	exp, pi, Euler–Mascheroni constant, ...
Mathematics functions	sin, cos, log, power, ...
Linear algebra	Matrix multiplication, transpose, …
Numerical Methods	MLS, FFT, ...
Complex Algebra	Is a default approach

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(-10, 10, .5)
y = np.power(x, 2)
plt.plot(x, y, '-ok')
plt.show()

noise = np.random.uniform( low=-10, high=10,
size=len(x))
plt.plot(x, noise, '-ok')
plt.show()

y_real = y + noise
plt.plot(x, y_real, '-ok')
plt.show()

A = np.vstack([x**2, x, np.ones_like(x)]).T
results = np.linalg.lstsq(A, y_real,
rcond=None)
print(results[0])
```

```
x = np.arange(-10, 10, .5)
y = np.power(x, 2)
plt.plot(x, y, '-ok')
plt.show()
```



```
noise =
np.random.uniform( low=-10,
high=10, size=len(x))
plt.plot(x, noise, '-ok')
plt.show()
```

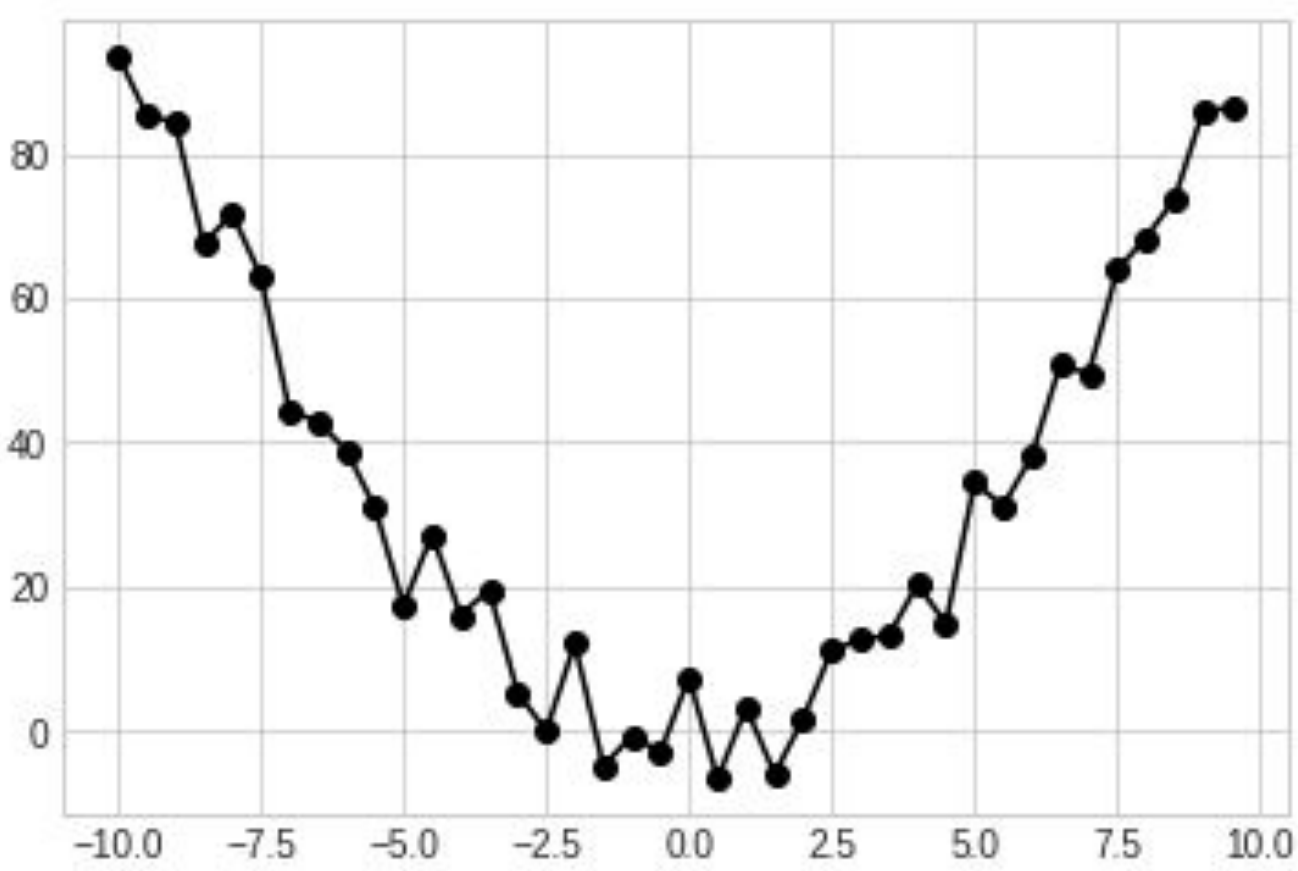


```
y_real = y + noise
plt.plot(x, y_real, '-ok')
plt.show()
```

```
A = np.vstack([x** 2, x,
np.ones_like(x)]).T
results = np.linalg.lstsq(A,
y_real, rcond=None)
print(results[0])
```

Result:

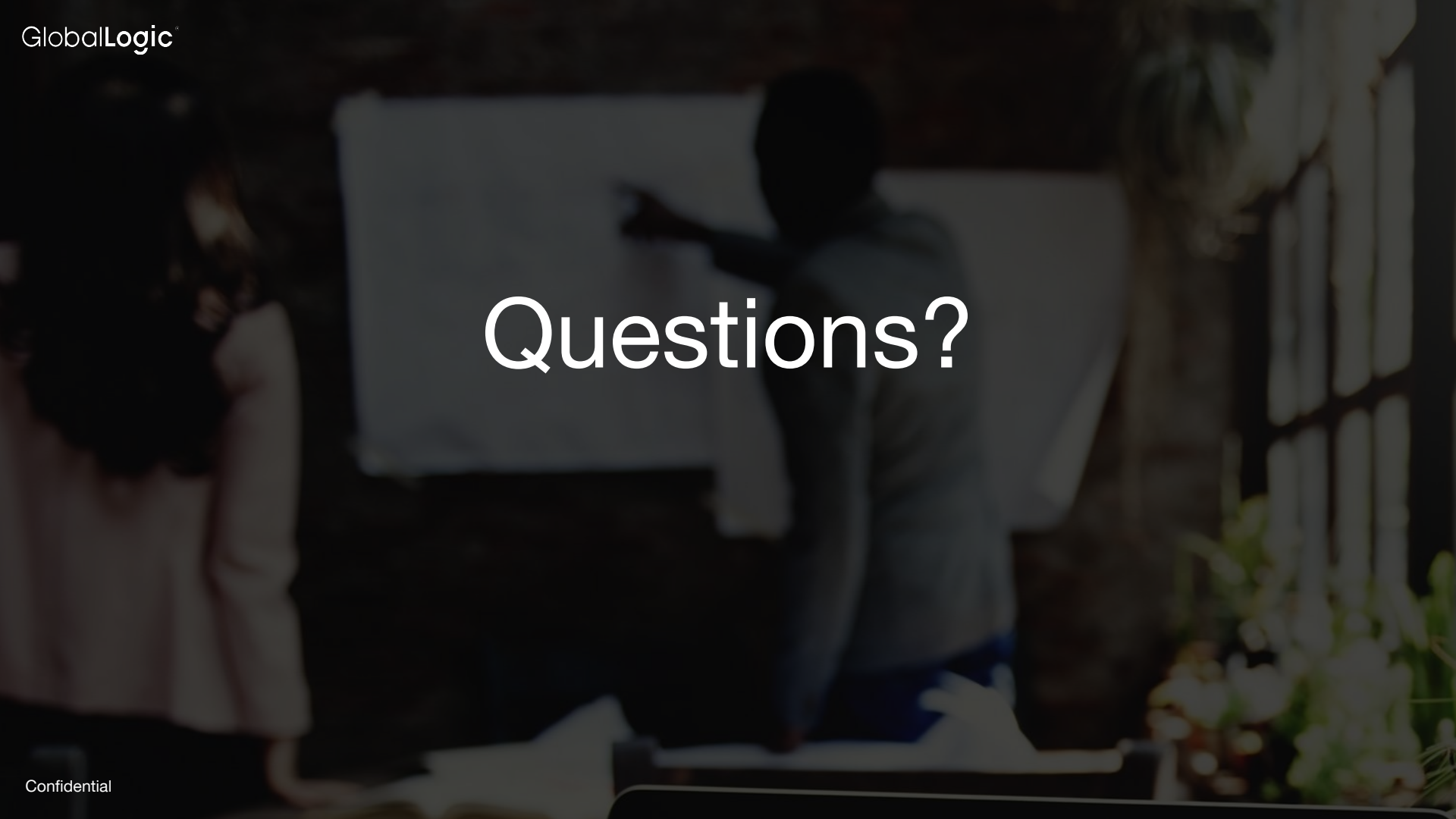
(array([
1.0097034, # a * x^2
0.20207776, # b * x
0.47732674 # c
]))



SymPy

```python
from sympy import *
init_printing()


x, y = symbols('x y')
f = (x**2 + y**2)
pprint(f)

F = Integral(f, x, y)
pprint(F)
pprint(F.doit())

pprint(diff(f,x))

lambda_ = Symbol('lambda')
res = sqrt(6 * Sum(1/lambda_**2,
(lambda_, 1, oo)))
pprint(res)
pprint(res.doit())
```

```
 2    2                                |
x  + y                                 | function "f"
---------------------------------------|
⌠ ⌠                                    |
⎮ ⎮ ⎛ 2    2⎞                          |
⎮ ⎮ ⎝x  + y ⎠ dx dy                    | integral of "f"
⌡ ⌡                                    |
---------------------------------------|
 3         3                           |
x ·y    x·y                            |
──── + ────                            | integration result of "f"
 3      3                              |
---------------------------------------|
2·x                                    | differenciation of "f" by "x"

=======================================

              ________                 |
             ╱   ∞                     |
            ╱   ___                    |
           ╱    ╲                      |
          ╱      ╲    1                |
√6·      ╱       ╱   ──                | infinit summ
        ╱       ╱     2                |
       ╱        ‾‾‾  λ                 |
     ╲╱        λ = 1                   |
---------------------------------------|
π                                      | result of summ
```



Questions?