TALLER DE DEEP LEARNING

Lectura 1:
Herramientas: Tensorflow, Keras, Python, Google Colab

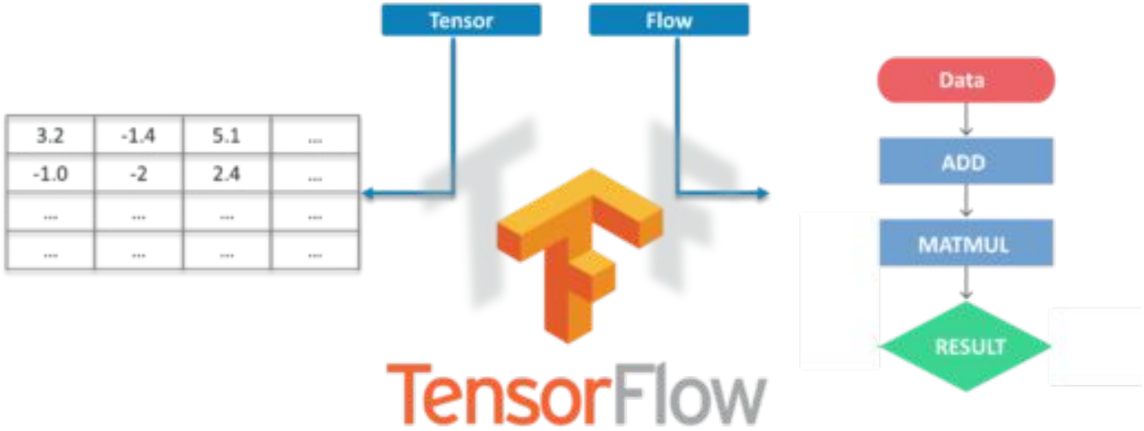
Dennis Núñez Fernández

Frameworks

- Para principiantes: **Keras**
- Para producción/investigación: **Tensorflow**
- Para investigación: **PyTorch**
- Para producción en AWS: MXNet
- Para producción en Azufre: CNTK
- Para desarrolladores de Java: DL4J

Frameworks

Frameworks

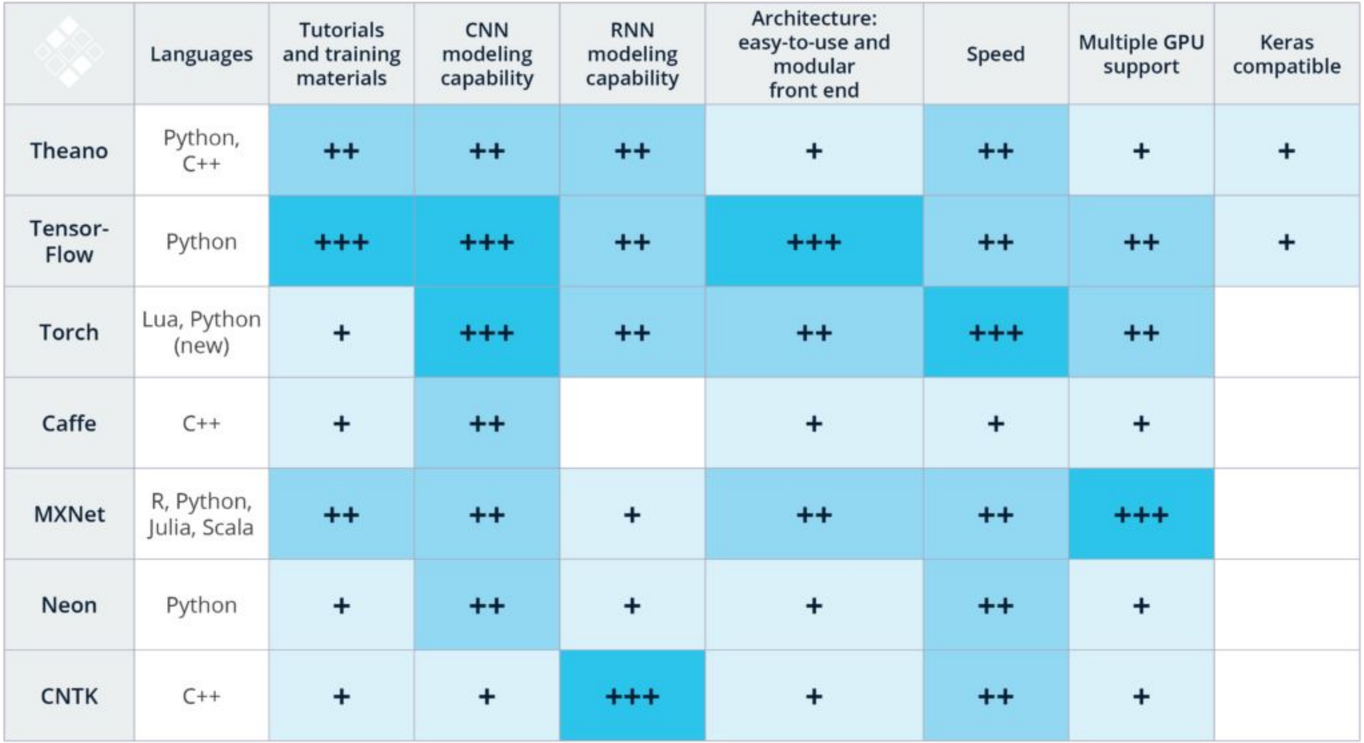


3.2	-1.4	5.1	...
-1.0	-2	2.4	...
...	...	...	...
...	...	...	...

Frameworks

Name	Platform	Written In	Cuda	Parallel Execution	Trained Model	RNN	CNN
Tensorflow	Linux, Window,MacOs, Rasbian,Mobile, Webapp	Python, C++, Cuda	Yes	Yes	Yes	Yes	Yes
Pytorch	Linux, Window, MacOs	Python, C++, Cuda	Yes	Yes	Yes	Yes	Yes
Keras	Linux, MacOs, window	Python	Yes	Yes	Yes	Yes	Yes
Mxnet	Linux, Window, Mac,Mobile, Webapp	C++, Python, R, Julia, Scala, Go, Perl	Yes	Yes	Yes	Yes	Yes
Deeplearning4j	Window, Linux,Mac, Mobile	Java, Scala, Cuda, C++, Perl, Python, Closure	Yes	Yes	Yes	Yes	Yes
Microsoft CNTK	Window, Linux	C++	Yes	Yes	Yes	Yes	Yes

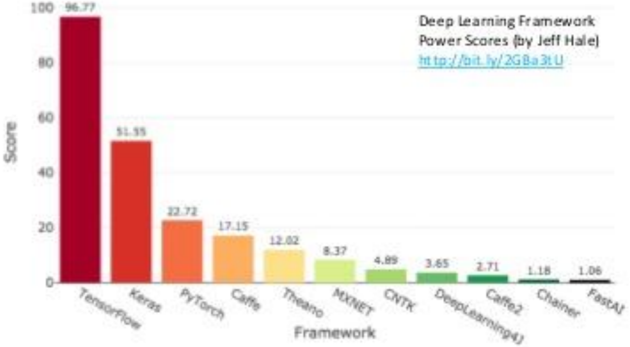
Frameworks



	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

Frameworks

Deep Learning Frameworks



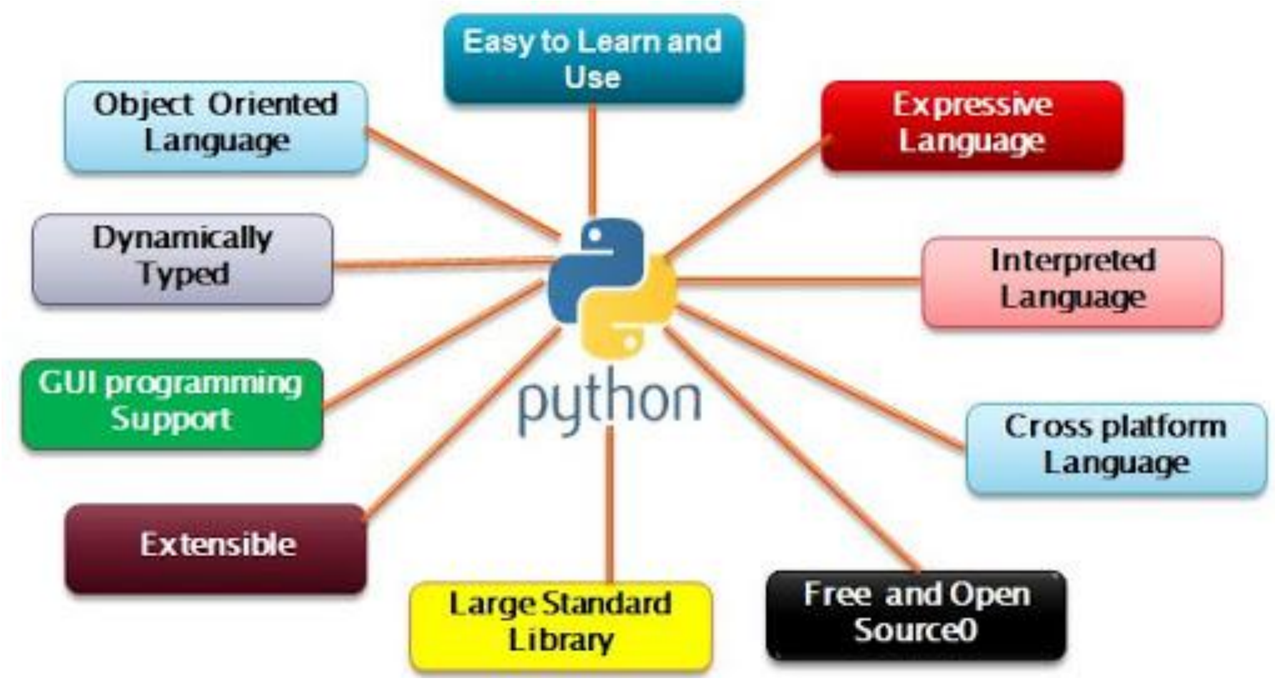
Deep Learning Framework Power Scores (by Jeff Hale) http://bit.ly/2GBa3tU

Factors to consider:

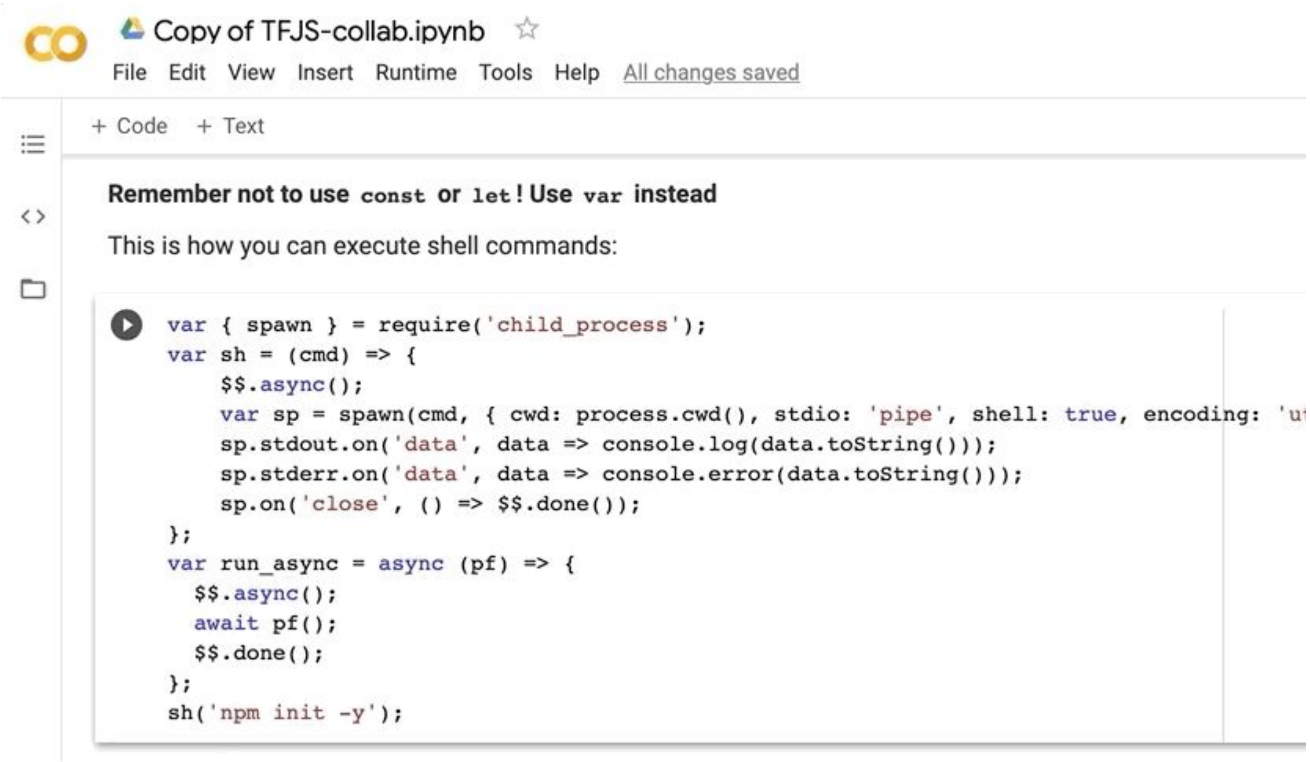
- Learning curve
- Speed of development
- Size and passion of community
- Number of papers implemented in framework
- Likelihood of long-term growth and stability
- Ecosystem of tooling

1. TensorFlow
2. K Keras
3. PyTorch
4. Caffe
5. theano
6. APACHE mxnet
7. Microsoft CNTK
8. DL4J
9. Caffe2
10. Chainer
11. fast.ai

Python


Easy to Learn and Use
Object Oriented Language
Expressive Language
Dynamically Typed
Interpreted Language
GUI programming Support
python
Cross platform Language
Extensible
Large Standard Library
Free and Open Source0

Google Colab

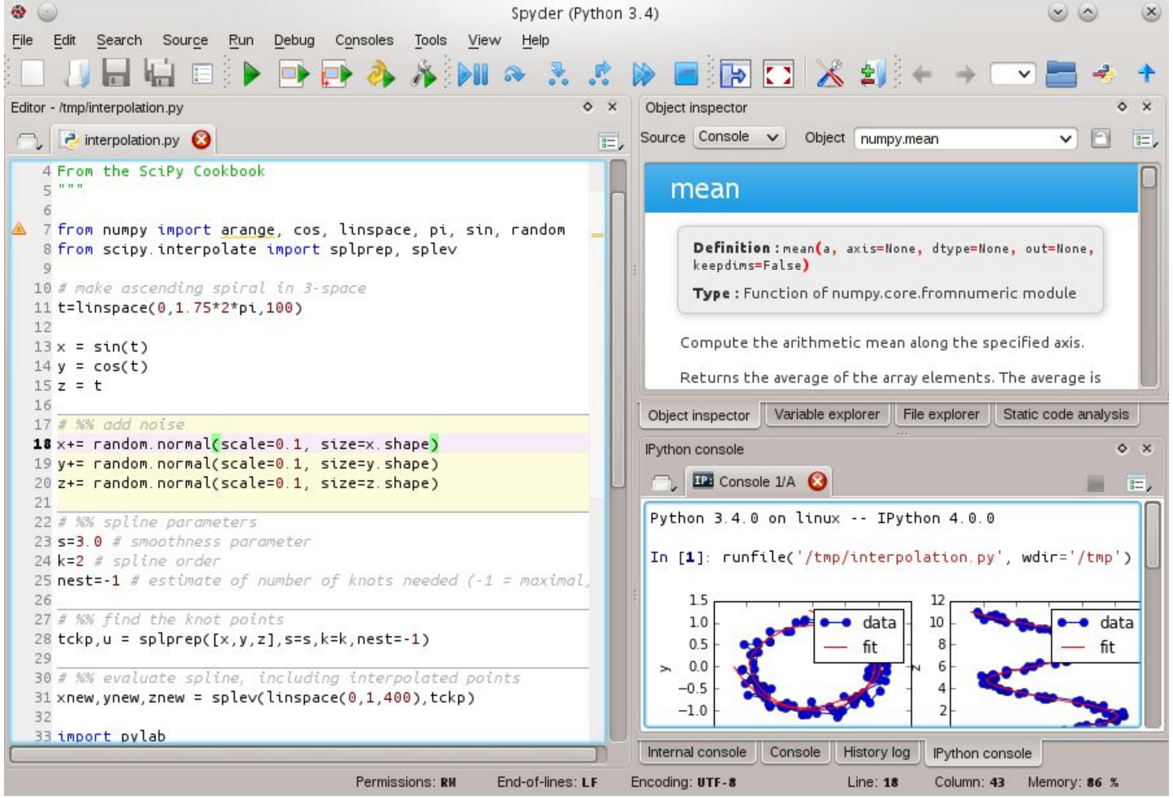


Remember not to use `const` or `let`! Use `var` instead

This is how you can execute shell commands:

```
var { spawn } = require('child_process');
var sh = (cmd) => {
    $$.async();
    var sp = spawn(cmd, { cwd: process.cwd(), stdio: 'pipe', shell: true, encoding: 'u
    sp.stdout.on('data', data => console.log(data.toString()));
    sp.stderr.on('data', data => console.error(data.toString()));
    sp.on('close', () => $$.done());
};
var run_async = async (pf) => {
  $$.async();
  await pf();
  $$.done();
};
sh('npm init -y');
```

Spyder



```
From the SciPy Cookbook
"""

from numpy import arange, cos, linspace, pi, sin, random
from scipy.interpolate import splprep, splev

# make ascending spiral in 3-space
t=linspace(0,1.75*2*pi,100)

x = sin(t)
y = cos(t)
z = t

# %% add noise
x+= random.normal(scale=0.1, size=x.shape)
y+= random.normal(scale=0.1, size=y.shape)
z+= random.normal(scale=0.1, size=z.shape)

# %% spline parameters
s=3.0 # smoothness parameter
k=2 # spline order
nest=-1 # estimate of number of knots needed (-1 = maximal,

# %% find the knot points
tckp,u = splprep([x,y,z],s=s,k=k,nest=-1)

# %% evaluate spline, including interpolated points
xnew,ynew,znew = splev(linspace(0,1,400),tckp)

import pylab
```

mean

Definition : mean(a, axis=None, dtype=None, out=None, keepdims=False)

Type : Function of numpy.core.fromnumeric module

Compute the arithmetic mean along the specified axis.

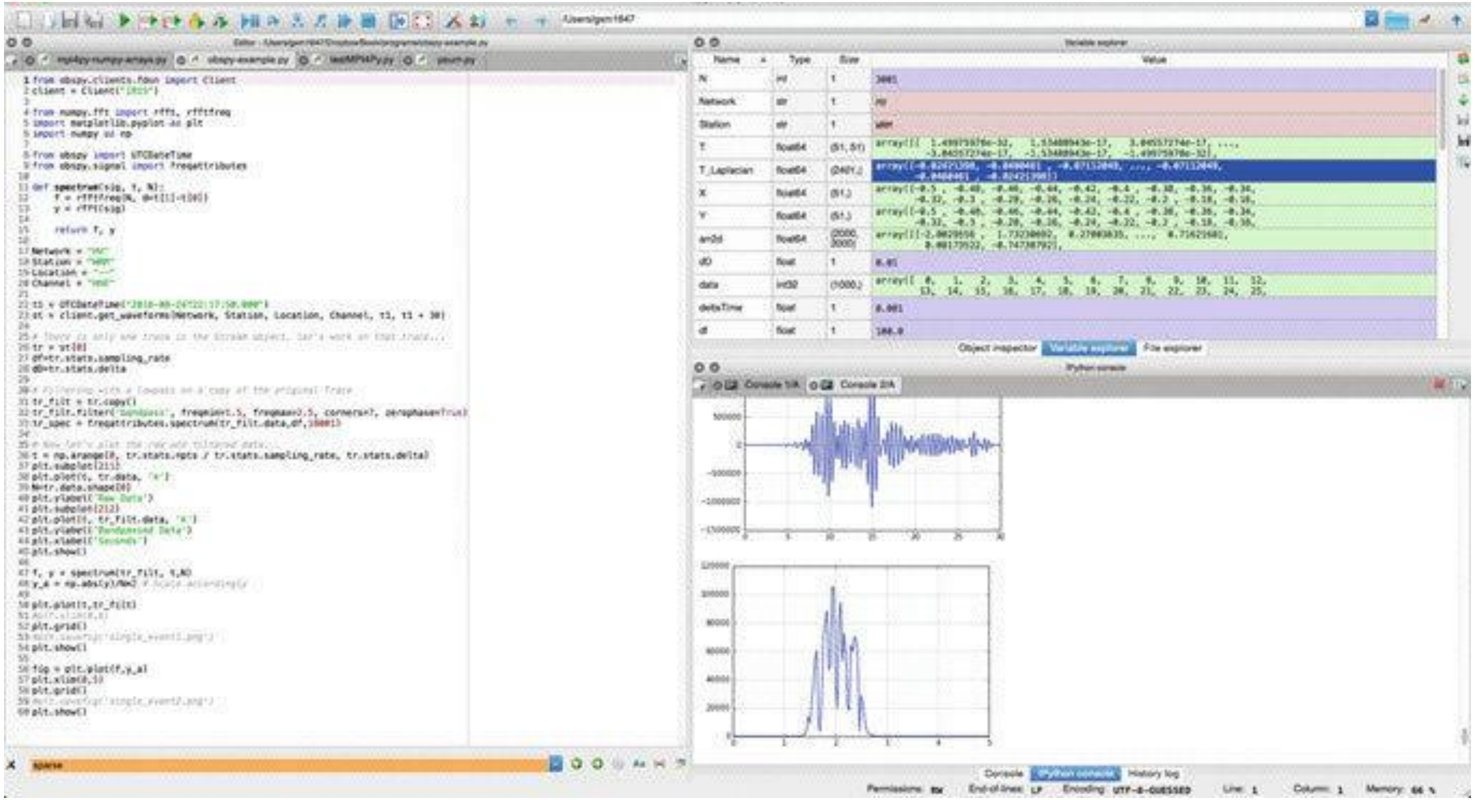
Returns the average of the array elements. The average is

```
Python 3.4.0 on linux -- IPython 4.0.0

In [1]: runfile('/tmp/interpolation.py', wdir='/tmp')
```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 18 Column: 43 Memory: 86 %

Spyder



```
from obspy.clients.fdsn import Client
client = Client("IRIS")

from numpy.fft import rfft, rfftfreq
import matplotlib.pyplot as plt
import numpy as np

from obspy import UTCDateTime
from obspy.signal import freqattributes

def spectrum(sig, t, N):
    f = rfftfreq(N, d=t[1]-t[0])
    y = rfft(sig)

    return f, y

Network = "HV"
Station = "WRM"
Location = "--"
Channel = "HHZ"

t1 = UTCDateTime("2010-08-24T22:17:50.000")
st = client.get_waveforms(Network, Station, Location, Channel, t1, t1 + 30)

# There is only one trace in the Stream object, let's work on that trace...
tr = st[0]
df=tr.stats.sampling_rate
dD=tr.stats.delta

# Filtering with a lowpass on a copy of the original Trace
tr_filt = tr.copy()
tr_filt.filter('bandpass', freqmin=1.5, freqmax=3.5, corners=7, zerophase=True)
tr_spec = freqattributes.spectrum(tr_filt.data,df,10001)

# Now let's plot the raw and filtered data...
t = np.arange(0, tr.stats.npts / tr.stats.sampling_rate, tr.stats.delta)
plt.subplot(211)
plt.plot(t, tr.data, 'k')
N=tr.data.shape[0]
plt.ylabel('Raw Data')
plt.subplot(212)
plt.plot(t, tr_filt.data, 'k')
plt.ylabel('Bandpassed Data')
plt.xlabel('Seconds')
plt.show()

f, y = spectrum(tr_filt, t,N)
y_a = np.abs(y)/N*2 # Scale accordingly

plt.plot(t,tr_filt)
#plt.xlim(0,8)
plt.grid()
#plt.savefig('simple_event1.png')
plt.show()

fig = plt.plot(f,y_a)
plt.xlim(0,5)
plt.grid()
#plt.savefig('simple_event2.png')
plt.show()
```

sparse

Name	Type	Size	Value
N	int	1	3001
Network	str	1	HV
Station	str	1	WRM
T	float64	(51, 51)	array([[1.49975978e-32, 1.53480943e-17, 3.04557274e-17, ..., -3.04557274e-17, -1.53480943e-17, -1.49975978e-32],
T_Laplacian	float64	(2401,)	array([-0.02421398, -0.0400461 , -0.07112040, ..., -0.07112040, -0.0400461 , -0.02421398])
X	float64	(51,)	array([-0.5 , -0.48, -0.46, -0.44, -0.42, -0.4 , -0.38, -0.36, -0.34, -0.32, -0.3 , -0.28, -0.26, -0.24, -0.22, -0.2 , -0.18, -0.16,
Y	float64	(51,)	array([-0.5 , -0.48, -0.46, -0.44, -0.42, -0.4 , -0.38, -0.36, -0.34, -0.32, -0.3 , -0.28, -0.26, -0.24, -0.22, -0.2 , -0.18, -0.16,
arr2d	float64	(2000, 2000)	array([[-2.0029558 , 1.73230692, 0.27003035, ..., 0.71621681, 0.00173522, -0.74730792],
dD	float	1	0.01
data	int32	(1000,)	array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
deltaTime	float	1	0.001
df	float	1	100.0

Permissions: RW End-of-lines: LF Encoding: UTF-8-GUESSED Line: 1 Column: 1 Memory: 66 %