# Intro to NumPy, SciPy, and Matplotlib

## Numpy

```
import numpy as np
```

## Comparision of Numpy Array and List

```
x = [2,3,4,6] #usual Python list
y = np.array(x) #numpy array

print(type(x),x)
print(type(y),y)
```

```
<class 'list'> [2, 3, 4, 6]
<class 'numpy.ndarray'> [2 3 4 6]
```

```
print(x[1:3])
```

```
[3, 4]
```

```
print(y[1:3])
```

```
[3 4]
```

```
print(x[[0,2]])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 print(x[[0,2]])

TypeError: list indices must be integers or slices, not list
```

```
print(y[[0,2]])
```

```
[2 4]
```

```
print(y[y>3])
```

```
[4 6]
```

```
print(x * 5)
```

```
[2, 3, 4, 6, 2, 3, 4, 6, 2, 3, 4, 6, 2, 3, 4, 6, 2, 3, 4, 6]
```

```
print(y * 5)
```

```
[10 15 20 30]
```

```
print(x ** 2)
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[11], line 1
----> 1 print(x ** 2)

TypeError: unsupported operand type(s) for ** or pow(): 'list' and
'int'
```

```
print(y ** 2)
```

```
[ 4  9 16 36]
```

```
matrix = [[1,2,4],[3,1,0]]
nd_array = np.array(matrix)
```

```
print(matrix[1][2])
```

```
0
```

```
print(nd_array[1, 2])
```

```
0
```

```
print(np.random.rand())
```

```
0.37493731407553665
```

```
print(np.random.randn())
```

```
0.2955722831707411
```

```
print(np.random.randn(4))
```

```
[ 0.059816   -0.46082397  1.30260184  1.96200824]
```

```
print(np.random.randn(4, 5))
```

```
[[-2.46003788  0.89135341 -1.14713491 -1.34096672  0.54791307]
 [-0.17436051  1.01946694 -0.12595682  1.41222676 -1.74029575]
 [ 0.81132472 -1.89380222  1.33932638  1.18751632 -1.32864742]
 [ 0.14559509 -1.26415164 -0.63193315 -0.43852112  0.06850322]]
```

## range() Method in Python and arange() in Numpy

```
print(np.arange(0, 8, 0.1))
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6
1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4
3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2
5.3
```

```
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.
7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9]
```

```
print(np.range(0, 8, 0.1)) #range only works with int
```

```
---------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
Cell In[21], line 1
----> 1 print(np.range(0, 8, 0.1))

File C:\ProgramData\anaconda3\lib\site-packages\numpy\__init__.py:311,
in __getattr__(attr)
    308     from .testing import Tester
    309     return Tester
--> 311 raise AttributeError("module {!r} has no attribute "
    312                      "{!r}".format(__name__, attr))

AttributeError: module 'numpy' has no attribute 'range'
```

```
list(range(0,8, 1))
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

```
%timeit np.arange(0, 10000)
%timeit range(0, 10000)
```

```
11 µs ± 735 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops
each)
516 ns ± 46.9 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops
each)
```

## SciPy

```
from scipy import optimize

def f(x):
    return (x[0] - 3.2) ** 2 + (x[1] - 0.1) ** 2 + 3

print(f([3.2, 0.1]))
```

```
3.0
```

```
x_min = optimize.minimize(f, [5, 5])
print(x_min)
```

```
  message: Optimization terminated successfully.
  success: True
   status: 0
      fun: 3.0000000000012976
        x: [ 3.200e+00  1.000e-01]
```

```
      nit: 3
      jac: [-2.146e-06  7.749e-07]
 hess_inv: [[ 9.406e-01 -1.618e-01]
            [-1.618e-01  5.594e-01]]
     nfev: 12
     njev: 4
```

```
print(x_min.x)
```

```
[3.19999893 0.10000038]
```

```
from scipy import linalg

a = np.array([[3, 2, 0], [1, -1, 0], [0, 5, 1]])
b = np.array([2, 4, -1])

x = linalg.solve(a, b)
print(x)
```

```
[ 2. -2.  9.]
```

```
print(np.dot(a, x))
```

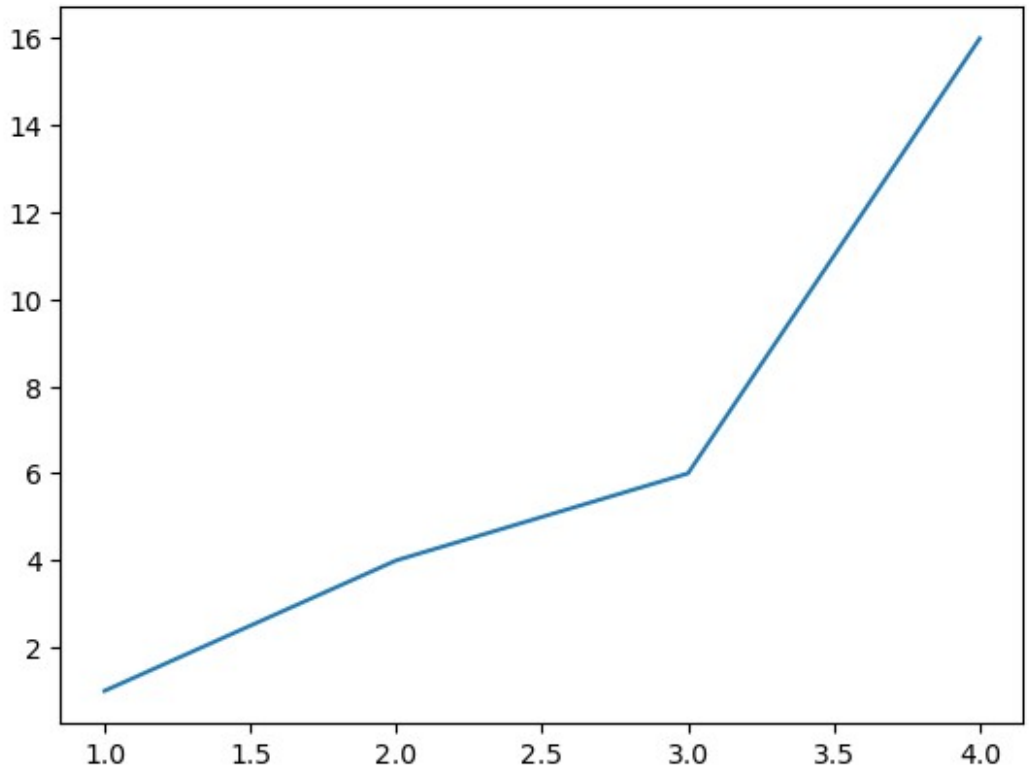
```
[ 2.  4. -1.]
```

```
X = np.random.randn(4, 3)
U, D, V = linalg.svd(X)
print(U.shape, D.shape, V.shape)
print(type(U), type(D), type(V))
```

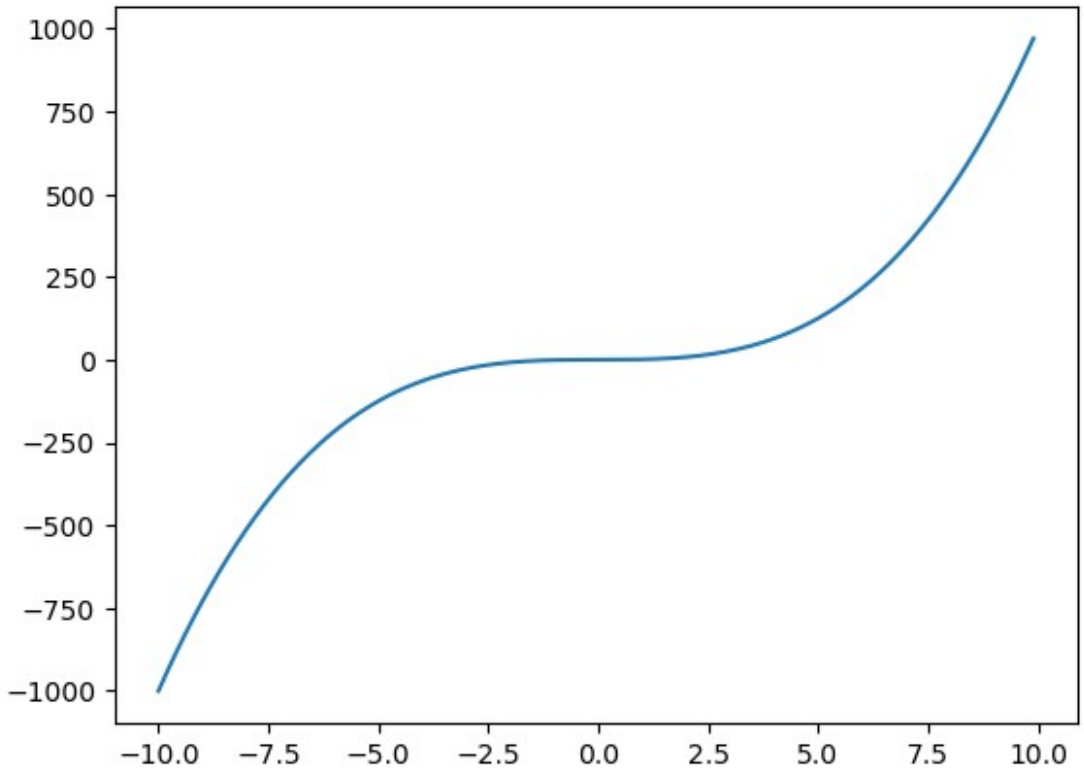
```
(4, 4) (3,) (3, 3)
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class
'numpy.ndarray'>
```

## Matplotlib

```
%matplotlib inline

from matplotlib import pylab as plt

plt.plot([1, 2, 3, 4],[1, 4, 6, 16])
plt.show()
```




```
x = np.arange(-10, 10, 0.1)
y = x ** 3
plt.plot(x, y)
plt.show()
```

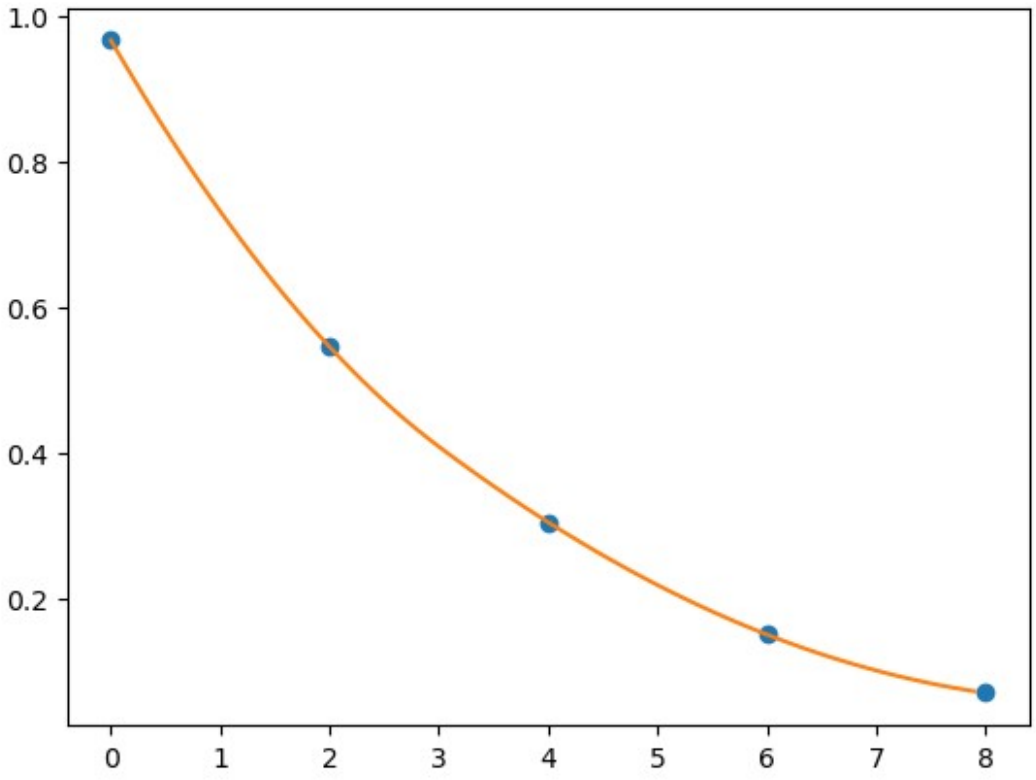



## Example of Using Matplotlib, Numpy and SciPy Together

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate

x = np.arange(0, 10.0, 2.0)
y = np.exp(-x/3.0) + np.random.randn(len(x)) * 0.05

print(x[:5])
print(y[:5])
```

```
[0. 2. 4. 6. 8.]
[0.96578507 0.54544041 0.30480403 0.15081353 0.07079136]
```

```
f = interpolate.interp1d(x, y, kind = 'quadratic')
xnew = np.arange(0.0, 8.0, 0.1)
ynew = f(xnew)

plt.plot(x, y, 'o', xnew, ynew, '-')
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8