Sixth_class

February 18, 2024

1 Introduction to Python - 6

Today we will learn one of the most powerful library **numpy**.

1.1 Introduction to Numpy

[ ]:
```
g = [1,2,3,44,5]
print(g)
```

```
[1, 2, 3, 44, 5]
```

[ ]:
```
g_square = g**2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/home/aburousan/Classes/Cosmic_Charade/Classes/Sixth_class.ipynb Cell 5 line 1

----> <a href='vscode-notebook-cell:/home/aburousan/Classes/Cosmic_Charade/
 ↪Classes/Sixth_class.ipynb#X20sZmlsZQ%3D%3D?line=0'>1</a> g_square = g**2

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

[ ]:
```
import math as m
```

[ ]:
```
m.tan(10)
```

[ ]:
```
0.6483608274590866
```

[ ]:
```
a_tan = m.tan(g)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/home/aburousan/Classes/Cosmic_Charade/Classes/Sixth_class.ipynb Cell 7 line 1

----> <a href='vscode-notebook-cell:/home/aburousan/Classes/Cosmic_Charade/
 ↪Classes/Sixth_class.ipynb#X15sZmlsZQ%3D%3D?line=0'>1</a> a_tan = m.tan(g)

TypeError: must be real number, not list
```

[ ]:
```
a_tan = []
for i in g:
    a_tan.append(m.tan(i))
```

[ ]:
```
a_tan
```

[ ]:
```
[1.5574077246549023,
 -2.185039863261519,
 -0.1425465430742778,
 0.017704699278685777,
 -3.380515006246586]
```

[ ]:
```
import numpy as np
```

[ ]:
```
np.tan(np.pi/3)
```

[ ]:
```
1.7320508075688767
```

[ ]:
```
a_tan_np = np.tan(g)
print(a_tan_np)
```

```
[ 1.55740772 -2.18503986 -0.14254654  0.0177047  -3.38051501]
```

[ ]:
```
lis1 = [1,2,3,4,100,101]
```

[ ]:
```
list_np = np.array([1,2,3,4,100,101])
print(list_np)
```

```
[  1   2   3   4 100 101]
```

[ ]:
```
list_np_2 = np.array(g)
print(list_np_2)
```

```
[ 1  2  3 44  5]
```

[ ]:
```
list_np_2**np.pi
```

[ ]:
```
array([1.00000000e+00, 8.82497783e+00, 3.15442807e+01, 1.45565634e+05,
       1.56992545e+02])
```

[ ]:
```
list_np_2[1:-1]
```

[ ]:
```
array([ 2,  3, 44])
```

[ ]:
```
list_np_2[2]
```

[ ]:
```
3
```

Numpy array Functions

np.linspace(start,end,number of elements)

[ ]:
```
len(np.linspace(5,10,100))
```

[ ]:
```
100
```

[ ]:
```
np.linspace(1,6,6)
```

[ ]:
```
array([1., 2., 3., 4., 5., 6.])
```

Make a plot of sin function from 0 to 4 π

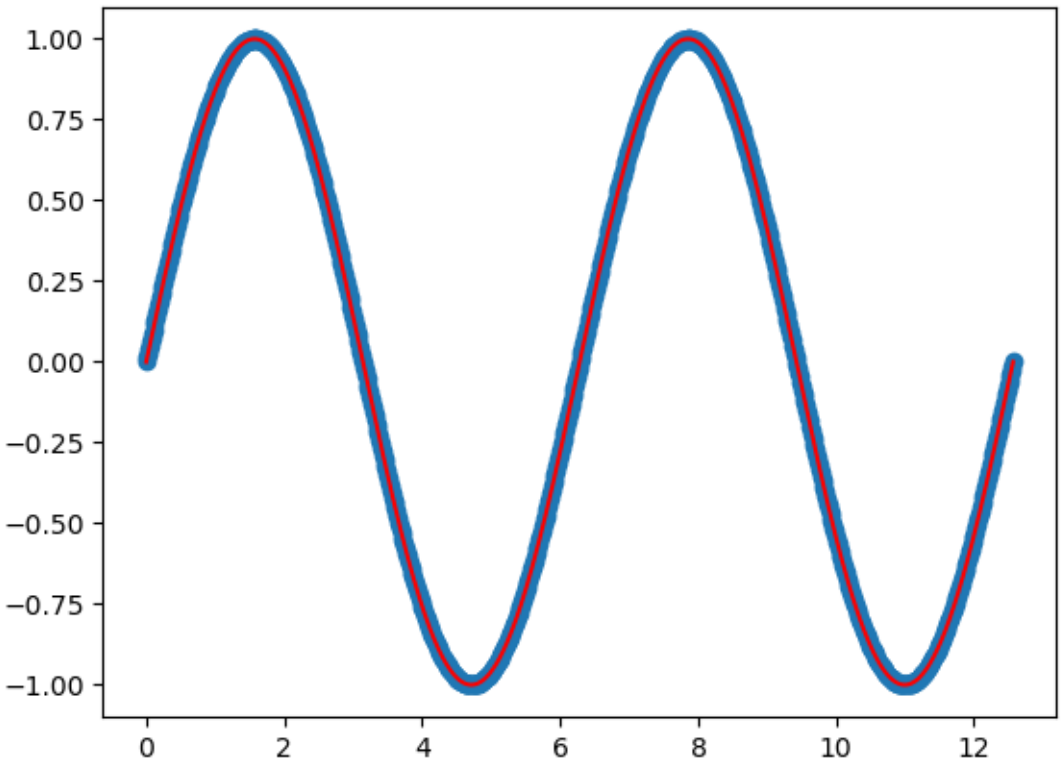
[ ]:
```
import matplotlib.pyplot as plt
```

[ ]:
```
x = np.linspace(0,4*np.pi,1000)
# print(x)
```

[ ]:
```
y = np.sin(x)
# print(y)
```

[ ]:
```
plt.plot(x,y,color="red")
plt.scatter(x,y)
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x7feaeadaad80>
```



np.arange(first_number, last_number, difference)

```
[ ]: np.arange(2,10+1,1,dtype=float)
```

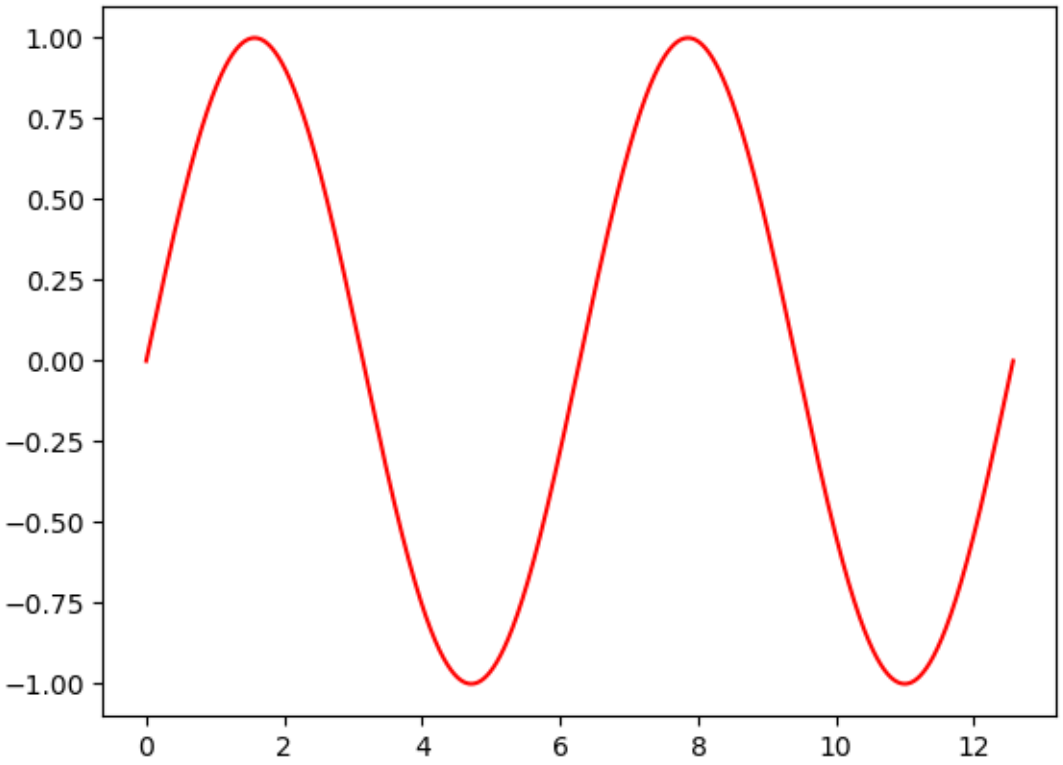
```
[ ]: array([ 2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

```
[ ]: x1 = np.arange(0,4*np.pi,0.0001)
```

```
[ ]: y1 = np.sin(x1)
```

```
[ ]: plt.plot(x1,y1,color="red")
     # plt.scatter(x,y)
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7feaeac99400>]
```




```
[ ]: np.logspace(0,4,num=4,base=2)
```

```
[ ]: array([ 1.        ,  2.5198421 ,  6.34960421, 16.        ])
```

[ ]:
```
a1 = np.zeros([2,2])
a1
```

[ ]:
```
array([[0., 0.],
       [0., 0.]])
```

[ ]:
```
a2 = np.ones(10)
a2
```

[ ]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

[ ]:
```
np.random.random(10)
```

[ ]:
```
array([0.01972457, 0.23959396, 0.71970662, 0.07246364, 0.57031936,
       0.50243736, 0.23788331, 0.56264584, 0.53063279, 0.01139774])
```

[ ]:
```
np.random.randn(10)/100
```

[ ]:
```
array([ 0.005487  , -0.00904539, -0.0050521 , -0.01785743, -0.00099428,
        0.003406  ,  0.00942915, -0.02880442,  0.01141769, -0.00820323])
```

[ ]:
```
a3 = np.array([[2,3, 10],[4,6, 11],[8,100,101]])# input rowswise
a3
```

[ ]:
```
array([[  2,   3,  10],
       [  4,   6,  11],
       [  8, 100, 101]])
```

[ ]:
```
np.log(a3)
```

[ ]:
```
array([[0.69314718, 1.09861229, 2.30258509],
       [1.38629436, 1.79175947, 2.39789527],
       [2.07944154, 4.60517019, 4.61512052]])
```

[ ]:
```
100/np.random.randn(10)
```

[ ]:
```
array([ -34.47018591,   43.24678499,   79.94630035,  424.41258002,
         99.55551808,  111.36245449,  103.61206934, -566.03847268,
        287.60492381, -190.76121439])
```

1.1.1 Masking

[ ]:
```
a = np.array([2,4,5,6,7,8,10])
```

[ ]:
```
a>5
```

[ ]:
```
array([False, False, False,  True,  True,  True,  True])
```

[ ]:
```
sum(a>5)
```

[ ]: 4

[ ]:
```
b = a[a>5]
b
```

[ ]: array([6, 7, 8, 10])

[ ]:
```
a = np.random.rand(10_000)
a
```

[ ]: array([0.95494444, 0.18959762, 0.27037102, …, 0.63294289, 0.5103118 ,
0.83263278])

[ ]:
```
np.mean(a)
```

[ ]: 0.49791621645321377

[ ]:
```
np.std(a)
```

[ ]: 0.28702749495074525

[ ]:
```
a = np.array([1,2,3,4])
np.cumsum(a)
```

[ ]: array([1, 3, 6, 10])

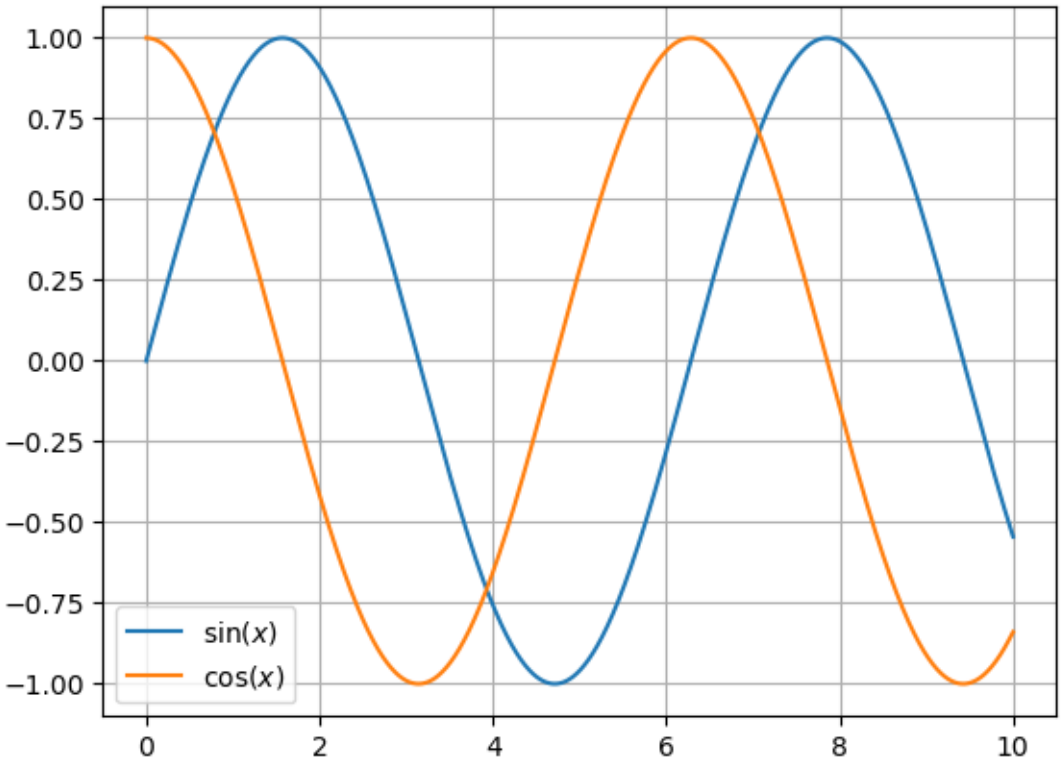
1.1.2 Calculus using Numpy

Take derivative of sin(x)

[ ]:
```
x = np.linspace(0,10,10_000)
y = np.sin(x)
```

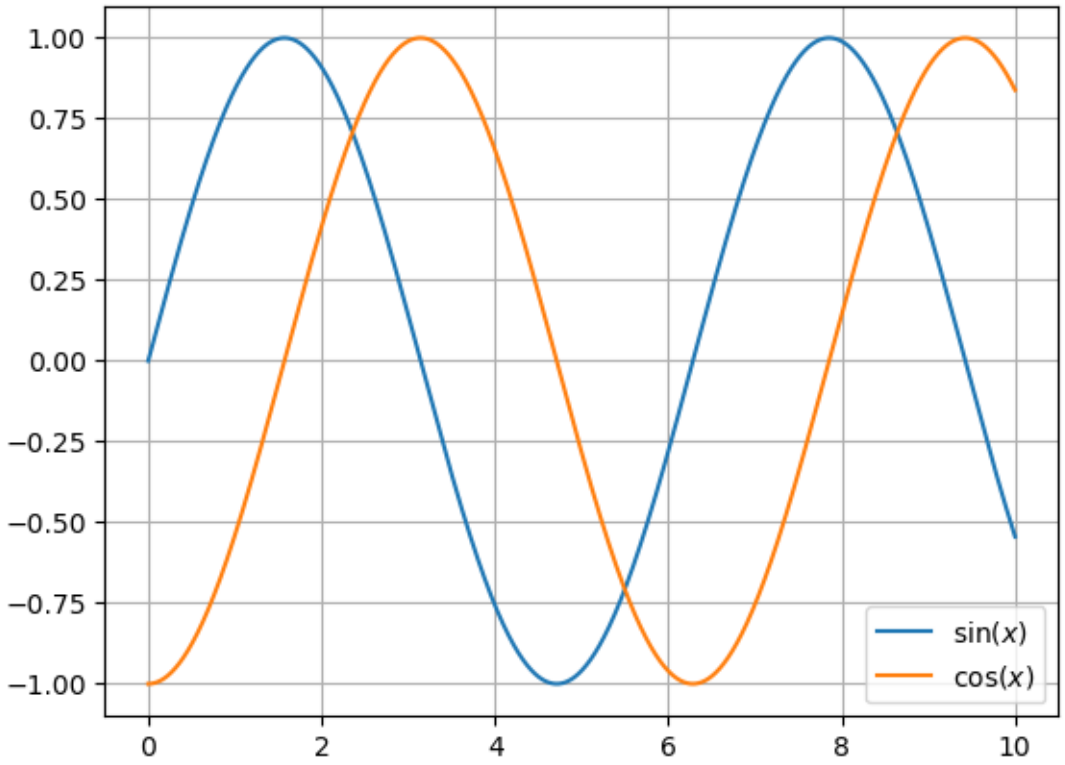
[ ]:
```
dy_dx = np.gradient(y,x)# find derivative
```

[ ]:
```
plt.plot(x,y,label=r"$\sin(x)$")
plt.plot(x,dy_dx,label=r"$\cos(x)$")
plt.legend()
plt.grid()
```



Take integartion sin(x) 0 - 10

[ ]:
```
y_integrate = np.cumsum(y)*(x[1]-x[0]) - 1
plt.plot(x,y,label=r"$\sin(x)$")
plt.plot(x,y_integrate,label=r"$\cos(x)$")
plt.legend()
plt.grid()
```



[ ]:
```
a1 = np.array([[1,23,8],[2,7,8]])
print(a1)
```

```
[[ 1 23  8]
 [ 2  7  8]]
```

[ ]:
```
a1.shape
```

[ ]:
```
(2, 3)
```

1.1.3 Multivariable functions

$$f(x, y) = z = x^2 + y^2$$

[ ]:
```
x = np.linspace(0,5,500)
y = np.linspace(0,5,500)
```

[ ]:
```
z = x**2 + y**2
```

[ ]:
```
print(z)
```

```
[0.00000000e+00 2.00802406e-04 8.03209626e-04 1.80722166e-03
 3.21283850e-03 5.02006016e-03 7.22888663e-03 9.83931791e-03
 1.28513540e-02 1.62649949e-02 2.00802406e-02 2.42970912e-02
 2.89155465e-02 3.39356067e-02 3.93572717e-02 4.51805414e-02
 5.14054160e-02 5.80318955e-02 6.50599797e-02 7.24896687e-02
 8.03209626e-02 8.85538612e-02 9.71883647e-02 1.06224473e-01
 1.15662186e-01 1.25501504e-01 1.35742427e-01 1.46384954e-01
 1.57429087e-01 1.68874824e-01 1.80722166e-01 1.92971113e-01
 2.05621664e-01 2.18673821e-01 2.32127582e-01 2.45982948e-01
 2.60239919e-01 2.74898494e-01 2.89958675e-01 3.05420460e-01
 3.21283850e-01 3.37548845e-01 3.54215445e-01 3.71283649e-01
 3.88753459e-01 4.06624873e-01 4.24897892e-01 4.43572516e-01
 4.62648744e-01 4.82126578e-01 5.02006016e-01 5.22287059e-01
 5.42969707e-01 5.64053960e-01 5.85539817e-01 6.07427279e-01
 6.29716347e-01 6.52407018e-01 6.75499295e-01 6.98993177e-01
 7.22888663e-01 7.47185754e-01 7.71884450e-01 7.96984751e-01
 8.22486657e-01 8.48390167e-01 8.74695282e-01 9.01402002e-01
 9.28510327e-01 9.56020257e-01 9.83931791e-01 1.01224493e+00
 1.04095967e+00 1.07007602e+00 1.09959398e+00 1.12951354e+00
 1.15983470e+00 1.19055747e+00 1.22168184e+00 1.25320782e+00
 1.28513540e+00 1.31746459e+00 1.35019538e+00 1.38332778e+00
 1.41686178e+00 1.45079739e+00 1.48513460e+00 1.51987341e+00
 1.55501384e+00 1.59055586e+00 1.62649949e+00 1.66284473e+00
 1.69959157e+00 1.73674001e+00 1.77429006e+00 1.81224172e+00
 1.85059498e+00 1.88934984e+00 1.92850631e+00 1.96806439e+00
 2.00802406e+00 2.04838535e+00 2.08914824e+00 2.13031273e+00
 2.17187883e+00 2.21384653e+00 2.25621584e+00 2.29898675e+00
 2.34215927e+00 2.38573339e+00 2.42970912e+00 2.47408645e+00
 2.51886539e+00 2.56404593e+00 2.60962807e+00 2.65561182e+00
 2.70199718e+00 2.74878414e+00 2.79597271e+00 2.84356288e+00
 2.89155465e+00 2.93994803e+00 2.98874302e+00 3.03793961e+00
 3.08753780e+00 3.13753760e+00 3.18793900e+00 3.23874201e+00
 3.28994663e+00 3.34155285e+00 3.39356067e+00 3.44597010e+00
 3.49878113e+00 3.55199377e+00 3.60560801e+00 3.65962386e+00
 3.71404131e+00 3.76886037e+00 3.82408103e+00 3.87970329e+00
 3.93572717e+00 3.99215264e+00 4.04897972e+00 4.10620841e+00
 4.16383870e+00 4.22187059e+00 4.28030410e+00 4.33913920e+00
 4.39837591e+00 4.45801422e+00 4.51805414e+00 4.57849567e+00
 4.63933880e+00 4.70058353e+00 4.76222987e+00 4.82427781e+00
 4.88672736e+00 4.94957852e+00 5.01283127e+00 5.07648564e+00
 5.14054160e+00 5.20499918e+00 5.26985835e+00 5.33511914e+00
 5.40078152e+00 5.46684551e+00 5.53331111e+00 5.60017831e+00
 5.66744712e+00 5.73511753e+00 5.80318955e+00 5.87166317e+00
 5.94053839e+00 6.00981522e+00 6.07949366e+00 6.14957370e+00
 6.22005534e+00 6.29093859e+00 6.36222344e+00 6.43390990e+00
 6.50599797e+00 6.57848764e+00 6.65137891e+00 6.72467179e+00
 6.79836627e+00 6.87246236e+00 6.94696005e+00 7.02185935e+00
 7.09716025e+00 7.17286276e+00 7.24896687e+00 7.32547259e+00
```

7.40237991e+00 7.47968884e+00 7.55739937e+00 7.63551150e+00
7.71402524e+00 7.79294059e+00 7.87225754e+00 7.95197610e+00
8.03209626e+00 8.11261802e+00 8.19354139e+00 8.27486637e+00
8.35659295e+00 8.43872113e+00 8.52125092e+00 8.60418231e+00
8.68751531e+00 8.77124991e+00 8.85538612e+00 8.93992394e+00
9.02486335e+00 9.11020438e+00 9.19594700e+00 9.28209124e+00
9.36863707e+00 9.45558452e+00 9.54293356e+00 9.63068421e+00
9.71883647e+00 9.80739033e+00 9.89634580e+00 9.98570287e+00
1.00754615e+01 1.01656218e+01 1.02561837e+01 1.03471472e+01
1.04385123e+01 1.05302790e+01 1.06224473e+01 1.07150172e+01
1.08079887e+01 1.09013618e+01 1.09951366e+01 1.10893129e+01
1.11838908e+01 1.12788704e+01 1.13742515e+01 1.14700343e+01
1.15662186e+01 1.16628046e+01 1.17597921e+01 1.18571813e+01
1.19549721e+01 1.20531644e+01 1.21517584e+01 1.22507540e+01
1.23501512e+01 1.24499500e+01 1.25501504e+01 1.26507524e+01
1.27517560e+01 1.28531612e+01 1.29549681e+01 1.30571765e+01
1.31597865e+01 1.32627981e+01 1.33662114e+01 1.34700262e+01
1.35742427e+01 1.36788607e+01 1.37838804e+01 1.38893016e+01
1.39951245e+01 1.41013490e+01 1.42079751e+01 1.43150028e+01
1.44224320e+01 1.45302629e+01 1.46384954e+01 1.47471295e+01
1.48561652e+01 1.49656025e+01 1.50754415e+01 1.51856820e+01
1.52963241e+01 1.54073678e+01 1.55188132e+01 1.56306601e+01
1.57429087e+01 1.58555588e+01 1.59686106e+01 1.60820639e+01
1.61959189e+01 1.63101755e+01 1.64248336e+01 1.65398934e+01
1.66553548e+01 1.67712178e+01 1.68874824e+01 1.70041486e+01
1.71212164e+01 1.72386858e+01 1.73565568e+01 1.74748294e+01
1.75935036e+01 1.77125795e+01 1.78320569e+01 1.79519359e+01
1.80722166e+01 1.81928988e+01 1.83139827e+01 1.84354681e+01
1.85573552e+01 1.86796439e+01 1.88023341e+01 1.89254260e+01
1.90489195e+01 1.91728146e+01 1.92971113e+01 1.94218096e+01
1.95469095e+01 1.96724110e+01 1.97983141e+01 1.99246188e+01
2.00513251e+01 2.01784330e+01 2.03059425e+01 2.04338537e+01
2.05621664e+01 2.06908808e+01 2.08199967e+01 2.09495143e+01
2.10794334e+01 2.12097542e+01 2.13404765e+01 2.14716005e+01
2.16031261e+01 2.17350533e+01 2.18673821e+01 2.20001124e+01
2.21332444e+01 2.22667780e+01 2.24007133e+01 2.25350501e+01
2.26697885e+01 2.28049285e+01 2.29404701e+01 2.30764133e+01
2.32127582e+01 2.33495046e+01 2.34866527e+01 2.36242023e+01
2.37621536e+01 2.39005064e+01 2.40392609e+01 2.41784170e+01
2.43179746e+01 2.44579339e+01 2.45982948e+01 2.47390573e+01
2.48802214e+01 2.50217871e+01 2.51637544e+01 2.53061233e+01
2.54488938e+01 2.55920659e+01 2.57356396e+01 2.58796149e+01
2.60239919e+01 2.61687704e+01 2.63139505e+01 2.64595323e+01
2.66055156e+01 2.67519006e+01 2.68986872e+01 2.70458753e+01
2.71934651e+01 2.73414565e+01 2.74898494e+01 2.76386440e+01
2.77878402e+01 2.79374380e+01 2.80874374e+01 2.82378384e+01
2.83886410e+01 2.85398452e+01 2.86914510e+01 2.88434585e+01
2.89958675e+01 2.91486781e+01 2.93018904e+01 2.94555042e+01

```
2.96095196e+01 2.97639367e+01 2.99187553e+01 3.00739756e+01
3.02295975e+01 3.03856209e+01 3.05420460e+01 3.06988727e+01
3.08561010e+01 3.10137309e+01 3.11717624e+01 3.13301955e+01
3.14890302e+01 3.16482665e+01 3.18079044e+01 3.19679439e+01
3.21283850e+01 3.22892278e+01 3.24504721e+01 3.26121180e+01
3.27741656e+01 3.29366147e+01 3.30994655e+01 3.32627178e+01
3.34263718e+01 3.35904273e+01 3.37548845e+01 3.39197433e+01
3.40850037e+01 3.42506657e+01 3.44167293e+01 3.45831944e+01
3.47500612e+01 3.49173296e+01 3.50849997e+01 3.52530713e+01
3.54215445e+01 3.55904193e+01 3.57596957e+01 3.59293738e+01
3.60994534e+01 3.62699347e+01 3.64408175e+01 3.66121020e+01
3.67837880e+01 3.69558757e+01 3.71283649e+01 3.73012558e+01
3.74745483e+01 3.76482424e+01 3.78223381e+01 3.79968354e+01
3.81717343e+01 3.83470348e+01 3.85227369e+01 3.86988406e+01
3.88753459e+01 3.90522528e+01 3.92295613e+01 3.94072715e+01
3.95853832e+01 3.97638965e+01 3.99428115e+01 4.01221280e+01
4.03018462e+01 4.04819659e+01 4.06624873e+01 4.08434103e+01
4.10247348e+01 4.12064610e+01 4.13885888e+01 4.15711182e+01
4.17540492e+01 4.19373818e+01 4.21211160e+01 4.23052518e+01
4.24897892e+01 4.26747282e+01 4.28600688e+01 4.30458111e+01
4.32319549e+01 4.34185003e+01 4.36054474e+01 4.37927960e+01
4.39805463e+01 4.41686981e+01 4.43572516e+01 4.45462066e+01
4.47355633e+01 4.49253216e+01 4.51154815e+01 4.53060429e+01
4.54970060e+01 4.56883707e+01 4.58801370e+01 4.60723049e+01
4.62648744e+01 4.64578456e+01 4.66512183e+01 4.68449926e+01
4.70391685e+01 4.72337460e+01 4.74287252e+01 4.76241059e+01
4.78198883e+01 4.80160722e+01 4.82126578e+01 4.84096449e+01
4.86070337e+01 4.88048241e+01 4.90030161e+01 4.92016096e+01
4.94006048e+01 4.96000016e+01 4.97998000e+01 5.00000000e+01]
```

xv,yv = np.mesgrid(x,y)

[ ]:
```
xv,yv = np.meshgrid(x,y)
```

[ ]:
```
z = xv**2 + yv**2
```

[ ]:
```
z
```

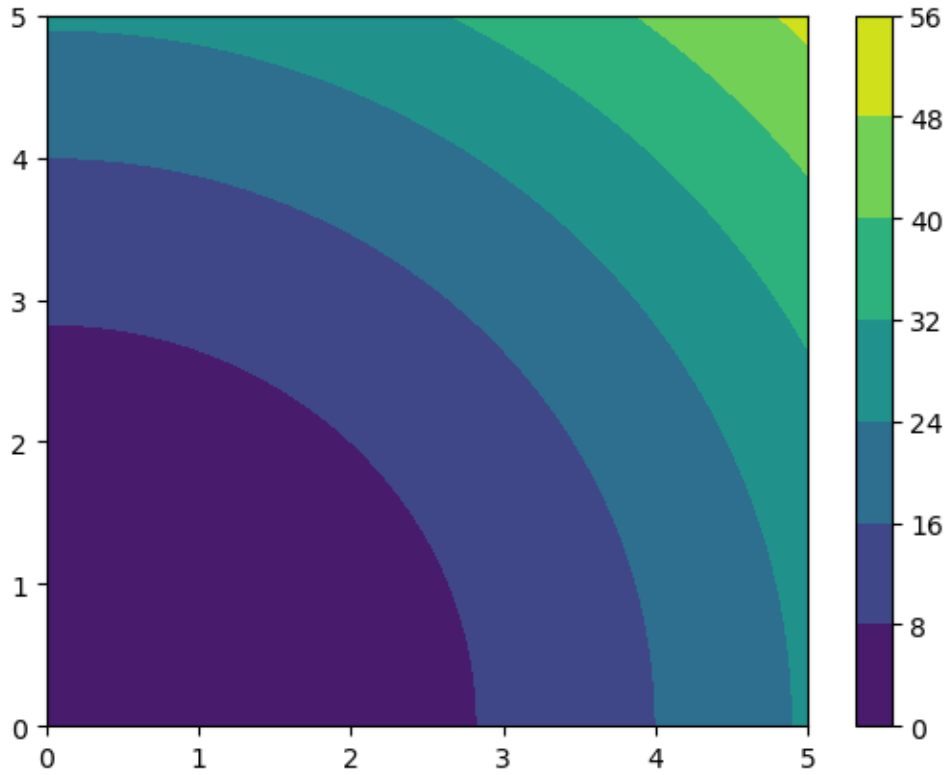
[ ]:
```
array([[0.00000000e+00, 1.00401203e-04, 4.01604813e-04, …,
        2.48000008e+01, 2.48999000e+01, 2.50000000e+01],
       [1.00401203e-04, 2.00802406e-04, 5.02006016e-04, …,
        2.48001012e+01, 2.49000004e+01, 2.50001004e+01],
       [4.01604813e-04, 5.02006016e-04, 8.03209626e-04, …,
        2.48004024e+01, 2.49003016e+01, 2.50004016e+01],
       …,
       [2.48000008e+01, 2.48001012e+01, 2.48004024e+01, …,
        4.96000016e+01, 4.96999008e+01, 4.98000008e+01],
       [2.48999000e+01, 2.49000004e+01, 2.49003016e+01, …,
```

```
 4.96999008e+01, 4.97998000e+01, 4.98999000e+01],
[2.50000000e+01, 2.50001004e+01, 2.50004016e+01, …,
 4.98000008e+01, 4.98999000e+01, 5.00000000e+01]])
```

[ ]:
```
plt.contourf(xv,yv,z,level=100)
plt.colorbar()
```

```
/tmp/ipykernel_606566/3686443545.py:1: UserWarning: The following kwargs were
not used by contour: 'level'
  plt.contourf(xv,yv,z,level=100)
```

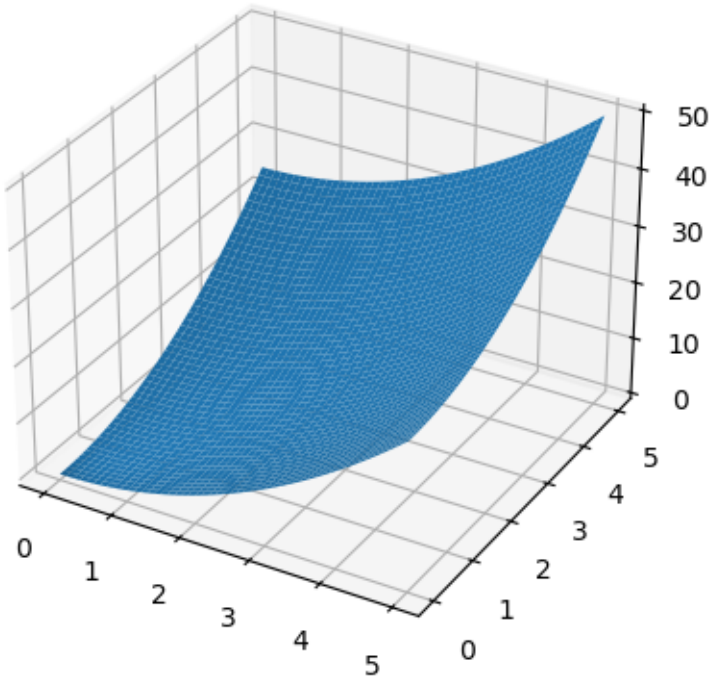
[ ]:
```
<matplotlib.colorbar.Colorbar at 0x7feaea6f3410>
```



[ ]:
```
from mpl_toolkits import mplot3d
```

[ ]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_surface(xv,yv,z)
```

[ ]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x7feae7802600>
```



1.2 Algebric Calculation

[ ]:
```
A = np.array([[3,2,1],[5,-5,4],[6,0,1]])
print(A)
```

```
[[ 3  2  1]
 [ 5 -5  4]
 [ 6  0  1]]
```

[ ]:
```
b1 = np.array([1,2,3])
b2 = np.array([-1,2,-5])
```

[ ]:
```
A@b1 #multiplication of matrix with vector
```

[ ]:
```
array([10,  7,  9])
```

[ ]:
```
A.T
```

[ ]:
```
array([[ 3,  5,  6],
       [ 2, -5,  0],
       [ 1,  4,  1]])
```

[ ]:
```
np.dot(b1,b2)
```

[ ]:
```
-12
```

[ ]:
```
np.cross(b1,b2)
```

[ ]:
```
array([-16,   2,   4])
```

[ ]:
```
eg = np.linalg.eig(A)
eg
```

[ ]:
```
EigResult(eigenvalues=array([ 5.98847677, -1.66137965, -5.32709712]),
eigenvectors=array([[-0.55522613, -0.36439757,  0.25391074],
       [-0.49573499,  0.43853605, -0.93677764],
       [-0.66781043,  0.82152331, -0.24078411]]))
```

[ ]:
```
eg[0]
```

[ ]:
```
array([ 5.98847677, -1.66137965, -5.32709712])
```

[ ]:
```
eg[1]
```

[ ]:
```
array([[-0.55522613, -0.36439757,  0.25391074],
       [-0.49573499,  0.43853605, -0.93677764],
       [-0.66781043,  0.82152331, -0.24078411]])
```

[ ]:
```
eigen_vec_2 = eg[1][:,1]
eigen_vec_2
```

[ ]:
```
array([-0.36439757,  0.43853605,  0.82152331])
```

[ ]:
```
eg_val, eg_vec = np.linalg.eig(A)
```

[ ]:
```
eg_val
```

[ ]:
```
array([ 5.98847677, -1.66137965, -5.32709712])
```

[ ]:
```
eg_vec
```

[ ]:
```
array([[-0.55522613, -0.36439757,  0.25391074],
       [-0.49573499,  0.43853605, -0.93677764],
       [-0.66781043,  0.82152331, -0.24078411]])
```

[ ]:
```
eg_vec[:,0]
```

[ ]:
```
array([-0.55522613, -0.49573499, -0.66781043])
```

$$3x + 2y + z = 4$$

$$5x - 5y + 4z = 3$$

$$6x + z = 0$$

[ ]:
```
A = np.array([[3,2,1],[5,-5,4],[6,0,1]])
Y = np.array([4,3,0])
x = np.linalg.solve(A,Y)# solve the above equation
```

[ ]:
```
x
```

[ ]:
```
array([-0.49056604,  1.26415094,  2.94339623])
```

[ ]:
```
a = np.diag([1,2,3,4])
print(a)
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
```

[ ]: