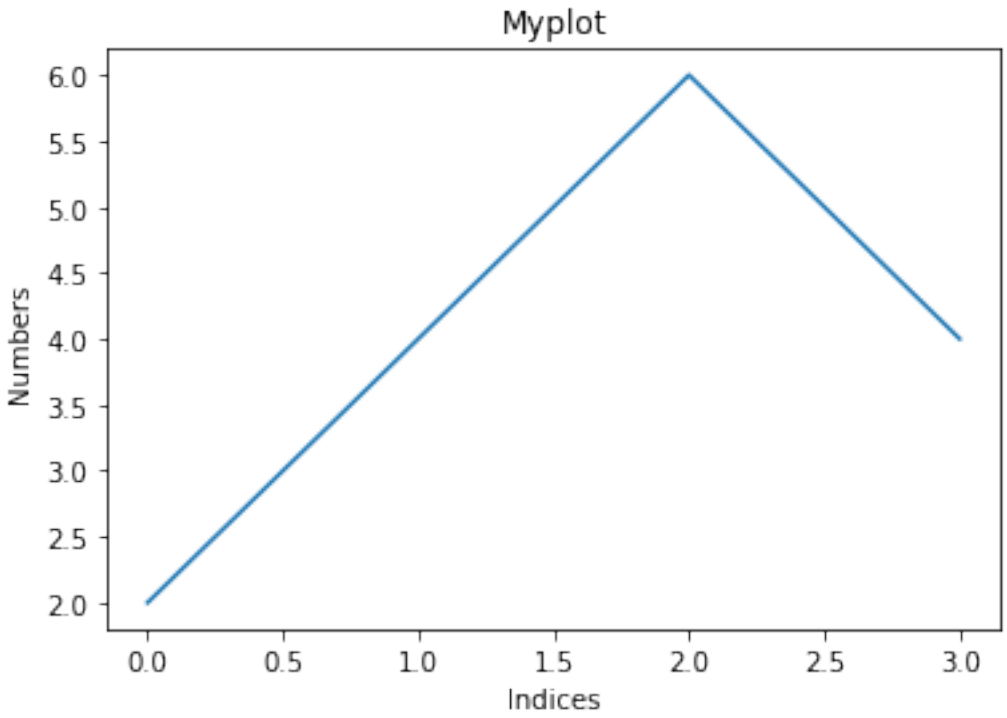
6.1 Getting started with Matplotlib

January 30, 2019

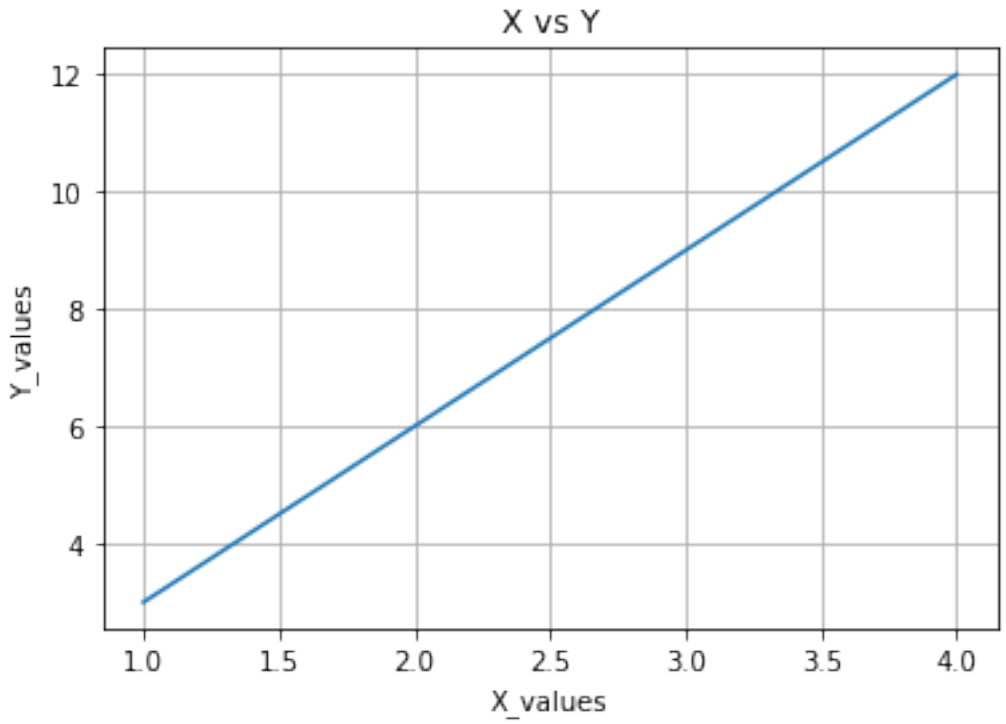
In [1]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
"""
When we dont give x points then matplotlib will take indices as x axis , like here:
the points will become [(0,2),(1,4),(2,6),(3,4)]
"""
plt.plot([2,4,6,4])
plt.ylabel("Numbers")
plt.xlabel("Indices")
plt.title("Myplot")
plt.show()
```



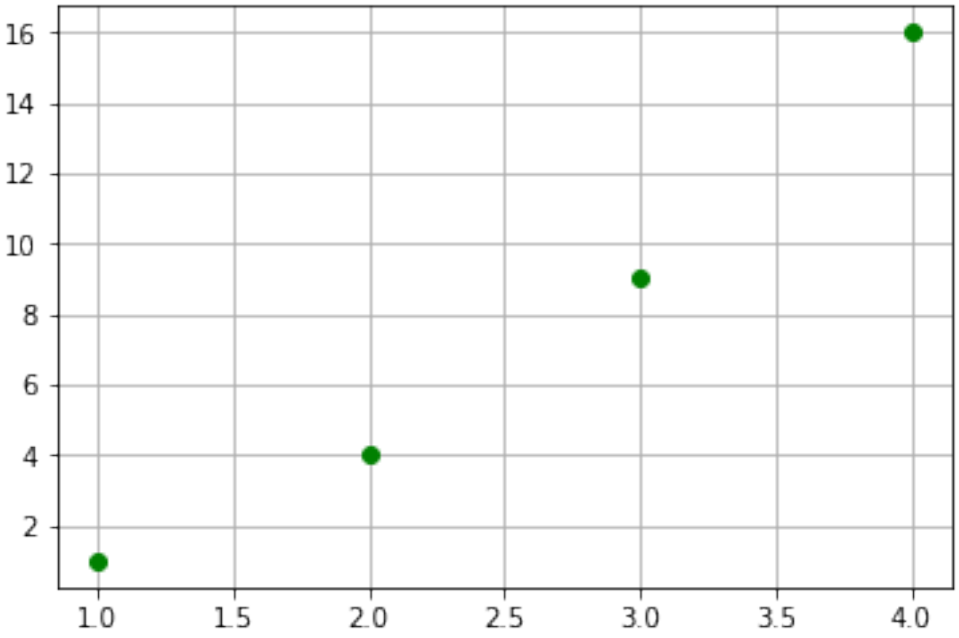
1 Plot x vs y

In [12]:
```
"""
When we give 2 arrays as an argument to matplotlib the first one become x axis
and second one become y axis
like [1,2,3,4],[3,6,9,12]
So ,x-axis=[1,2,3,4] and y-axis=[3,6,9,12]
"""
plt.plot([1,2,3,4],[3,6,9,12])
plt.ylabel('Y_values')
plt.xlabel('X_values')
plt.grid()  # turning grid on
plt.title("X vs Y")
plt.show()
```

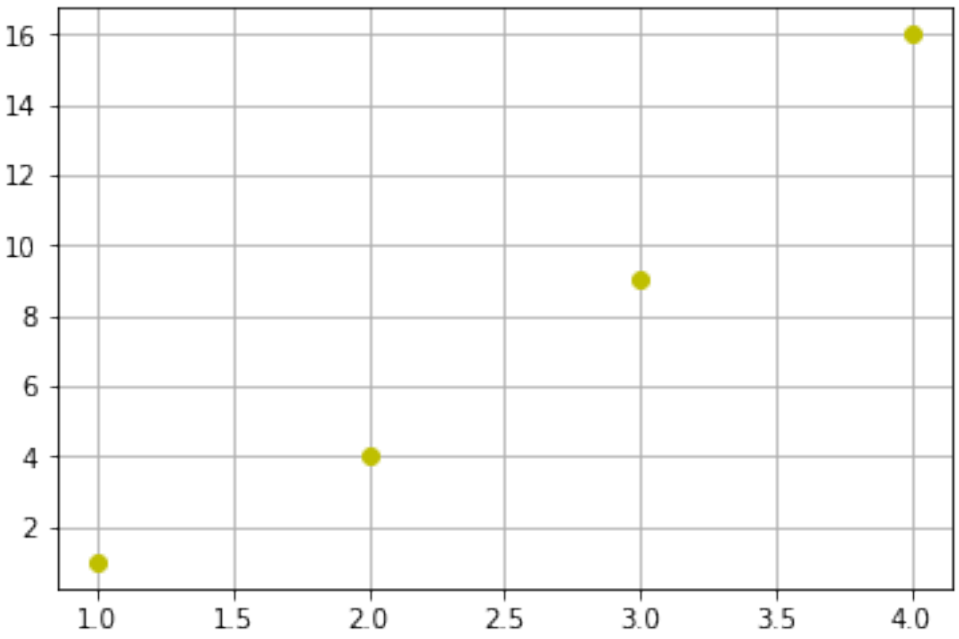


2 Plotting o-shaped plot istead of line

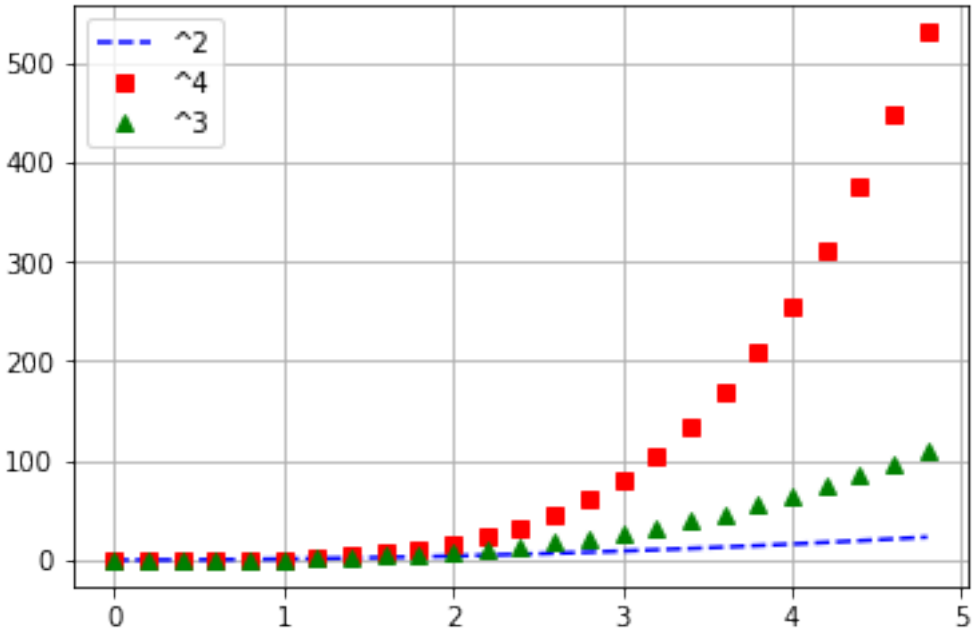
In [18]:
```
plt.plot([1,2,3,4],[1,4,9,16],'go') # g stands for color green and o for dot
plt.grid()
plt.show()
```



In [19]:
```
plt.plot([1,2,3,4],[1,4,9,16],'yo')
plt.grid()
plt.show()
```



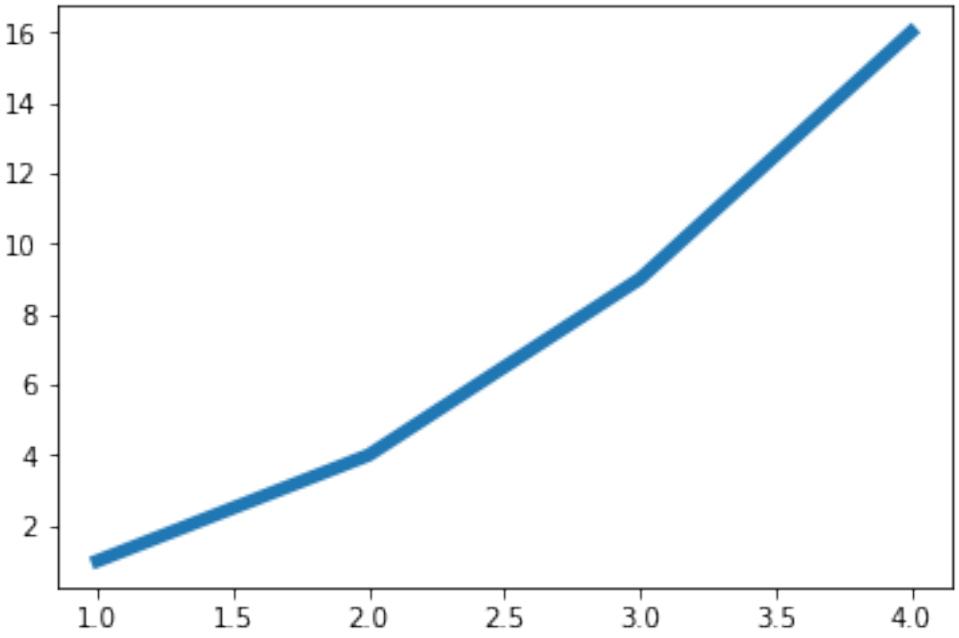
```
In [20]: import numpy as np

In [22]: t=np.arange(0.,5.,0.2) # start , end(exclusive),jumppoint
         t

Out[22]: array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. , 2.2, 2.4,
                2.6, 2.8, 3. , 3.2, 3.4, 3.6, 3.8, 4. , 4.2, 4.4, 4.6, 4.8])

In [30]: # blue dashed , red squares and green triangles
         plt.plot(t,t**2,'b--',label='^2')
         plt.plot(t,t**4,'rs',label='^4')
         plt.plot(t,t**3,'g^',label='^3')
         plt.grid()
         plt.legend() # add legend based on line labels
         plt.show()
```



3 Controlling line properties

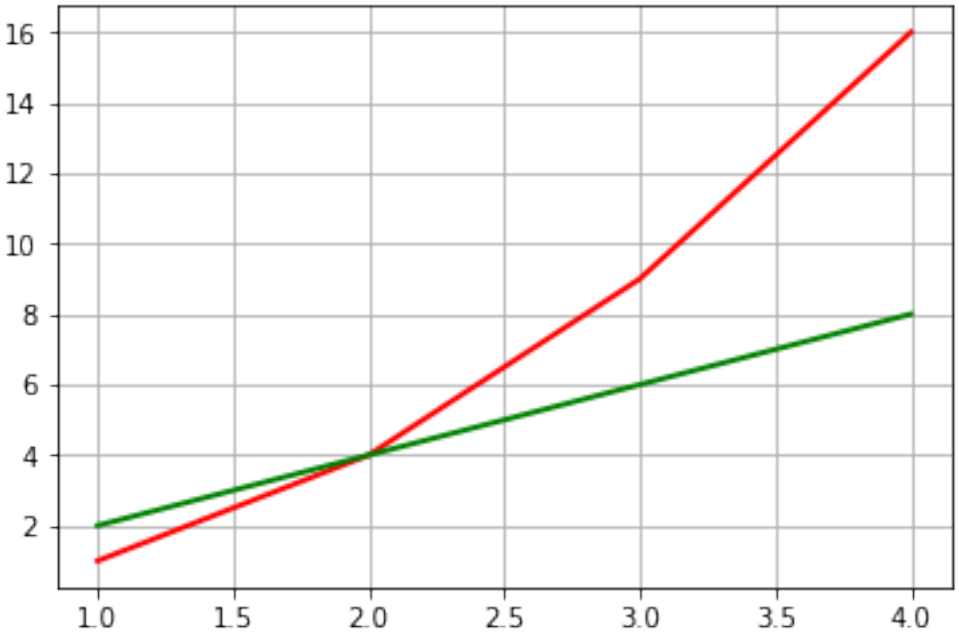
3.0.1 use keyword args

```
In [33]: x=[1,2,3,4]
         y=[1,4,9,16]
         plt.plot(x,y,linewidth=5.0)
         plt.show()
```



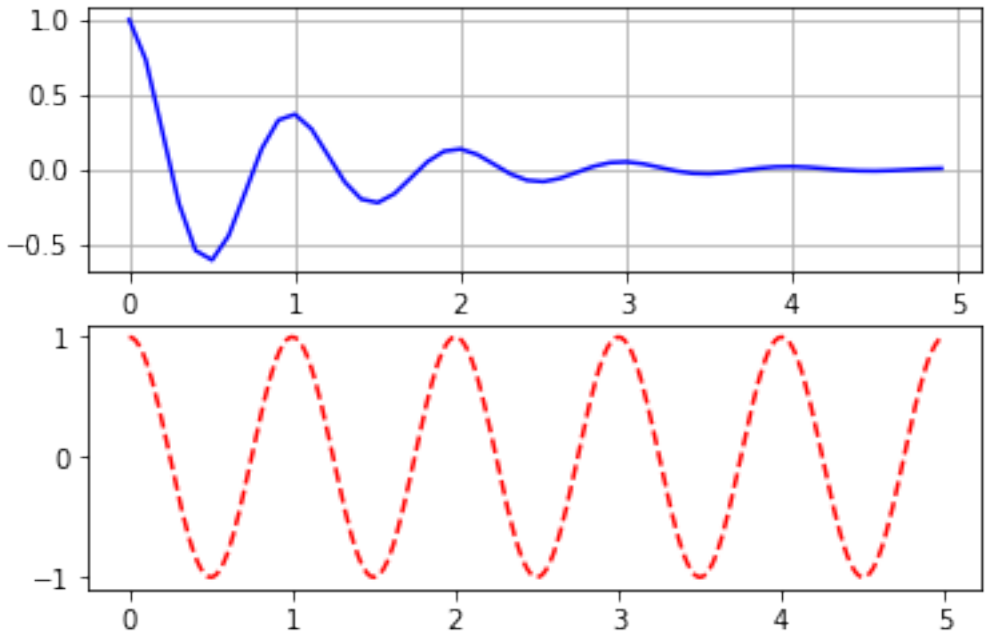
3.0.2 Using setp() – set properties

```
In [37]: x1=[1,2,3,4]
         y1=[1,4,9,16]
         x2=[1,2,3,4]
         y2=[2,4,6,8]
         lines=plt.plot(x1,y1,x2,y2)

         # use tthe keyword args
         plt.setp(lines[0],color='r',linewidth=2.0)
         # or we can use the above properties in matlab styles
         plt.setp(lines[1],'color','g','linewidth',2.0)
         plt.grid()
         plt.show()
```



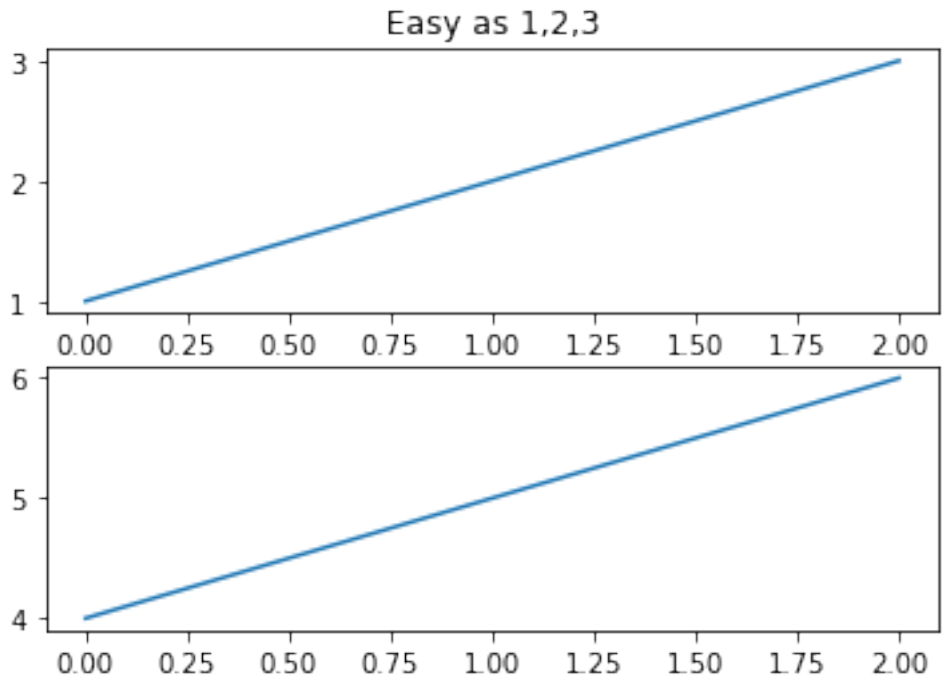
4 Working with multiple figures and axes

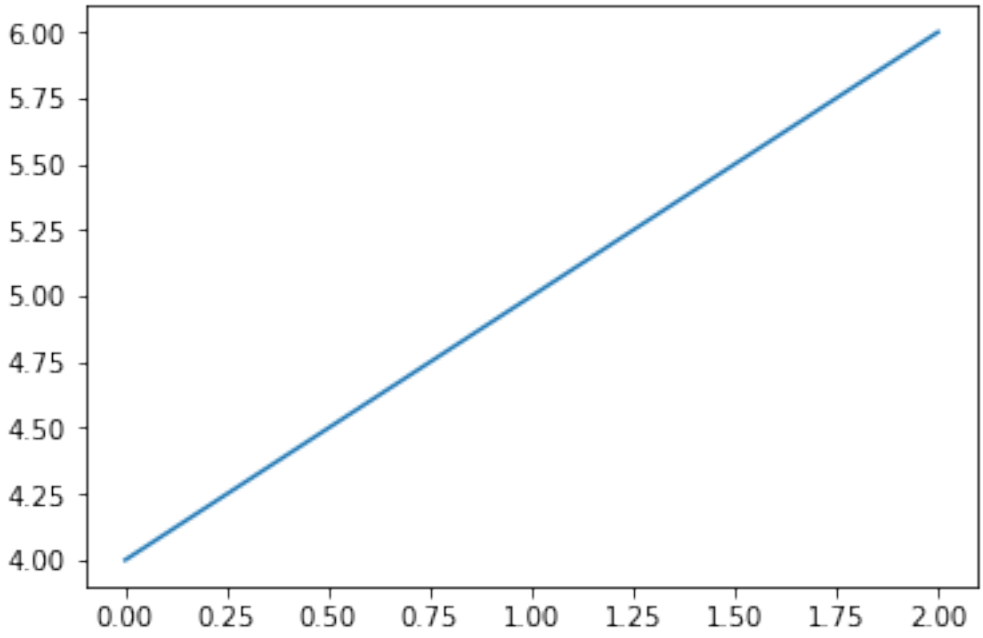
```
In [44]: def f(t):
             return np.exp(-t) * np.cos(2*np.pi*t)

         t1=np.arange(0.0,5.0,0.1)
         t2=np.arange(0.0,5.0,0.02)
         plt.figure(1) # this will create a figure (like a horizon)
         """the subplot () command specifies numrows, numcols
         figure number where figure number ranges from 1 to numrows*numcols
         """
         plt.subplot(211)
         plt.grid()
         plt.plot(t1,f(t1),'b-')

         plt.subplot(212)
         plt.plot(t2,np.cos(2*np.pi*t2),'r--')
         plt.show()
```



In [47]:
```
plt.figure(1)  # the first image
plt.subplot(211) # the first subplot in the first figure
plt.plot([1,2,3])
plt.subplot(212) #the second subplot in the first figure
plt.plot([4,5,6])

plt.figure(2)  # a second figure
plt.plot([4,5,6]) # creates a subplot (111) by default
plt.figure (1) # figure 1 current; subplot (212) still current
plt.subplot(211) # make subplot(211 ) in figure1 current
plt.title('Easy as 1,2,3') # subplot 211 titlw
plt.show()
```

```
/run/media/atif/New Volume/virtual_env/data_science_3/lib/python3.7/site-packages/matplotlib/fi
Adding an axes using the same arguments as a previous axes currently reuses the earlier instanc
  "Adding an axes using the same arguments as a previous axes "
```


Easy as 1,2,3
3
2
1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
6
5
4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


6.00
5.75
5.50
5.25
5.00
4.75
4.50
4.25
4.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

In [ ]: