toPy

May 28, 2020

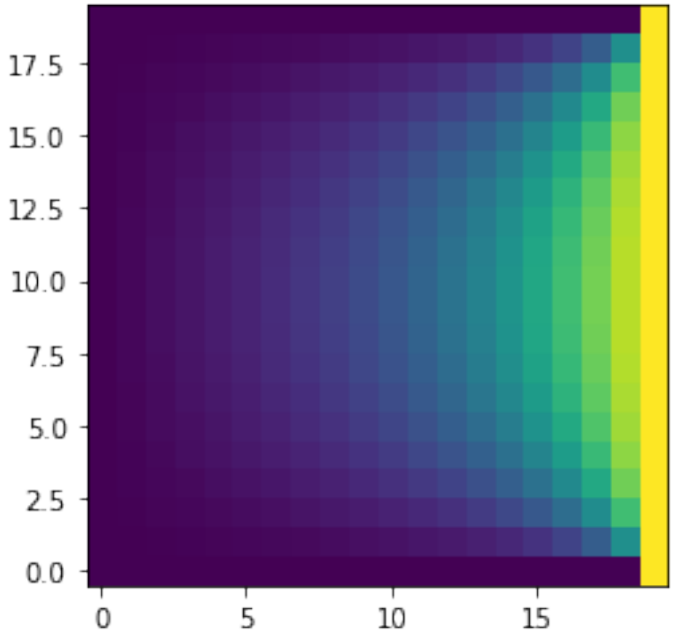
1 Topics:

The relaxation problem
A code to find π number
Some Beneficial Notes in Python:
Save array to a text file and load it
Pass variables to string by %
The Enumerate Function
Some notes in Matplotlib library
Random Library

2 The relaxation problem

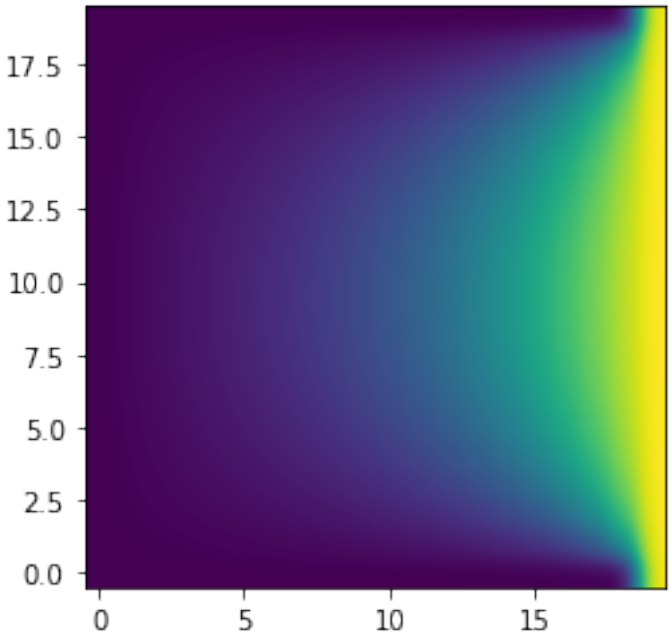
[2]:
```
import numpy as np

N= 20
v= np.random.random ([N, N])
v[:,0]= 0
v[0,:]= 0
v[-1,:]= 0
v[:,-1]= 1
lx= len (v[0])
ly= len (v)
"""
====================
  RELAX THE POTENTIAL
====================
"""
for _ in range (lx**2):
    for i in range (1,lx-1):
        for j in range (1,ly-1):
            v[j,i]= 0.25 * (v[j-1,i] + v[j,i+1] + v[j+1,i] + v[j,i-1])

"""
====================
  VISUALIZE THE POTENTIAL
```

```
==================
"""
import matplotlib.pyplot as plt
plt.imshow (v, origin= 'lower')
plt.show ()
```



2.0.1 or better visualization with using imshow options

[3]:
```
plt.imshow (v, interpolation= 'gaussian', origin= 'lower')
plt.show ()
```



2.1 Save array to a text file and load it:

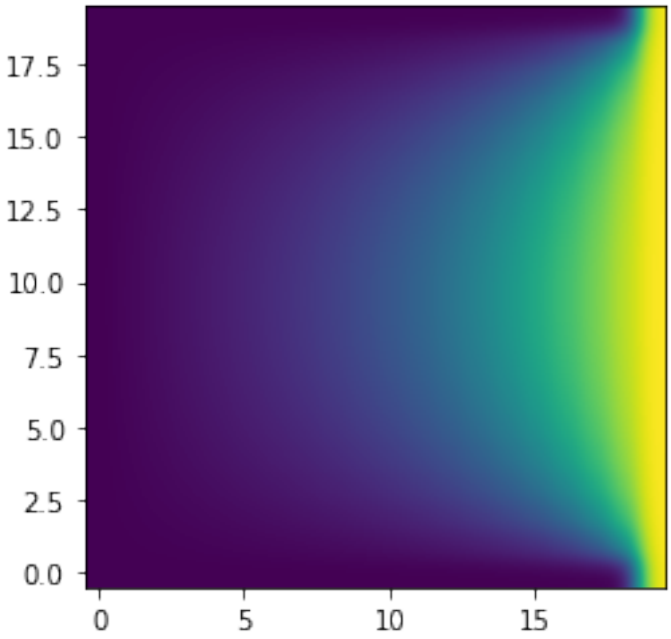
[5]:
```
file= open ('v.txt', 'w')
for i in range (lx):
    for j in range (ly):
        file.write (str (v[i,j]) + '  ')
    file.write ('\n')
file.close ()
```

or :

[6]:
```
np.savetxt('v1.txt', v, delimiter=' ')
```

[7]:
```
vv= np.loadtxt ('v1.txt')
plt.imshow (vv, interpolation= 'gaussian', origin= 'lower')
```

[7]:
```
<matplotlib.image.AxesImage at 0x7fc555d80b00>
```



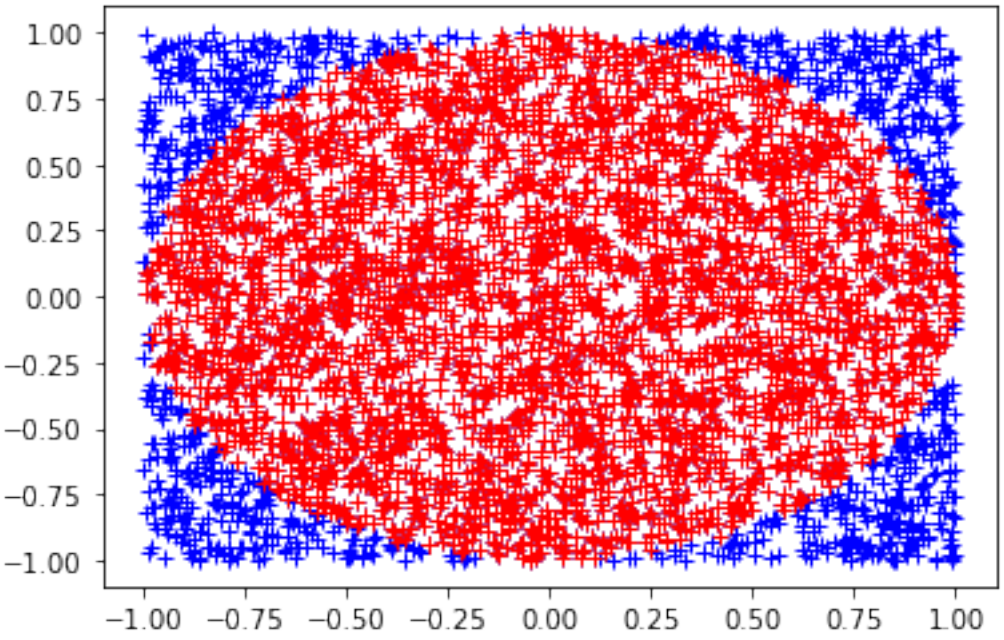
3 A code to find π number

$pi * r^2 = 4 * r^2$

[8]:
```
import numpy as np
import pylab as plt
```

[26]:
```
N= 5000
X= np.random.random (N)*2 - 1
Y= np.random.random (N)*2 - 1
# plt.plot (X, Y, '+')
Xcirc= []
Ycirc= []
counter= 0

# for i, x in enumerate (X):     #first output: the indices (i), second output:␣
 ↪the elements (x)
for i in range (N):
#     print (i, x)
    if (X[i]**2 + Y[i]**2) < 1:
        counter+= 1
        Xcirc.append (X[i])
        Ycirc.append (Y[i])

plt.plot (X, Y, 'b+')
plt.plot (Xcirc, Ycirc, 'r+')
```

[26]: `[<matplotlib.lines.Line2D at 0x7fc555af8fd0>]`



** better visualization comes further **

[10]:
```
Pi= 4*counter/N
print (Pi)
```

```
3.1336
```

4 Some Beneficial Notes in Python

4.1 Pass variables to string by % :

for more detals you can see **this link**

[24]:
```
Number= 100
st= 'the text'
```

[25]:
```
print ('this is a text with some variables like integer %d and floating points␣
 ↪%.1f and %.10f'  % (3, 3, 3))
print ('\n', 'this adds string: \'%s \' and this adds the variabel N= %d' %␣
 ↪(st, Number))
```

```
this is a text with some variables like integer 3 and floating points 3.0 and
3.0000000000

 this adds string: 'the text ' and this adds the variabel N= 100
```

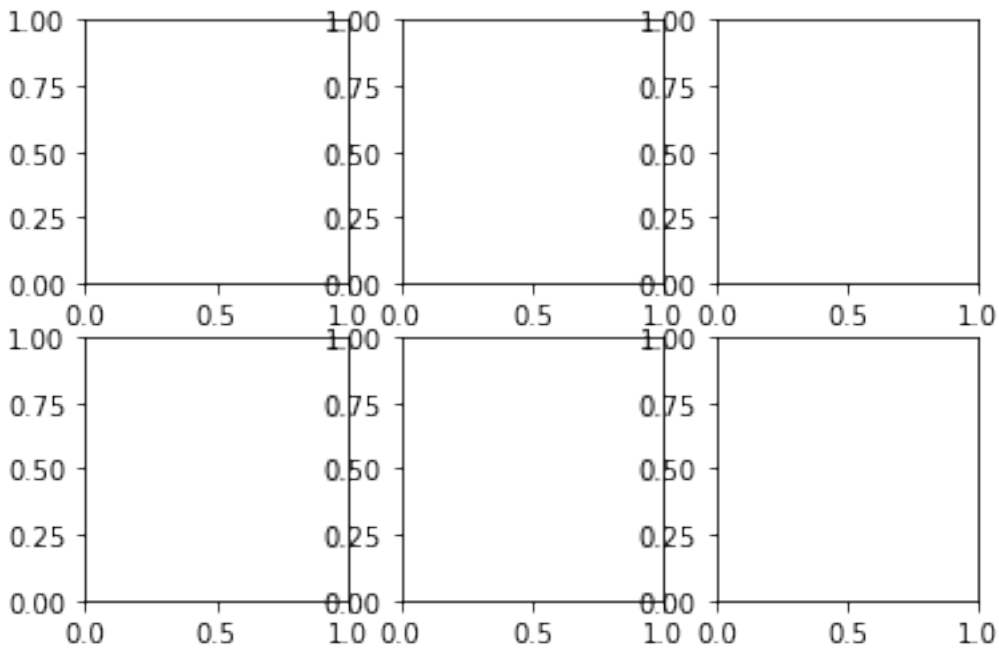
4.2 Enumerate Function:

[15]:
```
for i, x in enumerate (X[:10]):      #first output: the indices (i), second␣
 ↪output: the elements (x)
    print ('the index of element %f is %d' %(x, i))
```

```
the index of element -0.457814 is 0
the index of element -0.895211 is 1
the index of element 0.236677 is 2
the index of element 0.718637 is 3
the index of element -0.738296 is 4
the index of element -0.595272 is 5
the index of element 0.970748 is 6
the index of element 0.420357 is 7
the index of element 0.134271 is 8
the index of element -0.540252 is 9
```

4.3 Some notes in Matplotlib library

[16]:
```
import matplotlib.pyplot as plt
fig, axes= plt.subplots (2, 3)
```



[17]:
```
# see whats in axes
print (type(axes))
print (type(axes[0,0]))
```

```
<class 'numpy.ndarray'>
<class 'matplotlib.axes._subplots.AxesSubplot'>
```

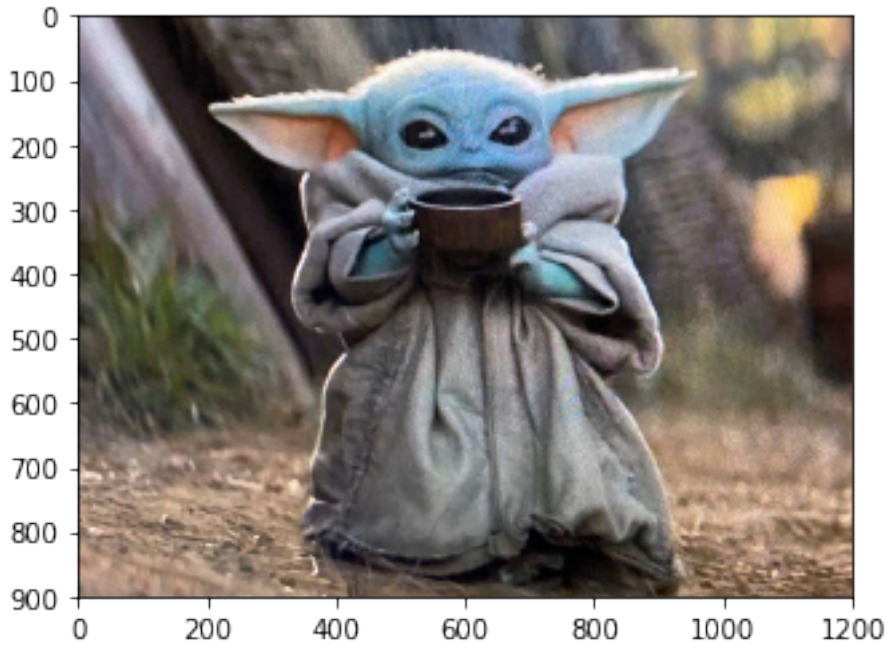
[18]:
```
axes.shape
```

[18]:
```
(2, 3)
```

4.3.1 Open an image with matplotlib

[19]:
```
from matplotlib.image import imread   #to open image file
```

[37]:
```
image_path= 'C:/Users/Farzin/Documents/toPython/babyYoda.jpeg'
image = imread(image_path )
plt.imshow(image)
plt.show ()
```



[38]:
```
type (image)
```

[38]:
```
numpy.ndarray
```

[39]:
```
np.shape (image)
```

[39]:
```
(900, 1200, 3)
```

Manipulate image data:

[45]:
```
image [100]= 0
```

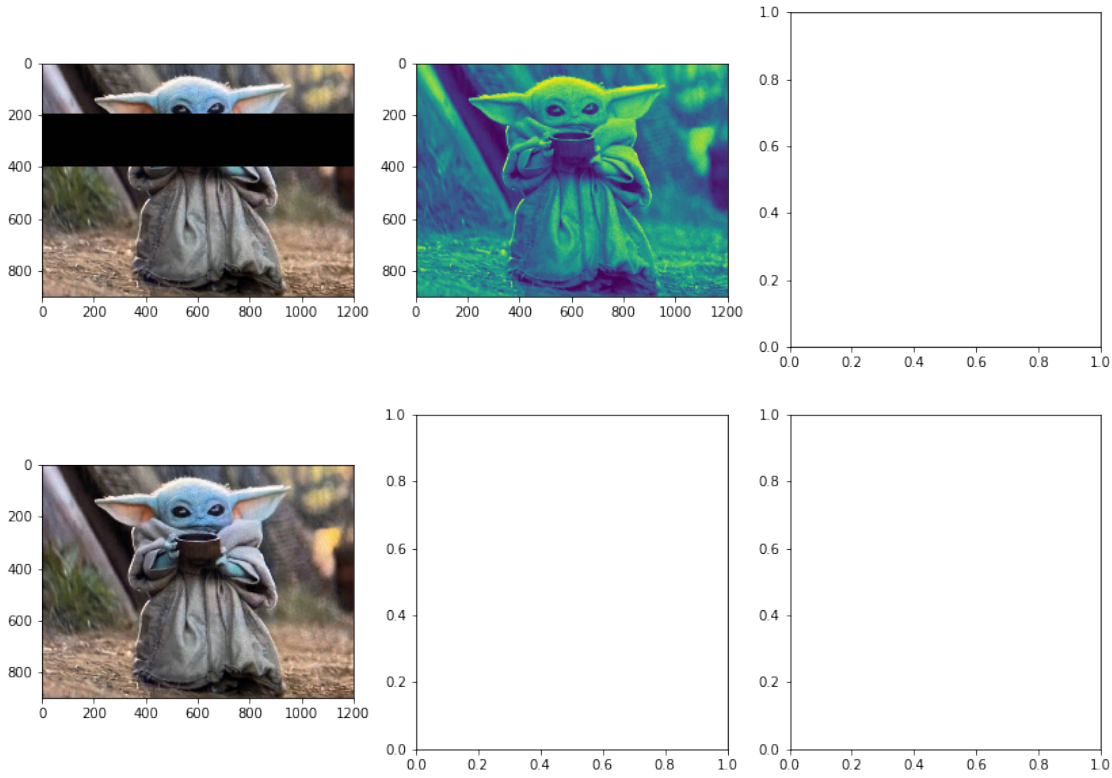
```
↪----------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-45-b92d90dc2343> in <module>()
   ----> 1 image [100]= 0


       ValueError: assignment destination is read-only
```

[100]:
```
data= image.copy ()
```

[50]:
```
data1= data.copy()
data1[200:400, :,:]=1
```

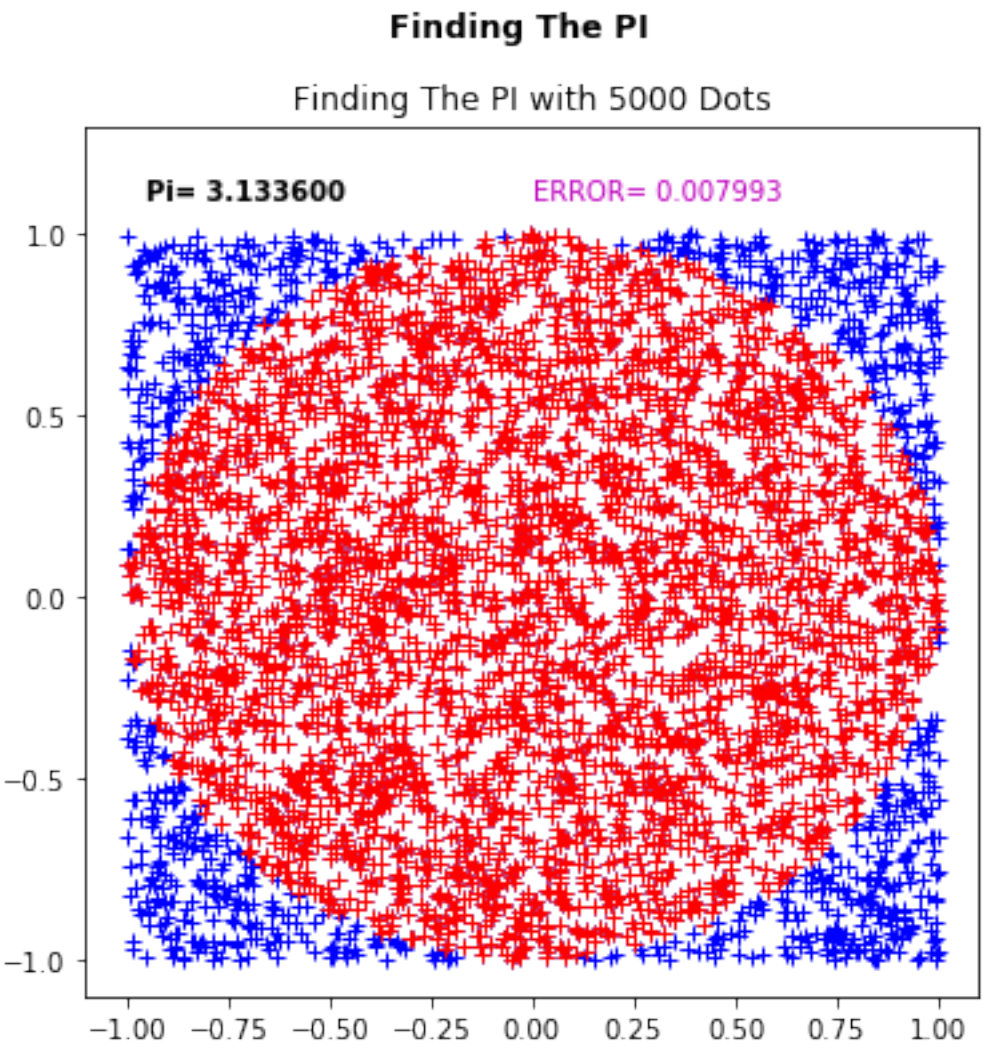
[106]:
```
fig, axes= plt.subplots (2, 3, figsize= [14,10])

axes[0,1].imshow(data[:,:,1])
axes[1,0].imshow(data)
axes[0,0].imshow(data1)
```

[106]:
```
<matplotlib.image.AxesImage at 0x24ac00ec6a0>
```



A better visualization for findig π problem

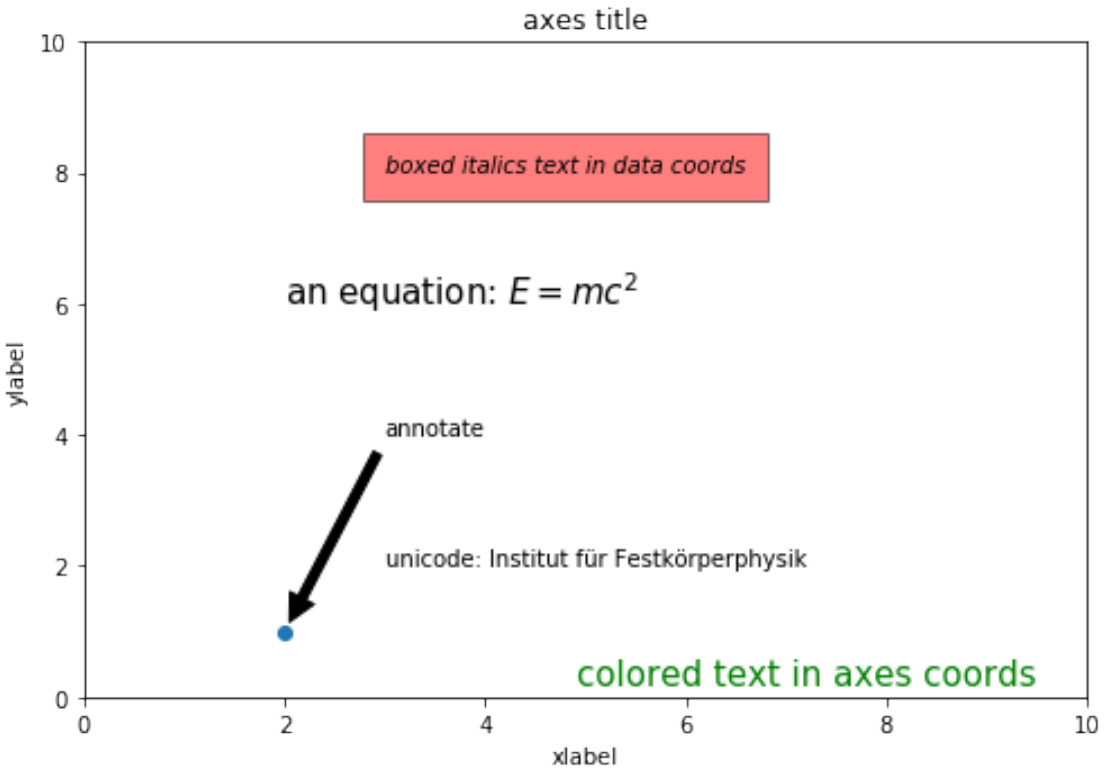
[27]:
```
fig= plt.figure (figsize= (6,6))
plt.plot (X, Y, 'b+')
plt.plot (Xcirc, Ycirc, 'r+')

plt.suptitle ("Finding The PI", fontweight='bold')
plt.axis ([-1.1 ,1.1, -1.1,1.3])     #coordinates or limits of the borders:␣
 ↪[left x, right x, bottom y, upper y]

t1 = plt.title ('Finding The PI with %d Dots' % N, color ='k')

t2 = plt.text(-.95, 1.1, '',color ='k', fontweight='bold')
t3 = plt.text(0, 1.1, '',color ='m')
t2.set_text('Pi= %.6f ' % Pi)
t3.set_text('ERROR= %.6f ' % abs(np.pi - Pi))

plt.show ()
```



4.3.2 Use some of matplotlib options

[28]:
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize= [8, 6])
fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold')

ax = fig.add_subplot(111)
fig.subplots_adjust(top=0.8)
ax.set_title('axes title')

ax.set_xlabel('xlabel')
ax.set_ylabel('ylabel')
```

```
ax.text(3, 8, 'boxed italics text in data coords', style='italic',
        bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})

ax.text(2, 6, r'an equation: $E=mc^2$', fontsize=15)

ax.text(3, 2, u'unicode: Institut f\374r Festk\366rperphysik')

ax.text(0.95, 0.01, 'colored text in axes coords',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax.transAxes,
        color='green', fontsize=15)


ax.plot([2], [1], 'o')
ax.annotate('annotate', xy=(2, 1), xytext=(3, 4),
            arrowprops=dict(facecolor='black', shrink=0.05))

ax.axis([0, 10, 0, 10])

plt.show()
```



4.4 Random Library

[29]:
```
import random

print (random.random())     # Generate a pseudo-random number between 0 and 1.
print (random.randint(1, 100))    # Pick a random number between 1 and 100.
print (random.uniform(1, 10))        #To generate a random floating point␣
 ↪number between 1 and 10 you can use the uniform() function
```

```
0.19142852177157565
85
2.0204370373769085
```

[30]:
```
np.random.random()*10
```

[30]:
```
7.179347303841448
```

[31]:
```
items = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
random.shuffle(items)      #shuffle a list

x = random.sample(items,  1)   # Pick a random item from the list
print (x[0])
```

```
7
```

[32]:
```
y = random.sample(items, 4)    # Pick 4 random items from the list
print (y, '\n')

# difference with np.random.choice
print ('this is from np.random:', np.random.choice (items, 20))
```

```
[1, 10, 9, 6]

this is from np.random: [ 2  1  8  7  1 10  6  2  9  4  7  5  8  6  7  4  4 10
6  3]
```

[ ]:
```

```