NumPy (Mr. P Solver)

Video Link: [https://youtu.be/DcfYgePyedM (https://youtu.be/DcfYgePyedM)](https://youtu.be/DcfYgePyedM)

Codes:
https://github.com/lukepolson/youtube_channel/blob/main/Python%20Tutorial%20Series/numpy_essentia
(https://github.com/lukepolson/youtube_channel/blob/main/Python%20Tutorial%20Series/numpy_essenti

NumPy means NUMERICAL PYTHON.

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

Creating Arrays

In [2]:
```
ar1= np.array([7,8,5,8,1,4])
ar2= np.zeros(6)
ar3= np.ones(5)
```

In [3]:
```
ar1, ar2, ar3
```

Out[3]:
```
(array([7, 8, 5, 8, 1, 4]),
 array([0., 0., 0., 0., 0., 0.]),
 array([1., 1., 1., 1., 1.]))
```

In [4]:
```
ar4= np.random.random(10)
ar5= np.random.randn(10)
```

In [5]:
```
ar4, ar5
```

Out[5]:
```
(array([0.43778108, 0.59564584, 0.56025243, 0.79524882, 0.65291272,
        0.03090156, 0.89890837, 0.09295612, 0.85845597, 0.02159939]),
 array([ 1.31824218, -1.12601269, -0.75309667,  0.92722903, -0.65351709,
        -0.81793512,  0.87573291,  0.07681538,  2.09646925,  0.21637756]))
```

In [6]:
```
ar6= np.linspace(0,2,10)
ar7= np.arange(0,5,0.4)
```

In [7]:
```
ar6, ar7
```

Out[7]:
```
(array([0.        , 0.22222222, 0.44444444, 0.66666667, 0.88888889,
        1.11111111, 1.33333333, 1.55555556, 1.77777778, 2.        ]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. , 4.4, 4.8]))
```

Array Operations

In [8]:
```
2/ar1+4
```

Out[8]:
```
array([4.28571429, 4.25      , 4.4       , 4.25      , 6.        ,
       4.5       ])
```

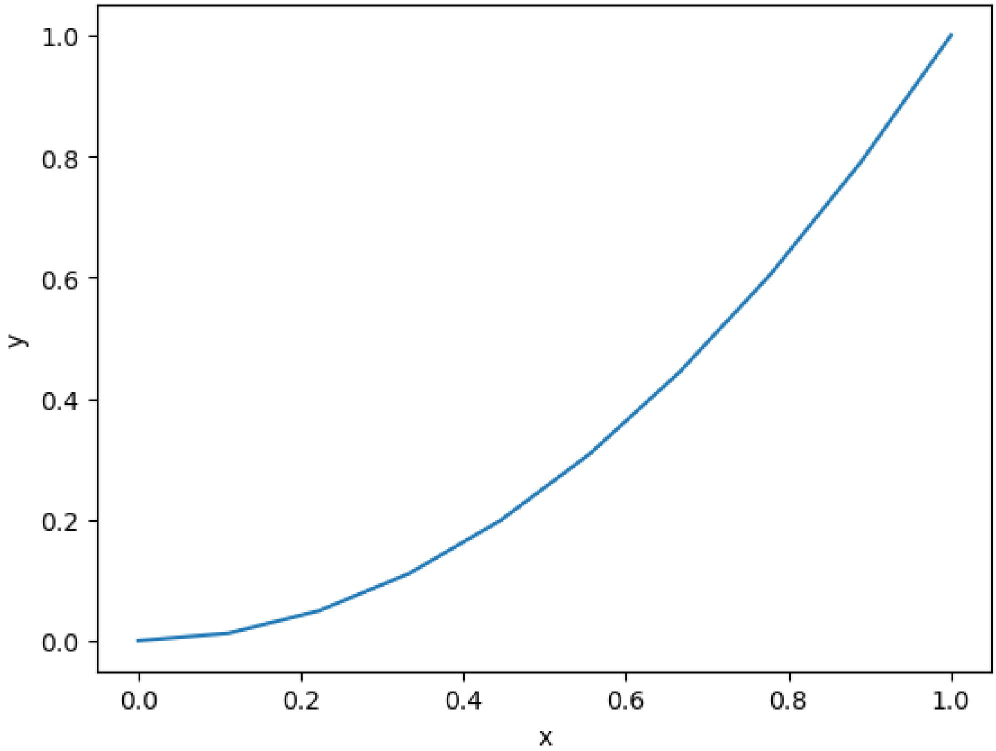
In [9]:
```
ar1>5
```

Out[9]:
```
array([ True,  True, False,  True, False, False])
```

In [10]:
```
x= np.linspace(0,1,10)
y= x**2
```

In [11]:
```
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
```

Out[11]:
```
Text(0, 0.5, 'y')
```

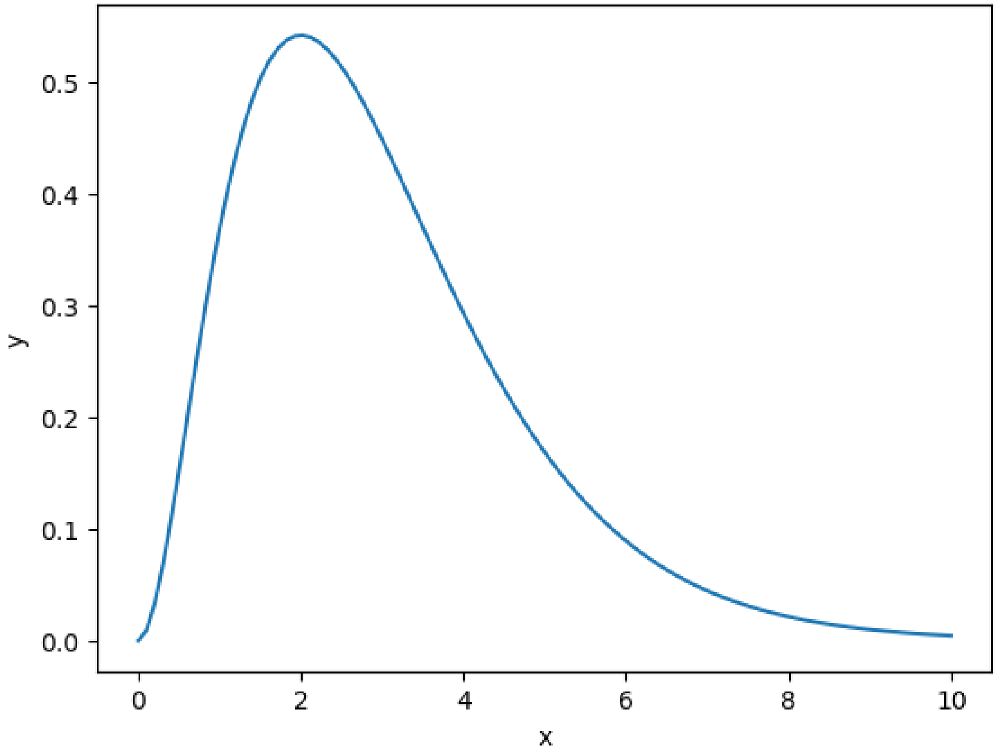


In [12]:
```
def f(x):
    return x**2 * np.exp(-x)

x= np.linspace(0,10,100)
y= f(x)
```

In [13]:
```
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
```

Out[13]: `Text(0, 0.5, 'y')`



list of many mathematical functions -

[https://numpy.org/doc/stable/reference/routines.math.html (https://numpy.org/doc/stable/reference/routines.math.html)](https://numpy.org/doc/stable/reference/routines.math.html)

array indexing/ slicing

In [14]:
```
arr1= np.array([9,8,6,6,1,2,7])
```

In [15]:
```
arr1[2:]
```

Out[15]: `array([6, 6, 1, 2, 7])`

In [16]:
```
arr1[:-2]
```

Out[16]: `array([9, 8, 6, 6, 1])`

In [17]:
```
arr1[::3]
```

Out[17]: `array([9, 6, 7])`

In [18]:
```
arr1>6
```

Out[18]: `array([ True,  True, False, False, False, False,  True])`

In [19]:
```python
arr1[arr1>6]
```
Out[19]:
```
array([9, 8, 7])
```
In [20]:
```python
arr1%3==0
```
Out[20]:
```
array([ True, False,  True,  True, False, False, False])
```
In [21]:
```python
arr1%3
```
Out[21]:
```
array([0, 2, 0, 0, 1, 2, 1], dtype=int32)
```
In [22]:
```python
arr1[arr1%3==0]
```
Out[22]:
```
array([9, 6, 6])
```
In [23]:
```python
l= lambda s: s[0]
l([4,3,7]), l('Suman')
```
In [25]:
```python
GOATS= np.array(['Messi', 'Maradona', 'Pele', 'Ronaldo'])
fst_ltr_m= np.vectorize(lambda s: s[0])(GOATS)=='M'
```
In [26]:
```python
np.vectorize(lambda s: s[0])(GOATS)
```
Out[26]:
```
array(['M', 'M', 'P', 'R'], dtype='<U1')
```
In [27]:
```python
fst_ltr_m
```
Out[27]:
```
array([ True,  True, False, False])
```
In [28]:
```python
GOATS[fst_ltr_m]
```
Out[28]:
```
array(['Messi', 'Maradona'], dtype='<U8')
```
calculus/ statistical functions

In [29]:
```python
a1= 3* np.random.randn(100) +15
```
mean, std dev and percenties of array

In [30]:
```python
np.mean(a1)
```
Out[30]:
```
14.389455665014271
```
In [31]:
```python
np.std(a1)
```
Out[31]:
```
3.314751652961054
```
In [32]:
```python
np.percentile(a1,80)
```
Out[32]:
```
16.64164472024565
```
read question 1 to understand

integral and derivatives

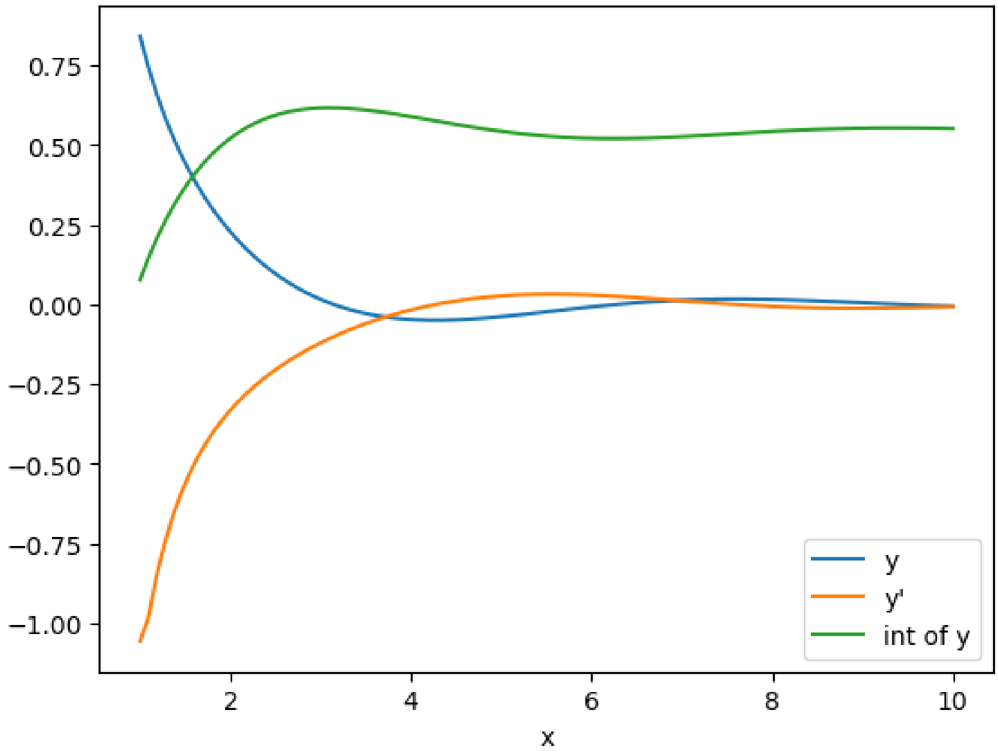
In [33]:
```
np.cumsum([1,2,3,4,5])
```

Out[33]:
```
array([ 1,  3,  6, 10, 15], dtype=int32)
```

In [34]:
```
x= np.linspace(1,10,100)
y= (1/x**2)* np.sin(x)
dydx= np.gradient(y,x)
y_int= np.cumsum(y)* (x[1]-x[0])
```

In [35]:
```
plt.plot(x,y, label='y')
plt.plot(x,dydx, label='y\'')
plt.plot(x,y_int, label='int of y')
plt.legend()
plt.xlabel('x')
```

Out[35]:
```
Text(0.5, 0, 'x')
```

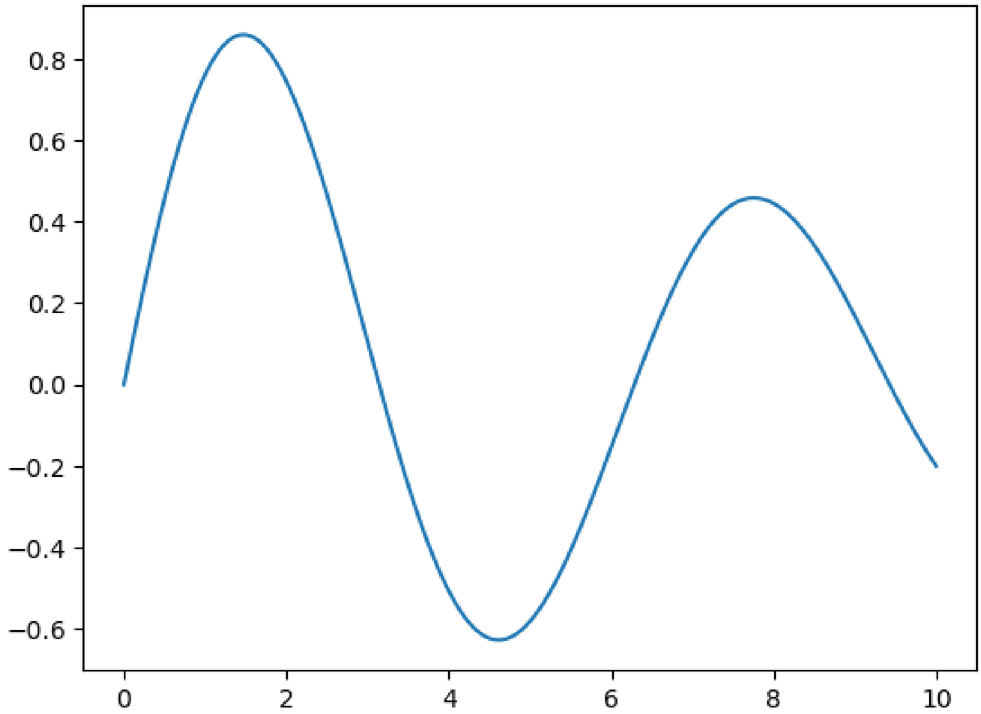


Examples

Question 1

In [36]:
```
#1
n= 10000
x= np.linspace(0,10,n+1)
y= np.exp(-x/10) * np.sin(x)
plt.plot(x,y)
```

Out[36]: `[<matplotlib.lines.Line2D at 0x1e2b8498910>]`



In [37]:
```
#2
mean= np.mean(y[(x>=4) * (x<=7)])
std= np.std(y[(x>=4) * (x<=7)])
print(mean, std)
```

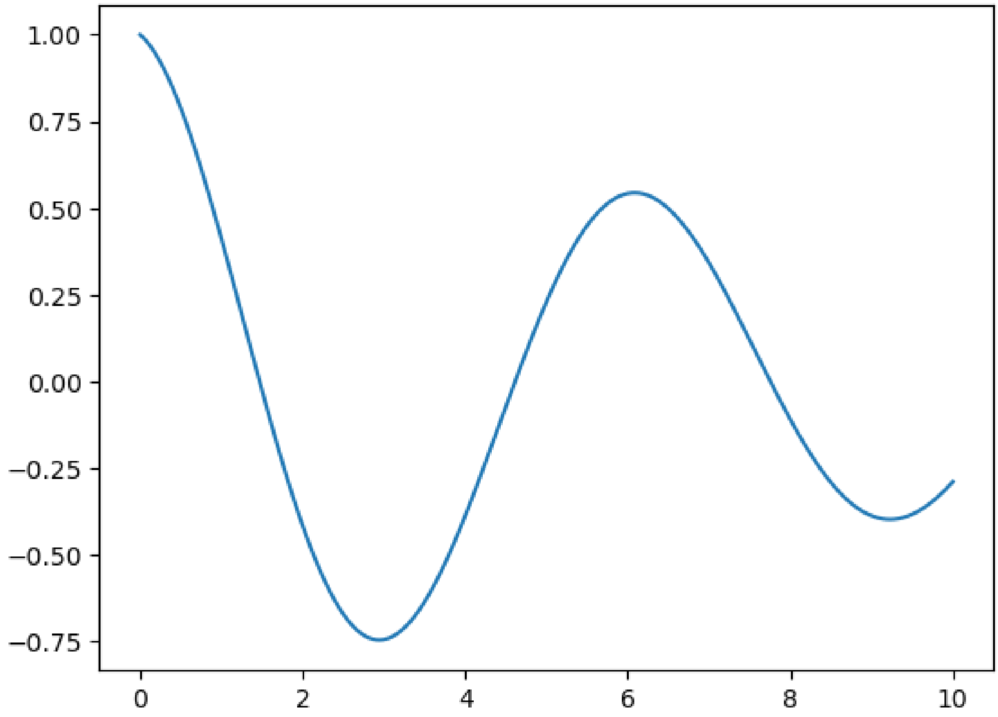
```
-0.2956023068963138 0.31448753875085117
```

In [38]:
```
#3
np.percentile(y[(x>=4) * (x<=7)], 80)
```

Out[38]: `0.06145551274590662`

In [39]:
```
#4
dydx= np.gradient(y,x)
plt.plot(x, dydx)
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x1e2b858a1c0>]
```



In [40]:
```
#5
dydx[1:]
```

Out[40]:
```
array([ 0.99979935,  0.9995979 ,  0.99939548, ..., -0.28918296,
       -0.28892346, -0.28879364])
```

In [41]:
```
dydx[:-1]
```

Out[41]:
```
array([ 0.99989984,  0.99979935,  0.9995979 , ..., -0.28944223,
       -0.28918296, -0.28892346])
```

In [42]:
```
dydx[1:] * dydx[:-1] < 0
```

Out[42]:
```
array([False, False, False, ..., False, False, False])
```

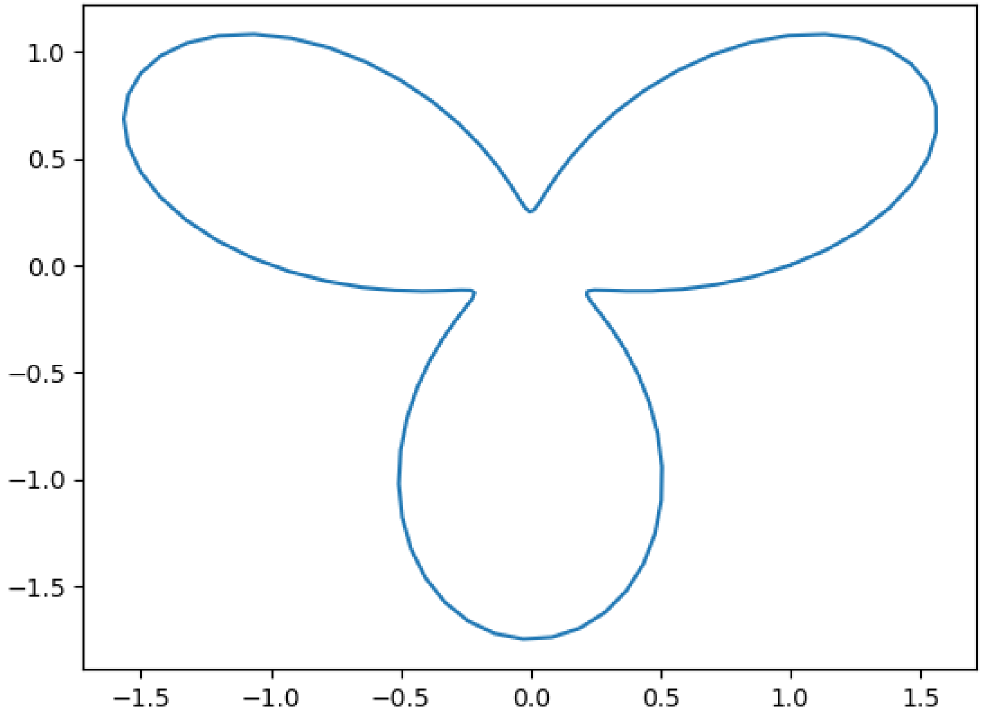
In [43]:
```
x[1:][dydx[1:] * dydx[:-1] < 0]
```

Out[43]:
```
array([1.472, 4.613, 7.755])
```

Question 3

In [44]:
```
# 1
th = np.linspace(0, 2*np.pi, 100)
r = 1 + 3/4 * np.sin(3*th)
x = r* np.cos(th)
y = r* np.sin(th)
plt.plot(x,y)
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x1e2b85f36d0>]
```



In [45]:
```
#2
A = 1/2 * sum(r**2) * (th[1]-th[0])
A
```

Out[45]:
```
4.0568988465390925
```

In [46]:
```
# 3
L = sum(np.sqrt(r**2 + np.gradient(r, th)**2)) * (th[1]-th[0])
L
```

Out[46]:
```
11.734540753337589
```

Question 4

$$P = A\sigma\epsilon T^4$$

or,

$$P/(A\sigma\epsilon) = T^4$$

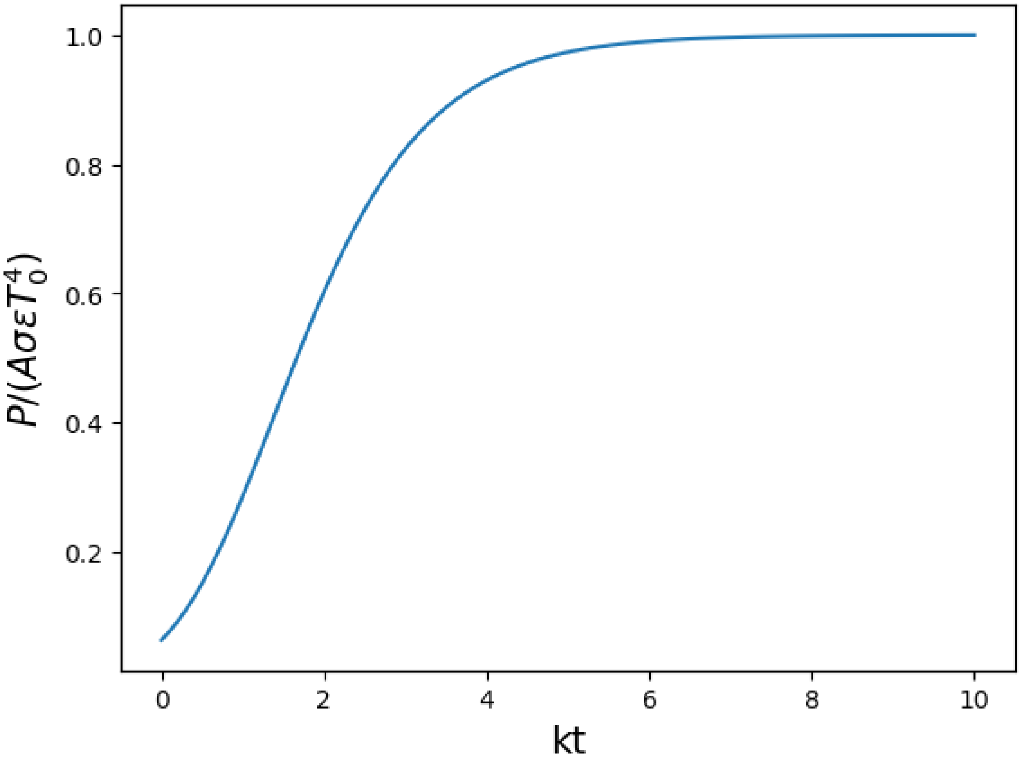
or,

$$P_1 = P/(A\sigma\epsilon T_0^4) = \left(\frac{1}{1+e^{-kt}}\right)^4$$

In [47]:
```
kt= np.linspace(0,10,100)
P1= (1/(1 + np.exp(-kt)))**4
E= np.cumsum(P1) * (kt[1]-kt[0])
```

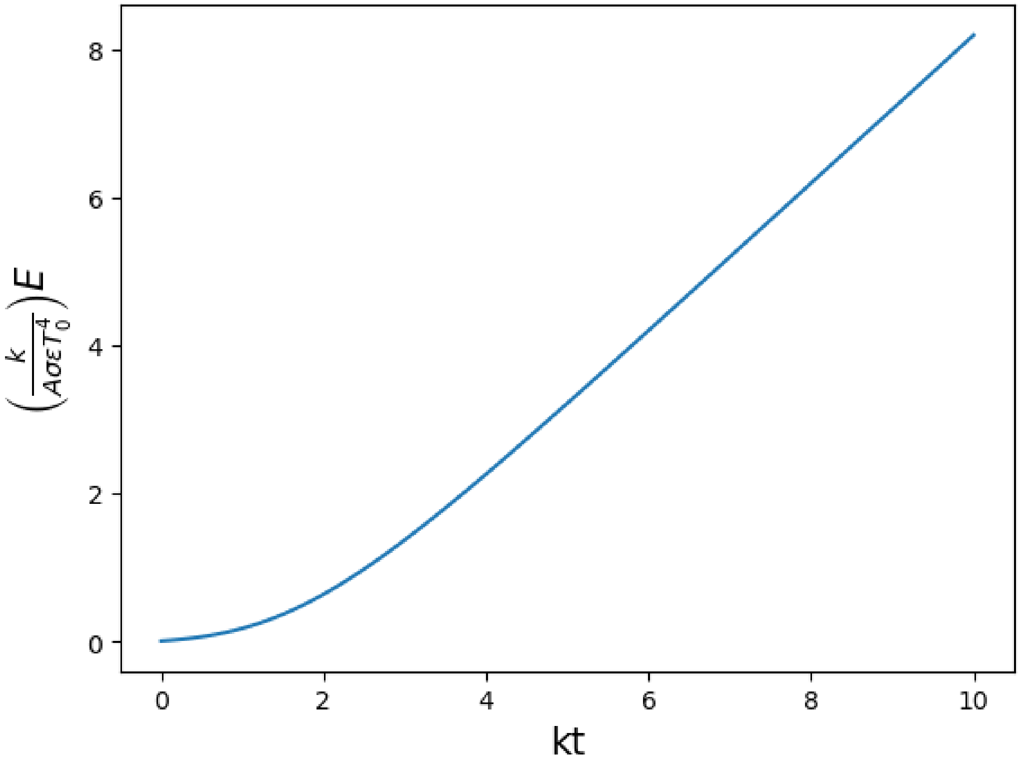
In [48]:
```
plt.plot(kt, P1)
plt.xlabel('kt', fontsize=15)
plt.ylabel('$ P / (A \sigma \epsilon T_0^4) $', fontsize=15)
```

Out[48]:
```
Text(0, 0.5, '$ P / (A \\sigma \\epsilon T_0^4) $')
```



In [49]:
```
plt.plot(kt, E)
plt.xlabel('kt', fontsize=15)
plt.ylabel(r'$ \left(\frac{k}{A \sigma \epsilon T_0^4} \right) E $', fontsize=15)
```

Out[49]:
```
Text(0, 0.5, '$ \\left(\\frac{k}{A \\sigma \\epsilon T_0^4} \\right) E $')
```



Multidimensional Arrays

In [50]:
```
arr1= np.array([[5,3,6],[10,2,3],[7,8,3]])
```

In [51]:
```
arr1*8/9+15
```

Out[51]:
```
array([[19.44444444, 17.66666667, 20.33333333],
       [23.88888889, 16.77777778, 17.66666667],
       [21.22222222, 22.11111111, 17.66666667]])
```

In [52]:
```
arr1.ravel()
```

Out[52]:
```
array([ 5,  3,  6, 10,  2,  3,  7,  8,  3])
```

In [53]:
```
arr2= np.random.randn(3,2)
arr2
```

Out[53]:
```
array([[ 0.03809453,  0.93246578],
       [-0.74852545,  0.0273295 ],
       [ 0.97853489, -0.91420894]])
```

In [54]:
```
arr1<20
```

Out[54]:
```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

In [55]:
```
arr1[arr1<20]
```

Out[55]:
```
array([ 5,  3,  6, 10,  2,  3,  7,  8,  3])
```

In [56]:
```
arr1
```

Out[56]:
```
array([[ 5,  3,  6],
       [10,  2,  3],
       [ 7,  8,  3]])
```

In [57]:
```
arr1[1], arr1[2][0], arr1[1,2], arr1[1][2]
```

Out[57]:
```
(array([10,  2,  3]), 7, 3, 3)
```

In [58]:
```
arr1[1:][0], arr1[1:,0]
```

Out[58]:
```
(array([10,  2,  3]), array([10,  7]))
```

In [59]:
```
arr1[1:][1][0], arr1[1:,1][0]
```

Out[59]:
```
(7, 2)
```

In [60]:
```
arr1[0,:2], arr1[1:,:2]
```

Out[60]:
```
(array([5, 3]),
 array([[10,  2],
        [ 7,  8]]))
```

See more in the original code linked at the top

dealing with 2D functions

In [61]:
```
x= np.linspace(0,10,50)
y= np.linspace(0,5,25)
```

need to use meshgrid here

In [62]:
```
xv, yv= np.meshgrid(x,y)
```

In [63]:
```
xv
```

Out[63]:
```
array([[ 0.        ,  0.20408163,  0.40816327, ...,  9.59183673,
         9.79591837, 10.        ],
       [ 0.        ,  0.20408163,  0.40816327, ...,  9.59183673,
         9.79591837, 10.        ],
       [ 0.        ,  0.20408163,  0.40816327, ...,  9.59183673,
         9.79591837, 10.        ],
       ...,
       [ 0.        ,  0.20408163,  0.40816327, ...,  9.59183673,
         9.79591837, 10.        ],
       [ 0.        ,  0.20408163,  0.40816327, ...,  9.59183673,
         9.79591837, 10.        ],
       [ 0.        ,  0.20408163,  0.40816327, ...,  9.59183673,
         9.79591837, 10.        ]])
```

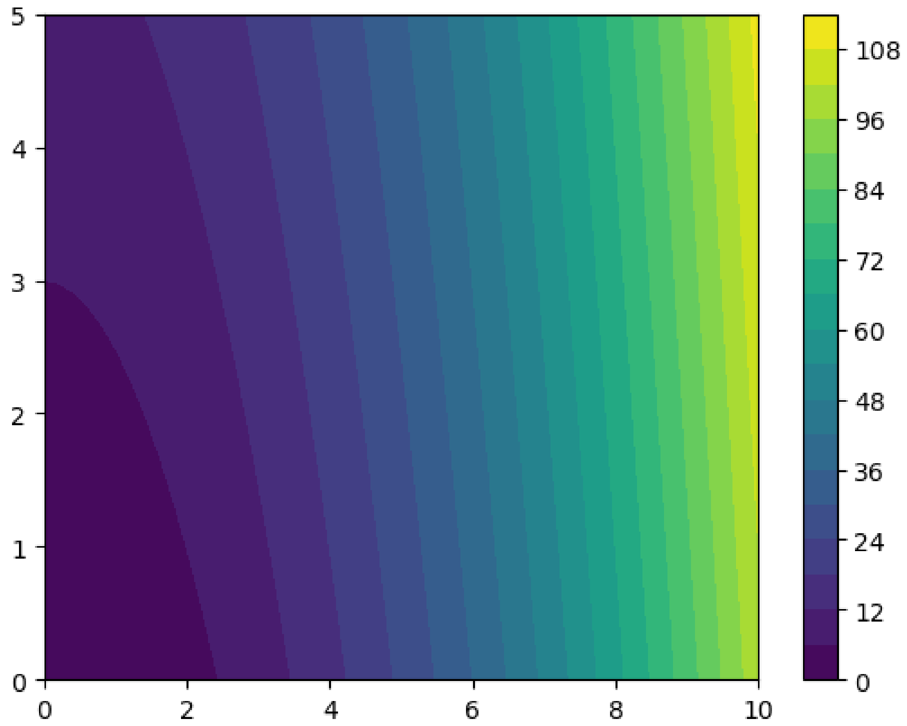
In [64]:
```
yv
```

Out[64]:
```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.20833333, 0.20833333, 0.20833333, ..., 0.20833333, 0.20833333,
        0.20833333],
       [0.41666667, 0.41666667, 0.41666667, ..., 0.41666667, 0.41666667,
        0.41666667],
       ...,
       [4.58333333, 4.58333333, 4.58333333, ..., 4.58333333, 4.58333333,
        4.58333333],
       [4.79166667, 4.79166667, 4.79166667, ..., 4.79166667, 4.79166667,
        4.79166667],
       [5.        , 5.        , 5.        , ..., 5.        , 5.        ,
        5.        ]])
```

In [65]:
```
zv= xv**2 + yv*2
```

In [66]:
```
plt.contourf(xv,yv,zv, levels= 20)
plt.colorbar()
```

Out[66]:
```
<matplotlib.colorbar.Colorbar at 0x1e2b9b34b20>
```



Basic Linear Algebra

Matrix operations

In [67]:
```
A= np.array([[6,7,2],[0.5,9,0],[1,4,-3]]) # matrix
b1= np.array([8,3,-4])  # vector
b2= np.array([4,-9,0])  # vector
```

In [68]:
```
A @ b1
```

Out[68]:
```
array([61., 31., 32.])
```

In [69]:
```
A.T
```
Out[69]:
```
array([[ 6. ,  0.5,  1. ],
       [ 7. ,  9. ,  4. ],
       [ 2. ,  0. , -3. ]])
```
In [70]:
```
np.dot(b1,b2)
```
Out[70]:
```
5
```
In [71]:
```
np.cross(b1,b2), np.cross(b2,b1)
```
Out[71]:
```
(array([-36, -16, -84]), array([36, 16, 84]))
```
systems of equations

3x + 2y + z = 4

5x -5y + 4z = 3

6x +z = 0

In [72]:
```
A= np.array([[3,2,1],[5,-5,4],[6,0,1]])
B= np.array([4,3,0])
```
In [73]:
```
X= np.linalg.solve(A,B)
X
```
Out[73]:
```
array([-0.49056604,  1.26415094,  2.94339623])
```
finding Eigenvalues

w is for the eigenvalues and v_1,v_2 and v_3 represent the corresponding eigenvectors

In [74]:
```
A
```
Out[74]:
```
array([[ 3,  2,  1],
       [ 5, -5,  4],
       [ 6,  0,  1]])
```
In [75]:
```
w, v = np.linalg.eig(A)
```
In [76]:
```
w
```
Out[76]:
```
array([ 5.98847677, -1.66137965, -5.32709712])
```
In [77]:
```
v
```
Out[77]:
```
array([[-0.55522613, -0.36439757,  0.25391074],
       [-0.49573499,  0.43853605, -0.93677764],
       [-0.66781043,  0.82152331, -0.24078411]])
```
In [ ]:
```

```

In [78]:
```
v1= v[:,0]  # taking 1st column
v1
```

Out[78]:
```
array([-0.55522613, -0.49573499, -0.66781043])
```

In [79]:
```
v2= v[:,1]
v2
```

Out[79]:
```
array([-0.36439757,  0.43853605,  0.82152331])
```

In [80]:
```
v3= v[:,2]
v3
```

Out[80]:
```
array([ 0.25391074, -0.93677764, -0.24078411])
```

In [81]:
```
# verification for the 1st eigenvalue
A @ v1, w[0]*v1
```

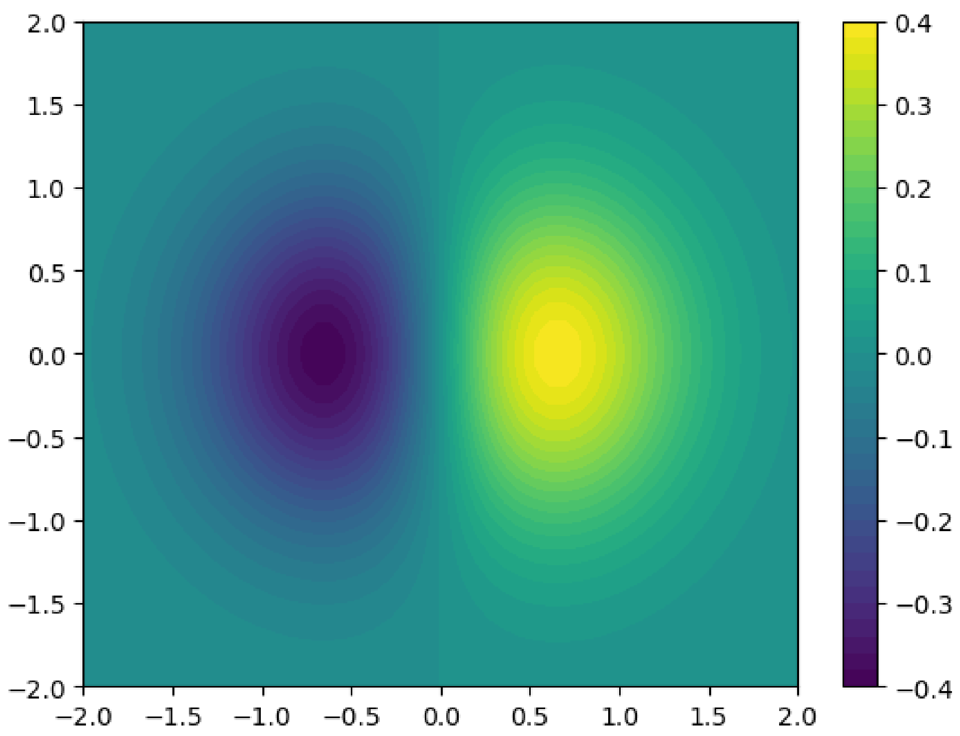
Out[81]:
```
(array([-3.3249588 , -2.96869744, -3.99916722]),
 array([-3.3249588 , -2.96869744, -3.99916722]))
```

examples

question 1

In [82]:
```
#1
x= np.linspace(-2,2,50)
y= np.linspace(-2,2,50)
xv,yv = np.meshgrid(x,y)
f= np.exp(-(xv**2 + yv**2)) * np.sin(xv)
plt.contourf(xv,yv,f, levels=50)
plt.colorbar()
```

Out[82]:
```
<matplotlib.colorbar.Colorbar at 0x1e2b9c3f070>
```



In [83]:
```
#2
np.abs(f.ravel()).sum() * np.diff(x)[0] * np.diff(y)[0]
```

Out[83]: 1.4904322675869597

In [84]:
```
#3
f1= f[(xv**2 + yv**2)**0.5 > 0.5]
np.abs(f1.ravel()).sum() * np.diff(x)[0] * np.diff(y)[0]
```

Out[84]: 1.3448352900350378

question 2

After examining a circuit full of resistors, you find that the voltage at 4 specified points is given by

$$3V_1 + 2V_2 + 3V_3 + 10V_4 = 4$$
$$2V_1 - 2V_2 + 5V_3 + 8V_4 = 1$$
$$3V_1 + 3V_2 + 4V_3 + 9V_4 = 3$$

$$3V_1 + 4V_2 - 3V_3 - 7V_4 = 2$$

Find all the voltages.

In [85]:
```
A= np.array([[3,2,3,10],[2,-2,5,8],[3,3,4,9],[3,4,-3,-7]])
B= np.array([4,1,3,2])
np.linalg.solve(A,B)
```

Out[85]: array([0.78378378, 0.03603604, -0.67567568, 0.36036036])

question 3

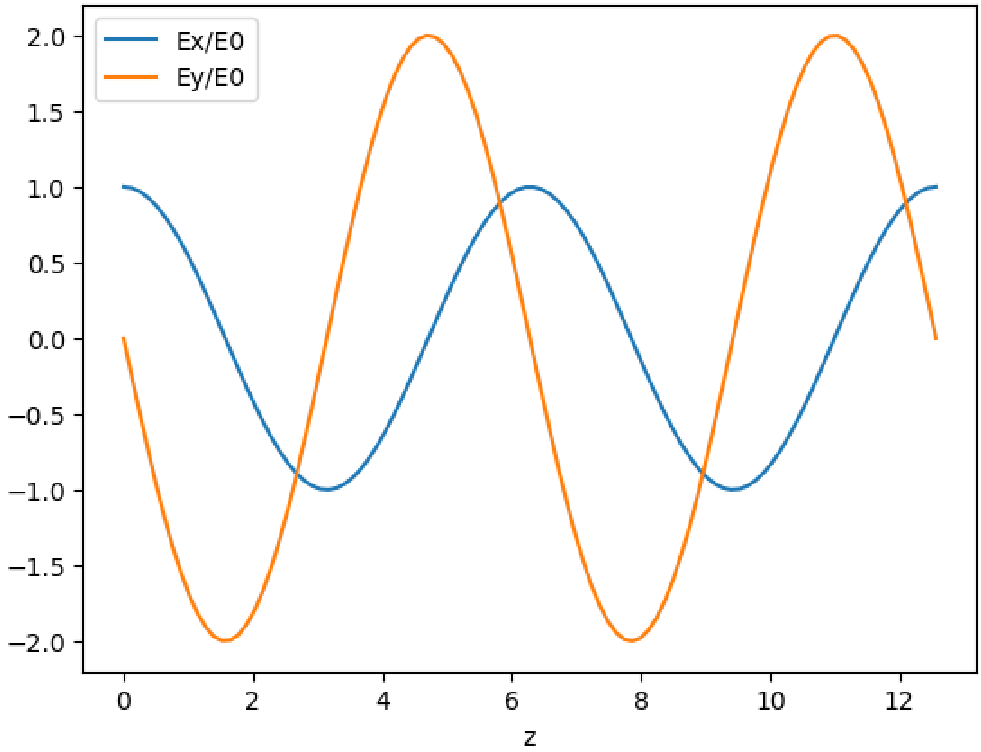
In [86]:
```
#1
z= np.linspace(0, 4* np.pi, 100)
t= np.linspace(0,10,100)
zv,tv = np.meshgrid(z,t)

Ex = np.cos(zv-tv)        # it's Ex/E0 actually
Ey = 2*np.cos(zv-tv+np.pi/2)
Ez = 0
```

Ex for z=z and t=0

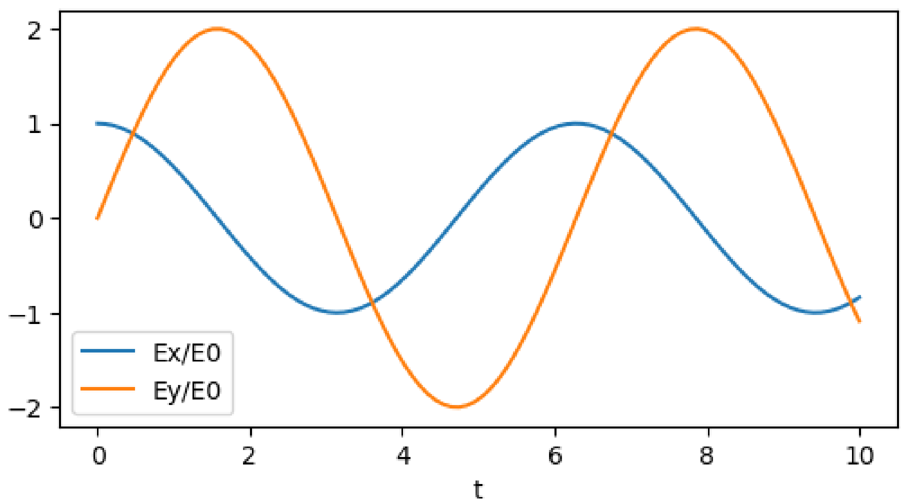
In [87]:
```
plt.plot(z, Ex[0], label='Ex/E0')
plt.plot(z, Ey[0], label='Ey/E0')
plt.xlabel('z')
plt.legend()
```

Out[87]: <matplotlib.legend.Legend at 0x1e2b9cad6a0>



Ex for z=0 and t=t

In [88]:
```
plt.figure(figsize=(6,3))
plt.plot(t, Ex[:,0], label='Ex/E0')
plt.plot(t, Ey[:,0], label='Ey/E0')
plt.xlabel('t')
plt.legend()
plt.show()
```



try the rest of the part later

question 4

In [ ]:

In [ ]: