Course Code	**:**	18ECE201J	**Course Title**	**:**	Python and Scientific Python
Reg. No.	**:**	RA1811004010179	**Name**	**:**	SAMARTH KATHURIA
Semester	**:**	V Semester	**Year**	**:**	III Year
Date of Expt.	**:**	01/12/2020	**Date of Submission**	**:**	02/12/2020
Name of the Lab Instructor :			M Vasanthi Ma'am		

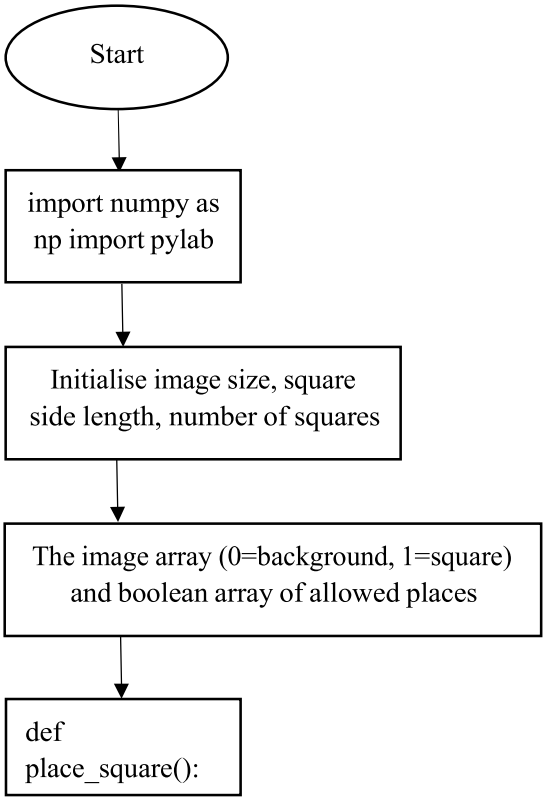
TITLE OF THE EXPERIMENT:

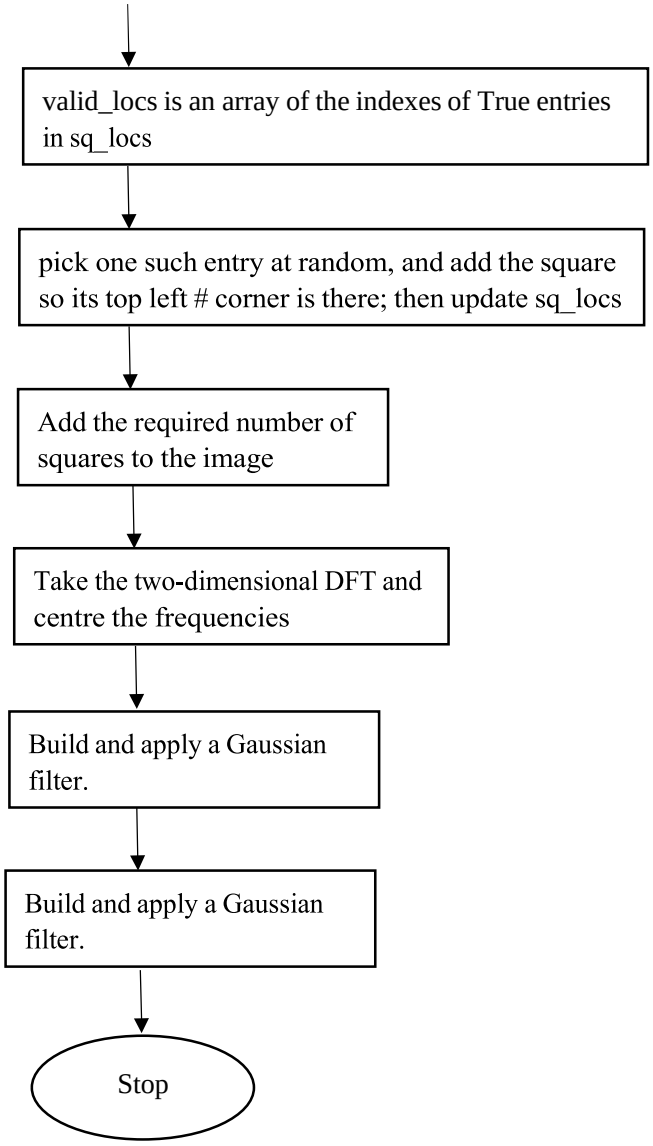
15. Numpy signal processing-Blurring an image with a Gaussian filter

AIM:

To write a python code to execute numpy signal processing-to blur an image with Gaussian filter.

FLOWCHART:




valid_locs is an array of the indexes of True entries in sq_locs
pick one such entry at random, and add the square so its top left # corner is there; then update sq_locs
Add the required number of squares to the image
Take the two-dimensional DFT and centre the frequencies
Build and apply a Gaussian filter.
Build and apply a Gaussian filter.
Stop

```python
#MINI PROJECT
# Class: III Year ECE - C
# Course: 18ECE201J - Python and Scientific Python
# Student Name and Reg Number: SAMARTH KATHURIA(RA1811004010179)
import numpy as np
import pylab
# image size, square side length, number of squares
ncols, nrows = 120, 120
sq_size, nsq = 10, 20
# The image array (0=background, 1=square) and boolean array of allowed places
# to add a square so that it doesn't touch another or the image sides
image = np.zeros((nrows, ncols))
sq_locs = np.zeros((nrows, ncols), dtype=bool)
sq_locs[1:-sq_size-1:,1:-sq_size-1] = True
def place_square():
#Place a square at random on the image and update sq_locs.
# valid_locs is an array of the indexes of True entries in sq_locs
    valid_locs = np.transpose(np.nonzero(sq_locs))
# pick one such entry at random, and add the square so its top left
# corner is there; then update sq_locs
    i, j = valid_locs[np.random.randint(len(valid_locs))]
    image[i:i+sq_size, j:j+sq_size] = 1
    imin, jmin = max(0,i-sq_size-1), max(0, j-sq_size-1)
    sq_locs[imin:i+sq_size+1, jmin:j+sq_size+1] = False
# Add the required number of squares to the image
for i in range(nsq):
    place_square()
    pylab.imshow(image)
    pylab.show()
# Take the two-dimensional DFT and center the frequencies
ftimage = np.fft.fft2(image)
ftimage = np.fft.fftshift(ftimage)
pylab.imshow(np.abs(ftimage))
pylab.show()
# Build and apply a Gaussian filter.
```

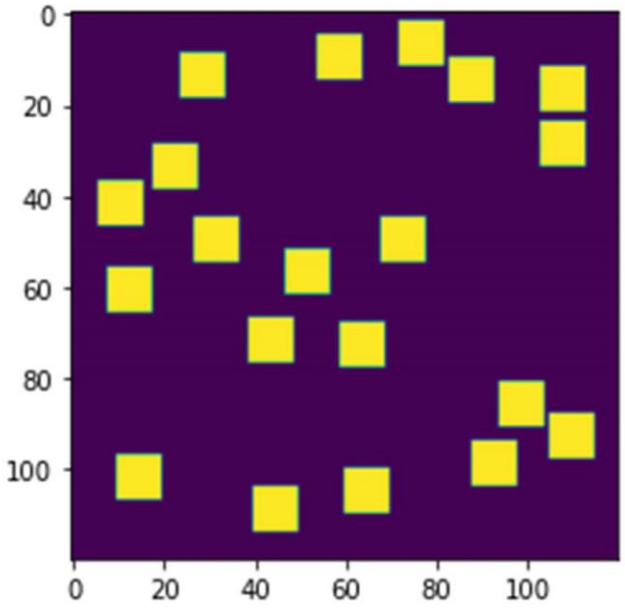
```python
# Build and apply a Gaussian filter.
sigmax, sigmay = 10, 10
cy, cx = nrows/2, ncols/2
x = np.linspace(0, nrows, nrows)
y = np.linspace(0, ncols, ncols)
X, Y = np.meshgrid(x, y)
gmask = np.exp(-(((X-cx)/sigmax)**2 + ((Y-cy)/sigmay)**2))
ftimagep = ftimage * gmask
pylab.imshow(np.abs(ftimagep))
pylab.show()
# Finally, take the inverse transform and show the blurred image
imagep = np.fft.ifft2(ftimagep)
pylab.imshow(np.abs(imagep))
pylab.show()
```

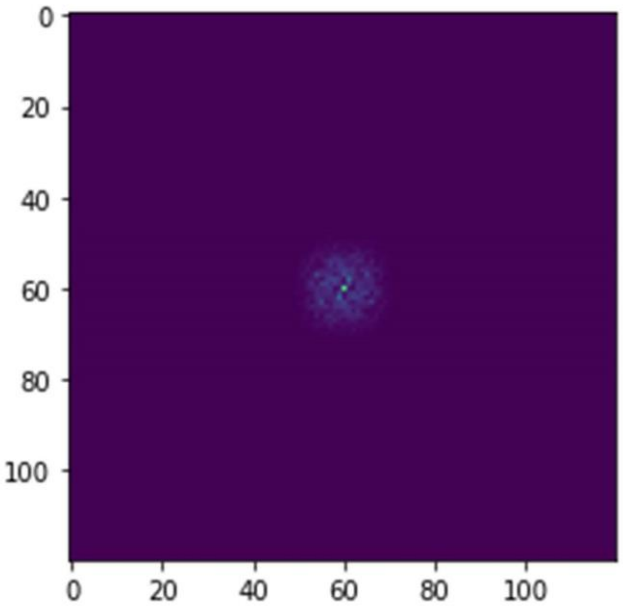
PROGRAM:

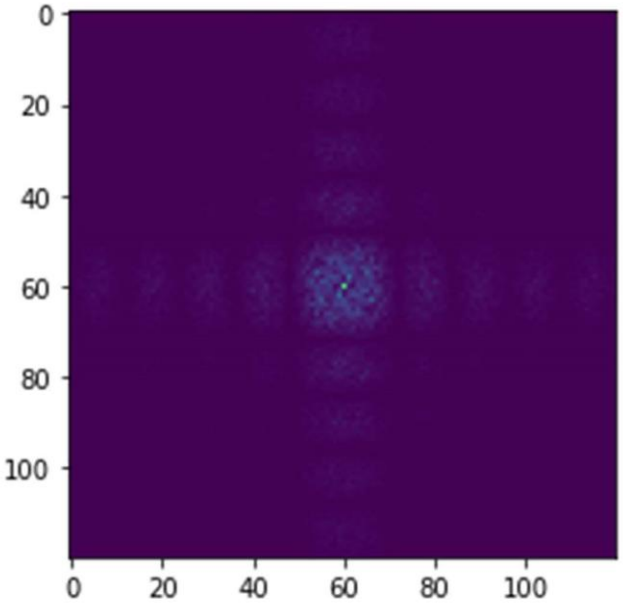
```
#MINI PROJECT
# Class: III Year ECE - C
# Course: 18ECE201J – Python and Scientific Python
# Student Name and Reg Number: SAMARTH
KATHURIA(RA1811004010179)
import numpy as np import pylab
# image size, square side length, number of squares
ncols, nrows = 120, 120 sq_size, nsq = 10, 20
# The image array (0=background, 1=square) and boolean array of allowed places
# to add a square so that it doesn’t touch another or the image sides
image = np.zeros((nrows, ncols)) sq_locs = np.zeros((nrows, ncols),
dtype=bool) sq_locs[1:-sq_size-1:,1:-sq_size-1] = True def
place_square():
    """ Place a square at random on the image and update sq_locs. """ #
valid_locs is an array of the indexes of True entries in sq_locs     valid_locs
= np.transpose(np.nonzero(sq_locs))
# pick one such entry at random, and add the square so its top left
# corner is there; then update sq_locs     i, j =
valid_locs[np.random.randint(len(valid_locs))]
image[i:i+sq_size, j:j+sq_size] = 1     imin, jmin = max(0,i-
sq_size-1), max(0, j-sq_size-1)     sq_locs[imin:i+sq_size+1,
jmin:j+sq_size+1] = False # Add the required number of
squares to the image for i in range(nsq):
    place_square()
pylab.imshow(image) pylab.show()
```

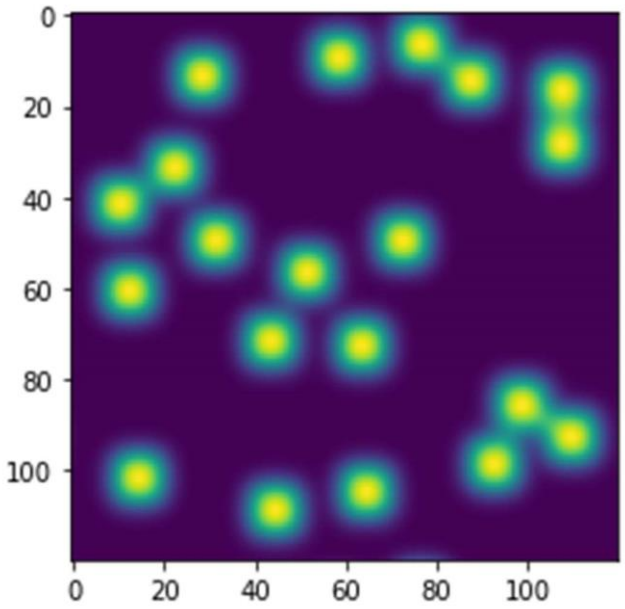
```
# Take the two-dimensional DFT and center the frequencies
ftimage = np.fft.fft2(image) ftimage = np.fft.fftshift(ftimage)
pylab.imshow(np.abs(ftimage)) pylab.show()
# Build and apply a Gaussian filter.
sigmax, sigmay = 10, 10 cy, cx = nrows/2, ncols/2 x =
np.linspace(0, nrows, nrows) y = np.linspace(0, ncols, ncols) X, Y
= np.meshgrid(x, y) gmask = np.exp(-(((X-cx)/sigmax)**2 + ((Y-
cy)/sigmay)**2)) ftimagep = ftimage * gmask
pylab.imshow(np.abs(ftimagep)) pylab.show()
# Finally, take the inverse transform and show the blurred image
imagep = np.fft.ifft2(ftimagep) pylab.imshow(np.abs(imagep))
pylab.show()
```

OUTPUT:




0
20
40
60
80
100
0
20
40
60
80
100


0
20
40
60
80
100
0
20
40
60
80
100



PRELAB QUESTIONS:

1) What type of Gaussian filter is used in above program?

2) What is the significance of np.fft.fft

3) Explain place_square() function in above program

POST LAB QUESTIONS:

1) Consider a signal in the time domain defined by the function $f(t) = \cos(2\pi\nu t)e^{-t/\tau}$, with frequency $\nu = 250$ Hz decaying exponentially with a lifetime $\tau = 0.2$ s. Plot the function, sampled at 1,000 Hz, and its discrete Fourier transform against frequency

PRELAB ANSWERS:

EXP-15

PRELAB QUESTIONS

1. What type of Gaussian filter is used in above program?

Ans The gaussian filter used is a non-uniform Low pass filter.

2. What is the significance of np.fft.fft.

Ans The function computes the 1-D n-point DFT with the efficient FFT algorithm.

3. Explain Plot, square() function in above program.

Ans. The function is used to fill up square at location on the image and to check if its a valid location.

POSTLAB ANSWERS:

jupyter EXPERIMENT 15 Last Checkpoint: 20 minutes ago (unsaved changes)

Logout

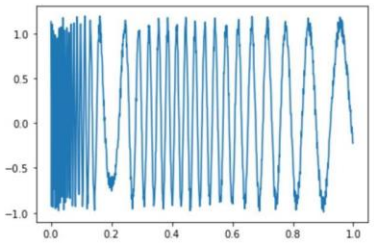
File Edit View Insert Cell Kernel Widgets Help

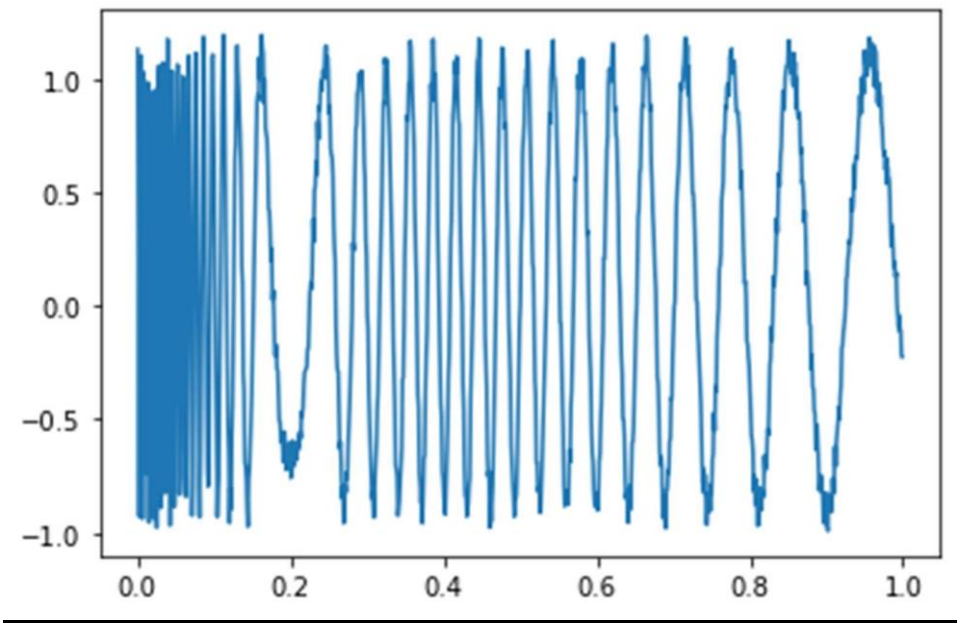
Trusted | Python 3

Run Code

In [9]:
```
import matplotlib.pyplot as plt
import numpy as np
frq=250
decay=0.2
fsamp=1000
T=1/fsamp
t=np.arange(0,1,T)
n=len(t)
f= np.cos((2*np.pi*frq*t)*np.exp(-(t/decay)))
f+=0.2* np.random.rand(n)
plt.plot(t,f)
```

Out[9]: [<matplotlib.lines.Line2D at 0x24232394e20>]





RESULT:-

Thus Numpy signal processing-Blurring an image with a Gaussian filter is executed using python.