Lab 16

Name :- Aryan Dilipbhai Langhanoja

Date :- 25-09-2023

Enrollment No :- 92200133030

CO1: To write, test, and debug simple Python programs

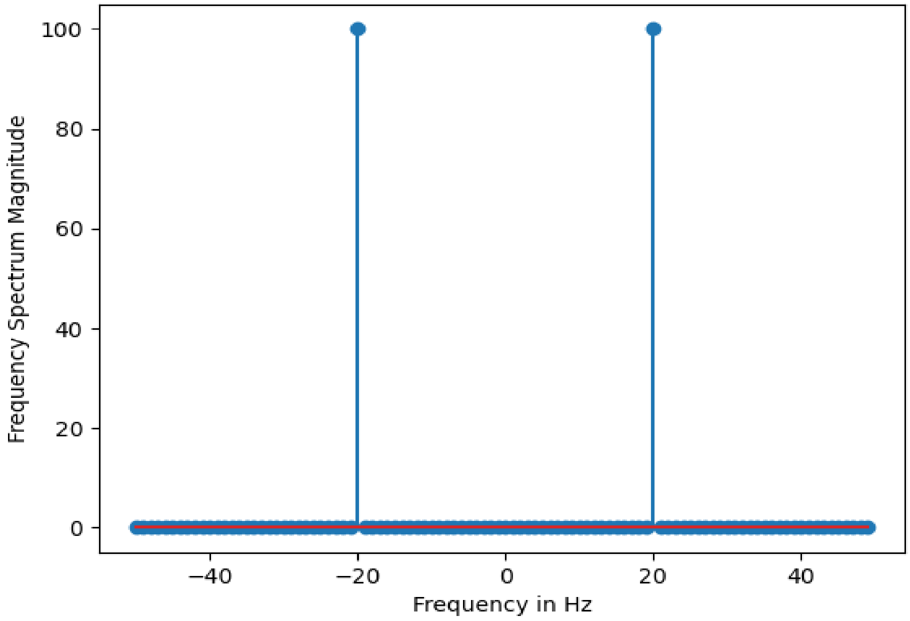
CO2: To implement Python programs with conditional, loops and functions

Task 1:- Frequency Spectrum Analysis of Sample Data using FFT

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import fftpack
fre_samp = 100  # Sample frequency in Hz
t = np.arange(0, 1, 1/fre_samp)  # Time vector
A = 5 * np.sin(2 * np.pi * 50 * t) + 2 * np.sin(2 * np.pi * 120 * t)
A_fft = fftpack.fft(A)
frequency = fftpack.fftfreq(len(A), 1 / fre_samp)
plt.stem(frequency, np.abs(A_fft), use_line_collection=True)
plt.xlabel('Frequency in Hz')
plt.ylabel('Frequency Spectrum Magnitude')
plt.show()
```

Output:

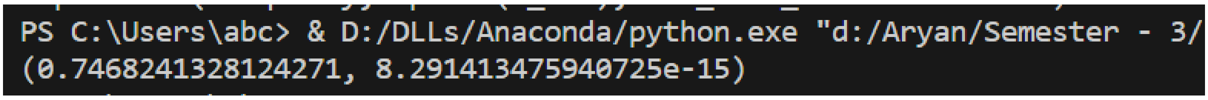


Task 2:- Numerical Integration of a Gaussian Function Using SciPy

Python Code:

```
from numpy import exp
import scipy.integrate
def f(x): return exp(-x**2)
i = scipy.integrate.quad(f, 0, 1)
print(i)
```

Output:



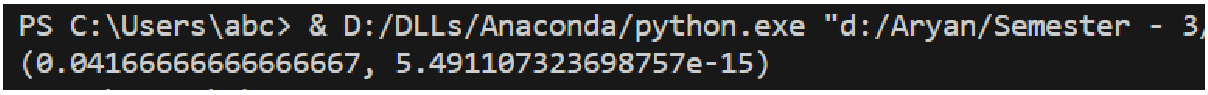
```
PS C:\Users\abc> & D:/DLLs/Anaconda/python.exe "d:/Aryan/Semester - 3/
(0.7468241328124271, 8.291413475940725e-15)
```

Task 3:- Double Integration of a Function over a Rectangular Region

Python Code:

```
import scipy.integrate
from numpy import exp
from math import sqrt
def f(x, y): return 2*x*y
def g(x): return 0
def h(y): return 4*y**2
i = scipy.integrate.dblquad(f, 0, 0.5, g, h)
print(i)
```

Output:



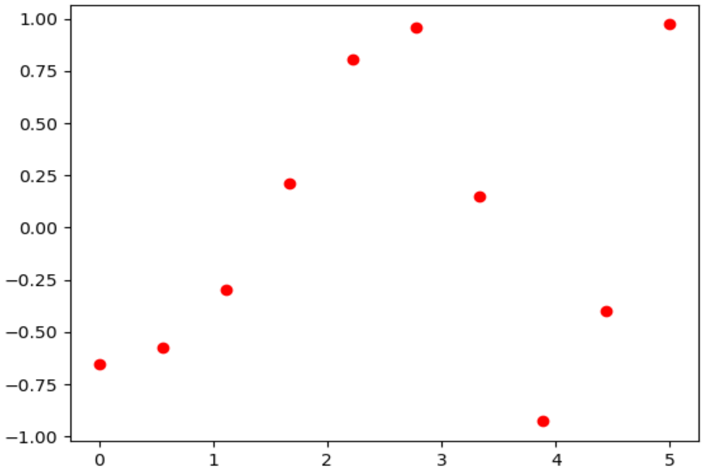
```
PS C:\Users\abc> & D:/DLLs/Anaconda/python.exe "d:/Aryan/Semester - 3,
(0.04166666666666667, 5.491107323698757e-15)
```

Task 4:- Double Integration of a Function over a Rectangular Region

Python Code:

```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
x = np.linspace(0, 5, 10)
y = np.cos(x**2/3+4)
plt.scatter(x,y,c='r')
plt.show()
```

Output:

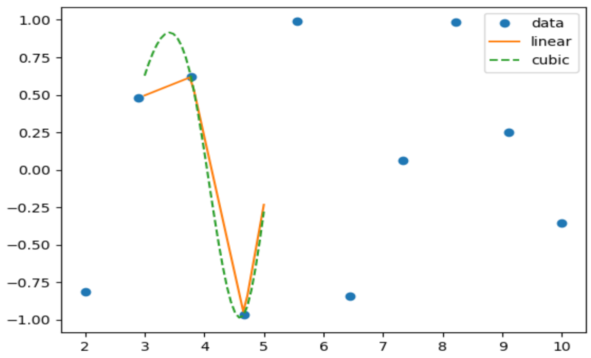


Task 5:- Interpolation Comparison with Scipy and Matplotlib

Python Code:

```
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(2, 10, 10)
y = np.sin(x**2/3+4)
fun1 = interp1d(x, y, kind='linear')
fun2 = interp1d(x, y, kind='cubic')
xnew = np.linspace(3, 5, 30)
plt.plot(x, y, 'o', xnew, fun1(xnew), '-', xnew, fun2(xnew), '--')
plt.legend(['data', 'linear', 'cubic', 'nearest'], loc='best')
plt.show().
```

Output:

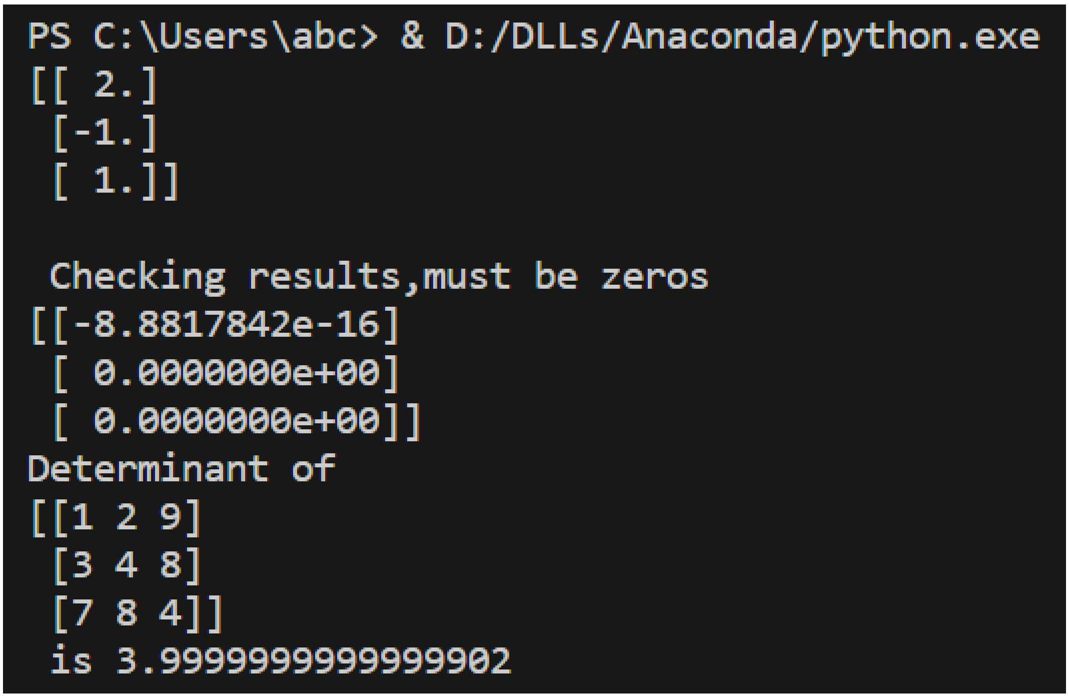


Task 6:- Linear System Solution and Determinant Calculation using SciPy and NumPy

Python Code:

```
import numpy as np
from scipy import linalg
import numpy as np
from scipy import linalg
a = np.array([[1, 2, -3], [2, -5, 4], [5, 4, -1]])
b = np.array([[-3], [13], [5]])
x = linalg.solve(a, b)
print(x)
print("\n Checking results,must be zeros")
print(a.dot(x) - b)
A = np.array([[1, 2, 9], [3, 4, 8], [7, 8, 4]])
x = linalg.det(A)
print('Determinant of \n{} \n is {}'.format(A, x))
```

Output:



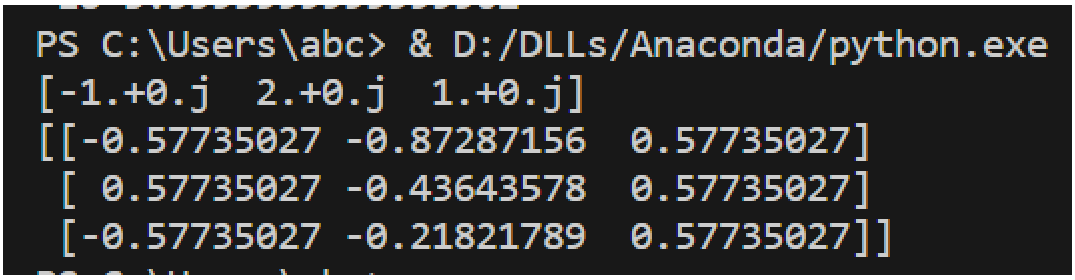
```
PS C:\Users\abc> & D:/DLLs/Anaconda/python.exe
[[ 2.]
 [-1.]
 [ 1.]]

 Checking results,must be zeros
[[-8.8817842e-16]
 [ 0.0000000e+00]
 [ 0.0000000e+00]]
Determinant of
[[1 2 9]
 [3 4 8]
 [7 8 4]]
 is 3.9999999999999902
```

Task 7:- Calculating Eigenvalues and Eigenvectors of a Matrix using SciPy and NumPy

Python Code:

```
from scipy import linalg
import numpy as np
A = np.array([[2, 1, -2], [1, 0, 0], [0, 1, 0]])
values, vectors = linalg.eig(A)
print(values)
print(vectors)
```

Output:



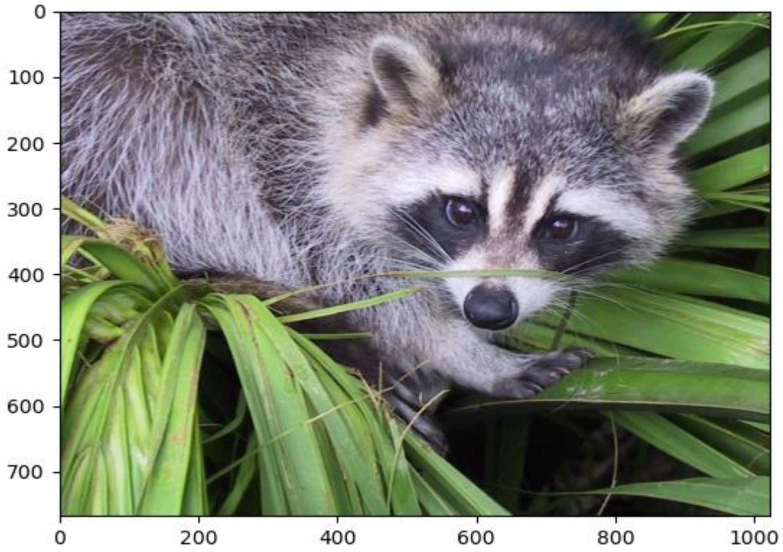
```
PS C:\Users\abc> & D:/DLLs/Anaconda/python.exe
[-1.+0.j  2.+0.j  1.+0.j]
[[-0.57735027 -0.87287156  0.57735027]
 [ 0.57735027 -0.43643578  0.57735027]
 [-0.57735027 -0.21821789  0.57735027]]
```

Task 8:- Displaying a Raccoon Face Image Using Matplotlib

Python Code:

```
import scipy.misc
import matplotlib.pyplot as plt
face = scipy.misc.face()  # returns an image of raccoon
plt.imshow(face)
plt.show()
```

Output:



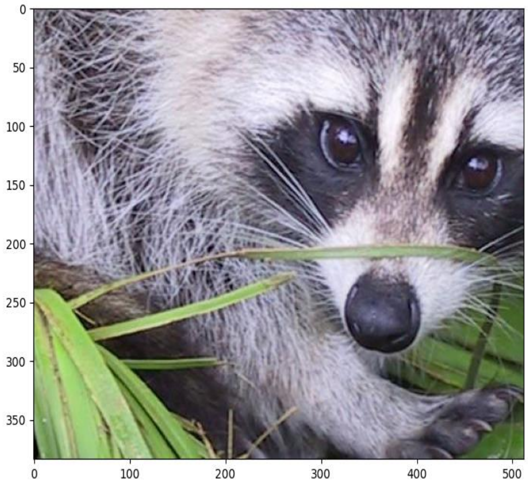
Task 9:- Image Cropping with SciPy and Matplotlib

Python Code:

```
import scipy.misc
import matplotlib.pyplot as plt
face = scipy.misc.face()  # returns an image of raccoon
lx, ly, channels = face.shape
crop_face = face[int(lx/4):int(-lx/4), int(ly/4):int(-ly/4)]
```

```
plt.imshow(crop_face)
plt.show()
```

Output:

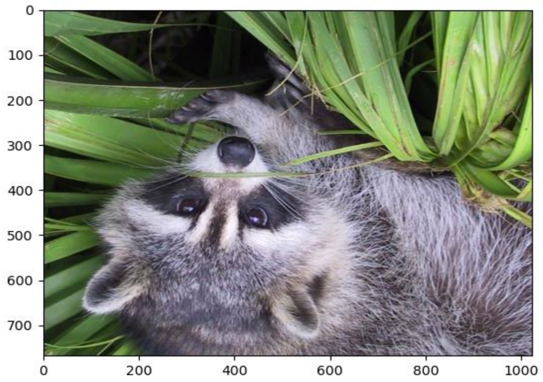


Task 10:- Image Rotation using SciPy and Matplotlib

Python Code:

```
from scipy import misc, ndimage
import matplotlib.pyplot as plt
face = misc.face()
rotate_face = ndimage.rotate(face, 180)
plt.imshow(rotate_face)
plt.show()
```

Output:

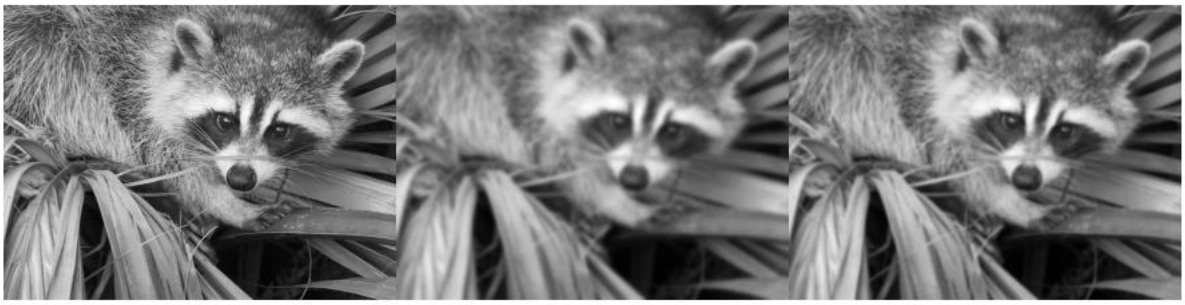


Task 11:- Image Blurring with Gaussian Filters and Visualization

Python Code:

```
import scipy as scipy
from scipy import ndimage, misc
import matplotlib.pyplot as plt
face = scipy.misc.face(gray=True)
blurred_face = ndimage.gaussian_filter(face, sigma=3)
very_blurred = ndimage.gaussian_filter(face, sigma=5)
plt.figure(figsize=(9, 3))
plt.subplot(131)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(132)
plt.imshow(very_blurred, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(133)
plt.imshow(blurred_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)
plt.show()
```

Output:



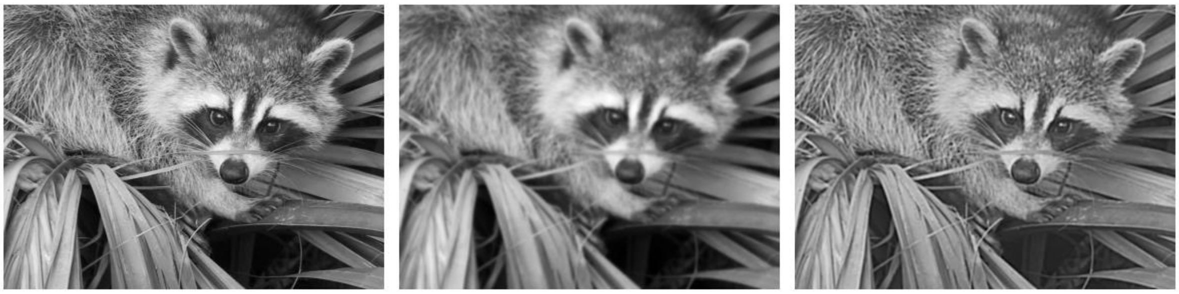
Task 12:- Image Sharpening with Gaussian Filters

Python Code:

```
import scipy
from scipy import ndimage
import matplotlib.pyplot as plt
f = scipy.misc.face(gray=True).astype(float)
blurred_f = ndimage.gaussian_filter(f, 3)
filter_blurred_f = ndimage.gaussian_filter(blurred_f, 1)
alpha = 30
sharpened = blurred_f + alpha * (blurred_f - filter_blurred_f)
```

```
plt.figure(figsize=(12, 4))
plt.subplot(131)
plt.imshow(f, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(132)
plt.imshow(blurred_f, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(133)
plt.imshow(sharpened, cmap=plt.cm.gray)
plt.axis('off')
plt.tight_layout()
plt.show()
```

Output:

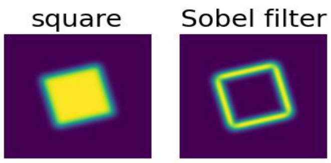


Task 13:- Image Processing: Rotation, Gaussian Smoothing, and Sobel Filter Application on a Square Image

Python Code:

```
import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt
im = np.zeros((256, 256))
im[64:-64, 64:-64] = 1
print(im)
im = ndimage.rotate(im, 15, mode='constant')
im = ndimage.gaussian_filter(im, 8)
sx = ndimage.sobel(im, axis=0, mode='constant')
sy = ndimage.sobel(im, axis=1, mode='constant')
sob = np.hypot(sx, sy)
plt.figure(figsize=(9, 5))
plt.subplot(141)
plt.imshow(im)
plt.axis('off')
plt.title('square', fontsize=20)
plt.subplot(142)
plt.imshow(sob)
plt.axis('off')
plt.title('Sobel filter', fontsize=20)
plt.show()
```

Output:



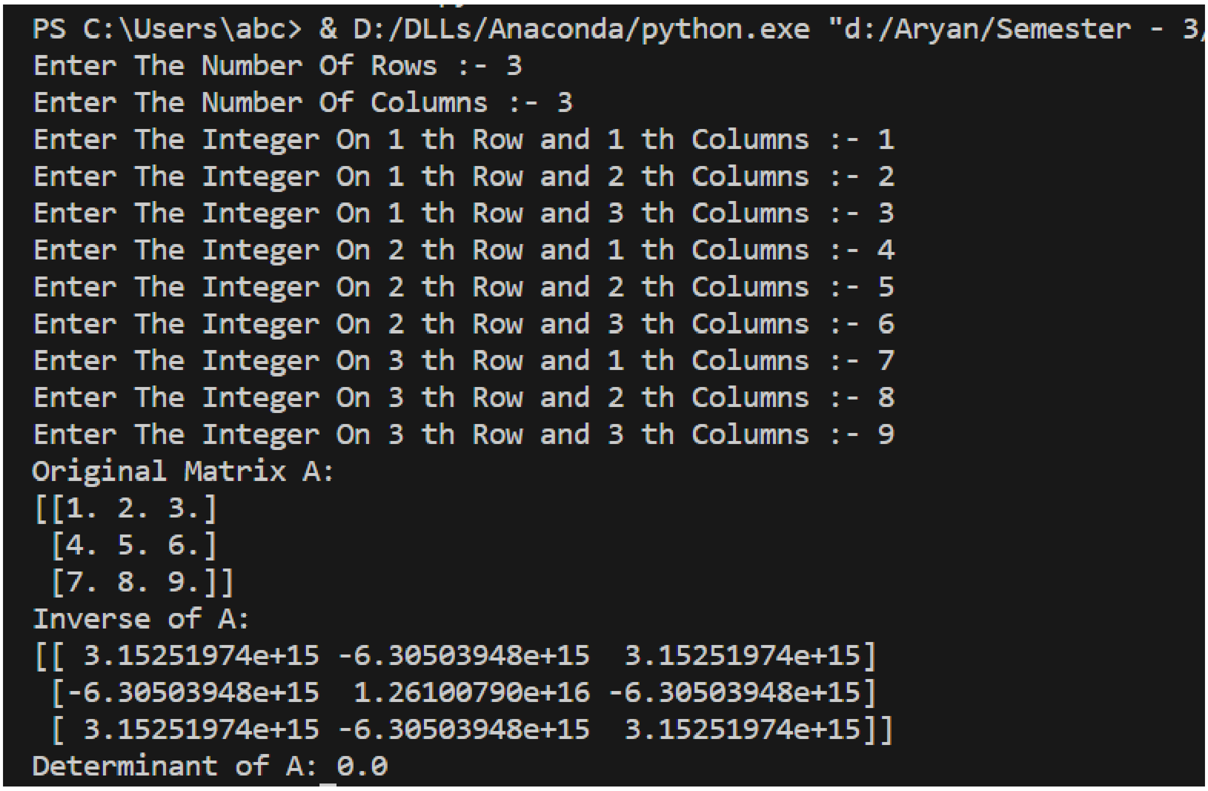
Post Lab

Task 1:- Calculating Inverses and Determinants With scipy.linalg

Python Code:

```
import numpy as np
from scipy.linalg import inv, det
rows = int(input("Enter The Number Of Rows :- "))
columns = int(input("Enter The Number Of Columns :- "))
A = np.empty((rows, columns))
if(rows == columns ) :
    for i in range(0,rows) :
        for j in range(0,columns) :
            element = int(input(f"Enter The Integer On {i + 1 } th Row and {j + 1 } th Columns :-
"))
            A[i][j] = element
    A_inv = inv(A)
    det_A = det(A)
    print("Original Matrix A:")
    print(A)
    print("Inverse of A:")
    print(A_inv)
    print("Determinant of A:", det_A)
else :
    print("Enter Valid Dimmenssions Of A Matrix.")
```

Output:



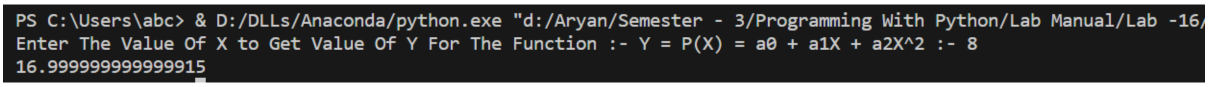
```
PS C:\Users\abc> & D:/DLLs/Anaconda/python.exe "d:/Aryan/Semester - 3/
Enter The Number Of Rows :- 3
Enter The Number Of Columns :- 3
Enter The Integer On 1 th Row and 1 th Columns :- 1
Enter The Integer On 1 th Row and 2 th Columns :- 2
Enter The Integer On 1 th Row and 3 th Columns :- 3
Enter The Integer On 2 th Row and 1 th Columns :- 4
Enter The Integer On 2 th Row and 2 th Columns :- 5
Enter The Integer On 2 th Row and 3 th Columns :- 6
Enter The Integer On 3 th Row and 1 th Columns :- 7
Enter The Integer On 3 th Row and 2 th Columns :- 8
Enter The Integer On 3 th Row and 3 th Columns :- 9
Original Matrix A:
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
Inverse of A:
[[ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]
 [-6.30503948e+15  1.26100790e+16 -6.30503948e+15]
 [ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]]
Determinant of A: 0.0
```

Task 2:- Calculating Inverses and Determinants With scipy.linalg

Python Code:

```
import numpy as np
from scipy.interpolate import lagrange
x = np.array([0, 1, 2, 3])
y = np.array([1, 3, 5, 7])
p = lagrange(x, y)
x_new = float(input("Enter The Value Of X to Get Value Of Y For The Function :- Y = P(X) = a0 + a1X + a2X^2 :- "))
y_new = p(x_new)
print(y_new)
```

Output:



```
PS C:\Users\abc> & D:/DLLs/Anaconda/python.exe "d:/Aryan/Semester - 3/Programming With Python/Lab Manual/Lab -16/
Enter The Value Of X to Get Value Of Y For The Function :- Y = P(X) = a0 + a1X + a2X^2 :- 8
16.999999999999915
```