**DATA SCIENCE & MACHINE LEARNING**

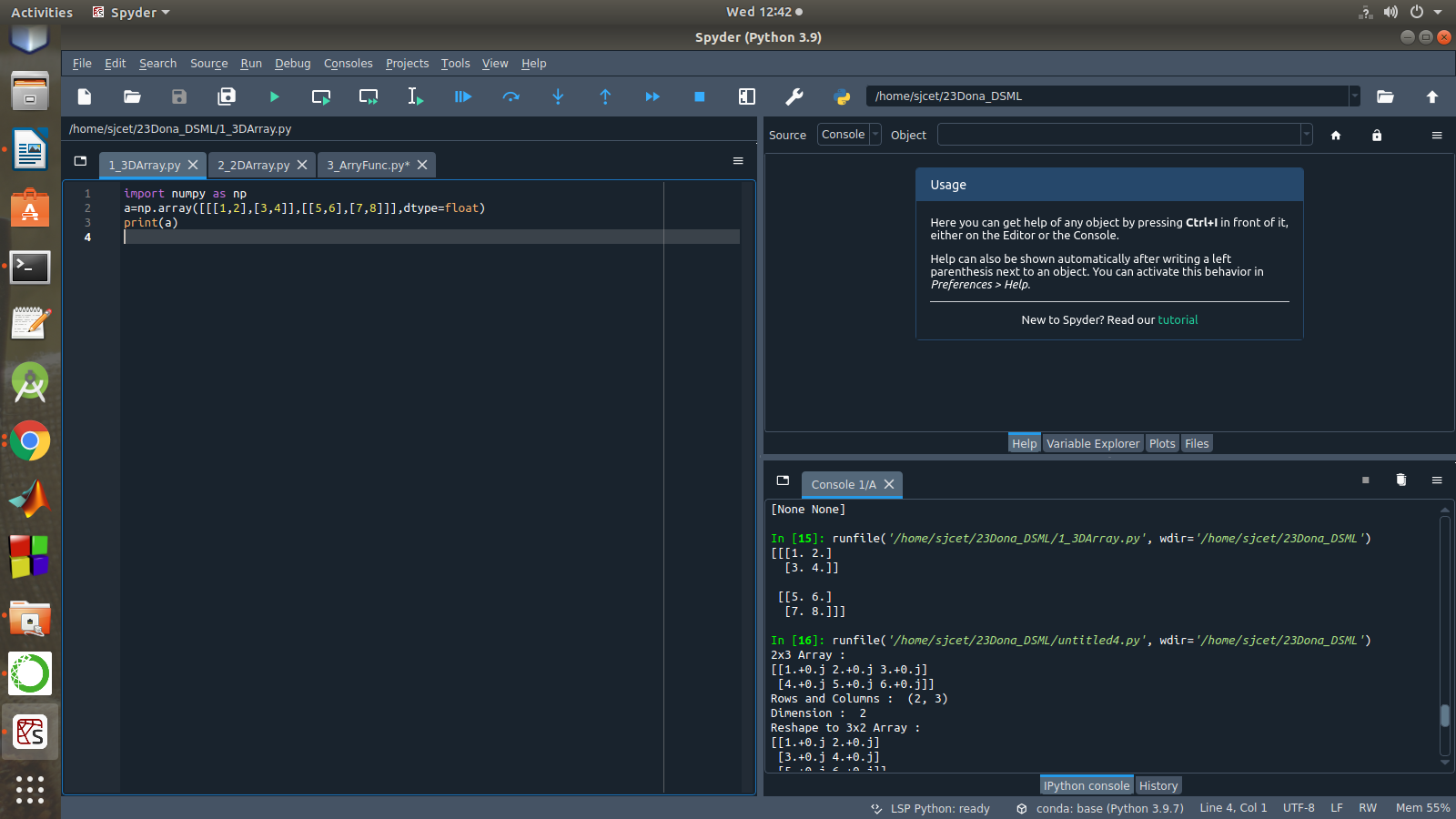
**LAB CYCLE 2**

1. **Create a three dimensional array specifying float data type and print it.**

import numpy as np

a=np.array([[[1,2],[3,4]],[[5,6],[7,8]]],dtype=float)

print(a)



1. **Create a 2 dimensional array (2X3) with elements belonging to complex data type and print it. Also display**
2. **the no: of rows and columns**
3. **dimension of an array**
4. **reshape the same array to 3X2**

import numpy as np

a=np.array([[1,2,3],[4,5,6]],dtype=complex)

print("2x3 Array :")

print(a)

print("Rows and Columns : ",a.shape)

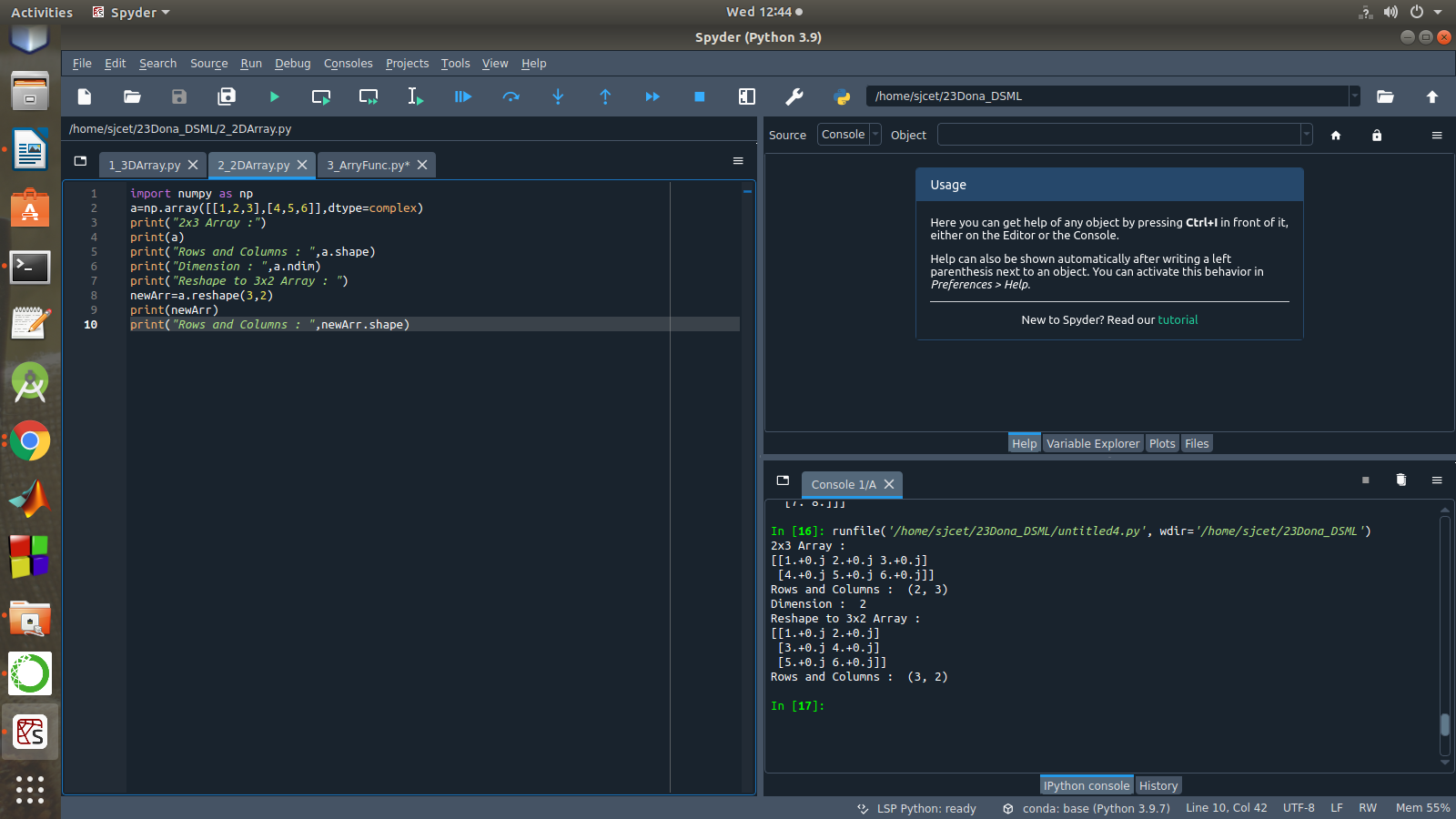
print("Dimension : ",a.ndim)

print("Reshape to 3x2 Array : ")

newArr=a.reshape(3,2)

print(newArr)

print("Rows and Columns : ",newArr.shape)



1. **Familiarize with the functions to create**
2. **an uninitialized array**
3. **array with all elements as 1**
4. **all elements as 0**

import numpy as np

print("Uninitialized Array")

a = np.full([2, 3], None)

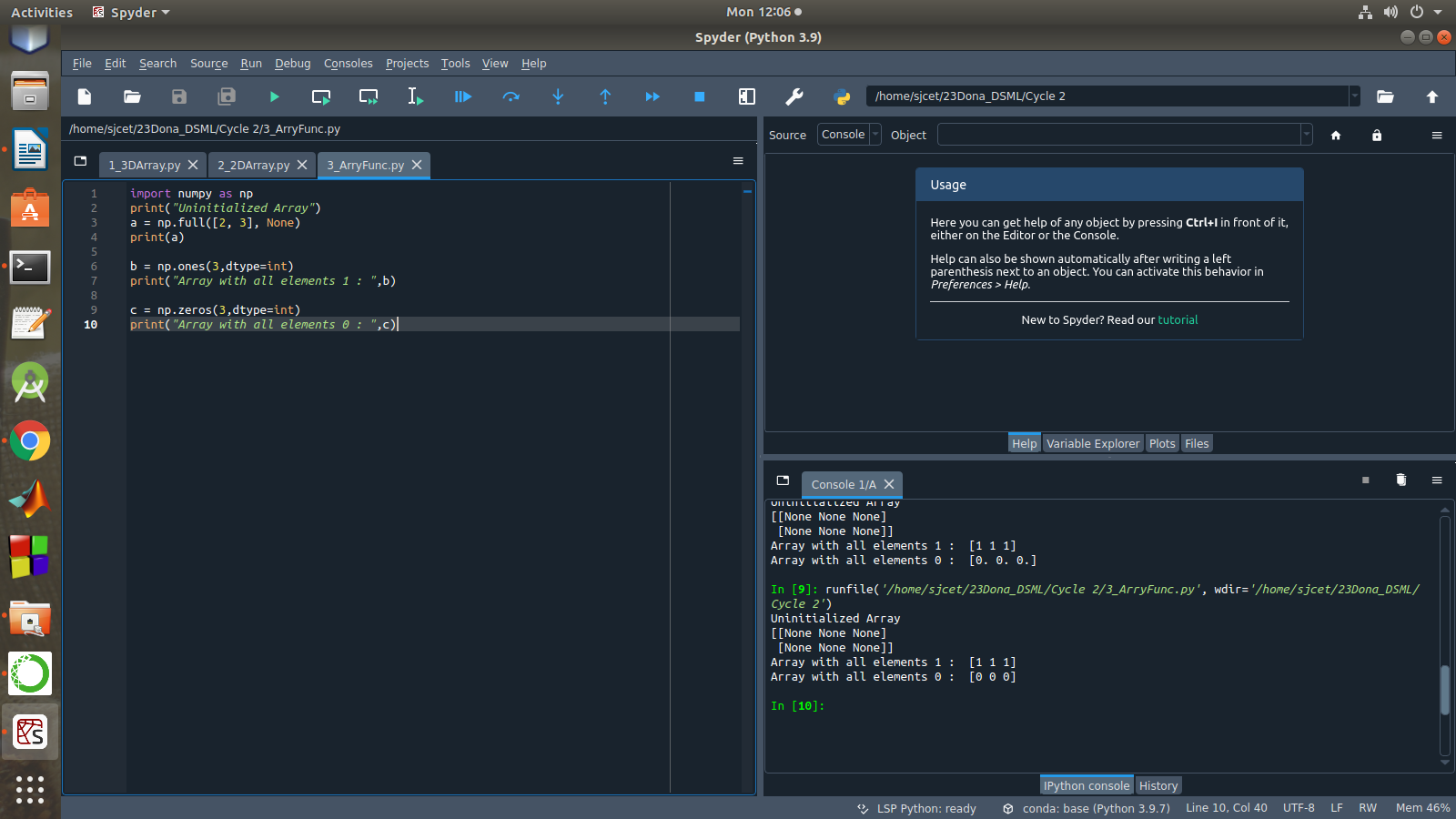
print(a)

b = np.ones(3,dtype=int)

print("Array with all elements 1 : ",b)

c = np.zeros(3,dtype=int)

print("Array with all elements 0 : ",c)



1. **Create an one dimensional array using arange function containing 10 elements.**

**Display**

1. **First 4 elements**
2. **Last 6 elements**
3. **Elements from index 2 to 7**

import numpy as np

a = np.arange(1, 11, 1)

print(a)

element1 = a[:4]

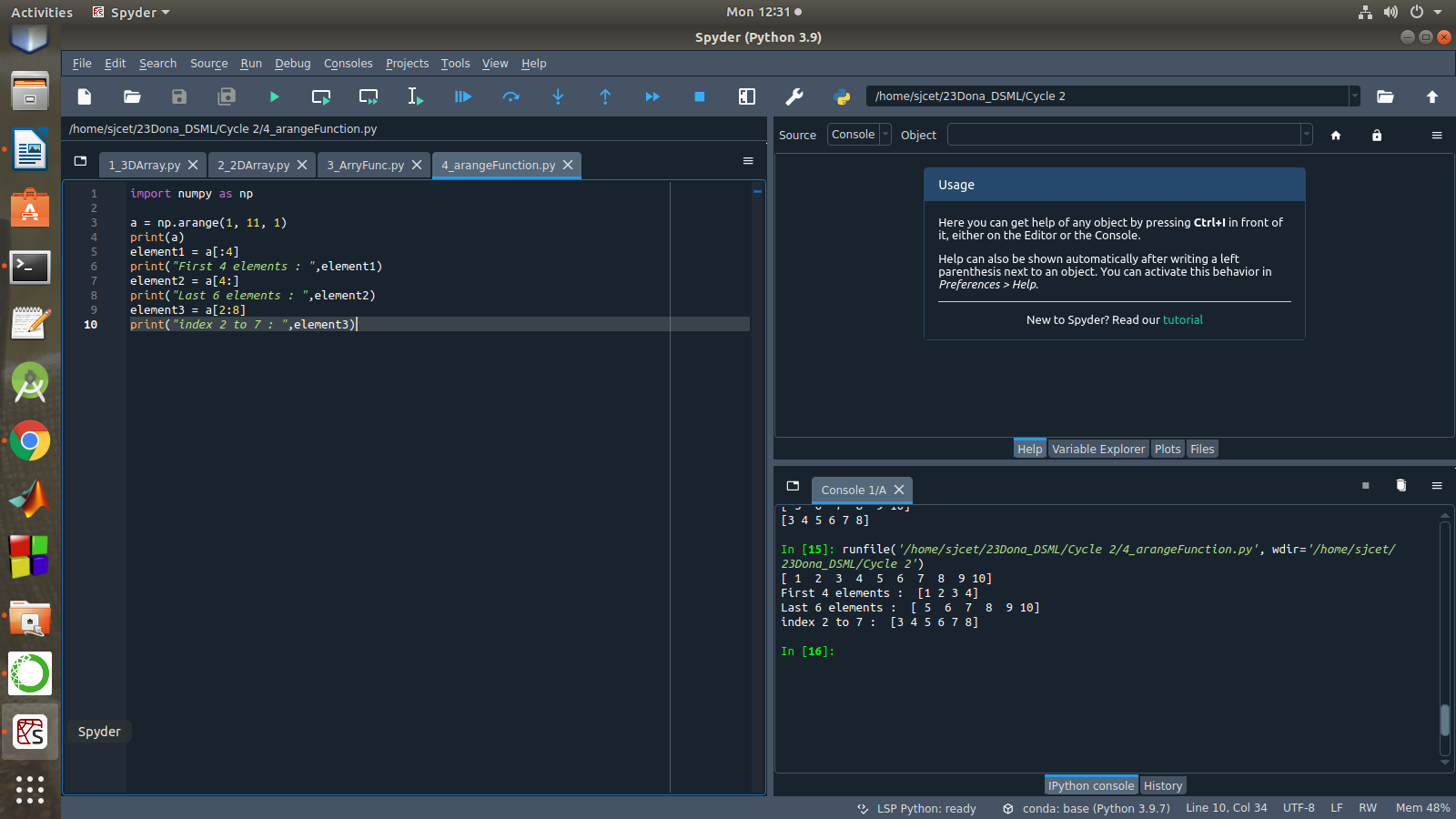
print("First 4 elements : ",element1)

element2 = a[4:]

print("Last 6 elements : ",element2)

element3 = a[2:8]

print("index 2 to 7 : ",element3)



1. **Create an 1D array with arange containing first 15 even numbers as elements**
2. **Elements from index 2 to 8 with step 2(also demonstrate the same using slice function)**
3. **Last 3 elements of the array using negative index**
4. **Alternate elements of the array**
5. **Display the last 3 alternate elements**

import numpy as np

n = int(input("How many elements you want? : "))

n = 2\*n

a=np.arange(0,n,2)

print("Even Array : ",a)

print("Elements from index 2 to 8 : ",a[2:9])

x = slice(2,9)

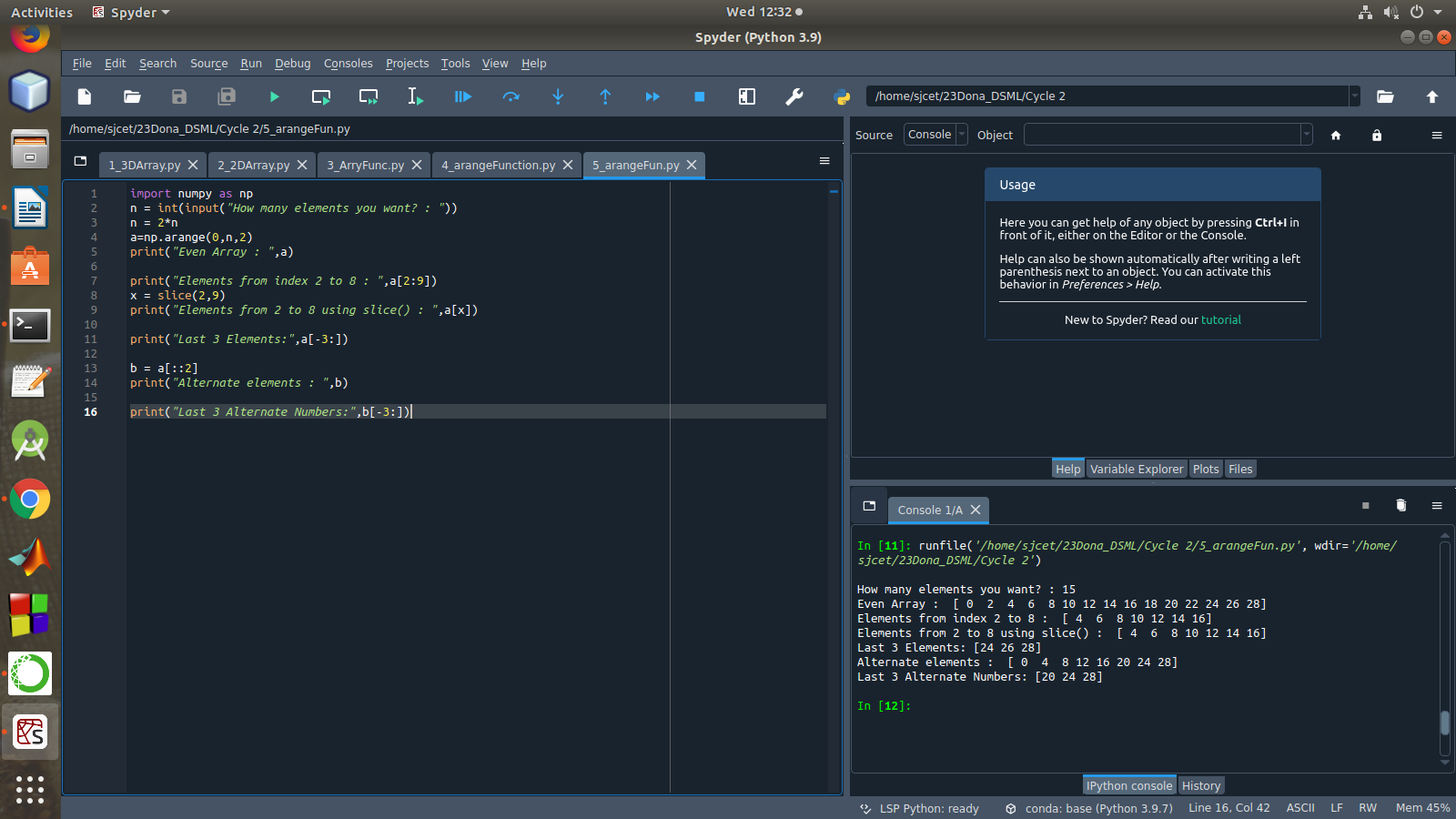
print("Elements from 2 to 8 using slice() : ",a[x])

print("Last 3 Elements:",a[-3:])

b = a[::2]

print("Alternate elements : ",b)

print("Last 3 Alternate Numbers:",b[-3:])



1. **Create a 2 Dimensional array with 4 rows and 4 columns.**
2. **Display all elements excluding the first row**
3. **Display all elements excluding the last column**
4. **Display the elements of 1st and 2nd column in 2nd and 3rd row**
5. **Display the elements of 2nd and 3rd column**
6. **Display 2nd and 3rd element of 1st row**
7. **Display the elements from indices 4 to 10 in descending order(use –values)**

import numpy as np

x = np.array([[2, 4, 6,1], [6, 8, 10,1],[1, 2, 1,1], [1, 1, 1,1]])

print("4x4 2D Array :")

print(x)

print("Display all elements excluding the first row")

print(x[1:])

print("Display all elements excluding the last column")

print(x[:, :3])

print("Display the elements of 1st & 2nd column in 2nd & 3rd row")

print(x[1:3, 0:2])

print("Display the elements of 2 nd and 3 rd column")

print(x[:, 1:3])

print("Display 2 nd and 3 rd element of 1 st row")

print(x[0,1])

print(x[0,2])

print("Display the elements from indices 4 to 10 in descending order(use-values)")

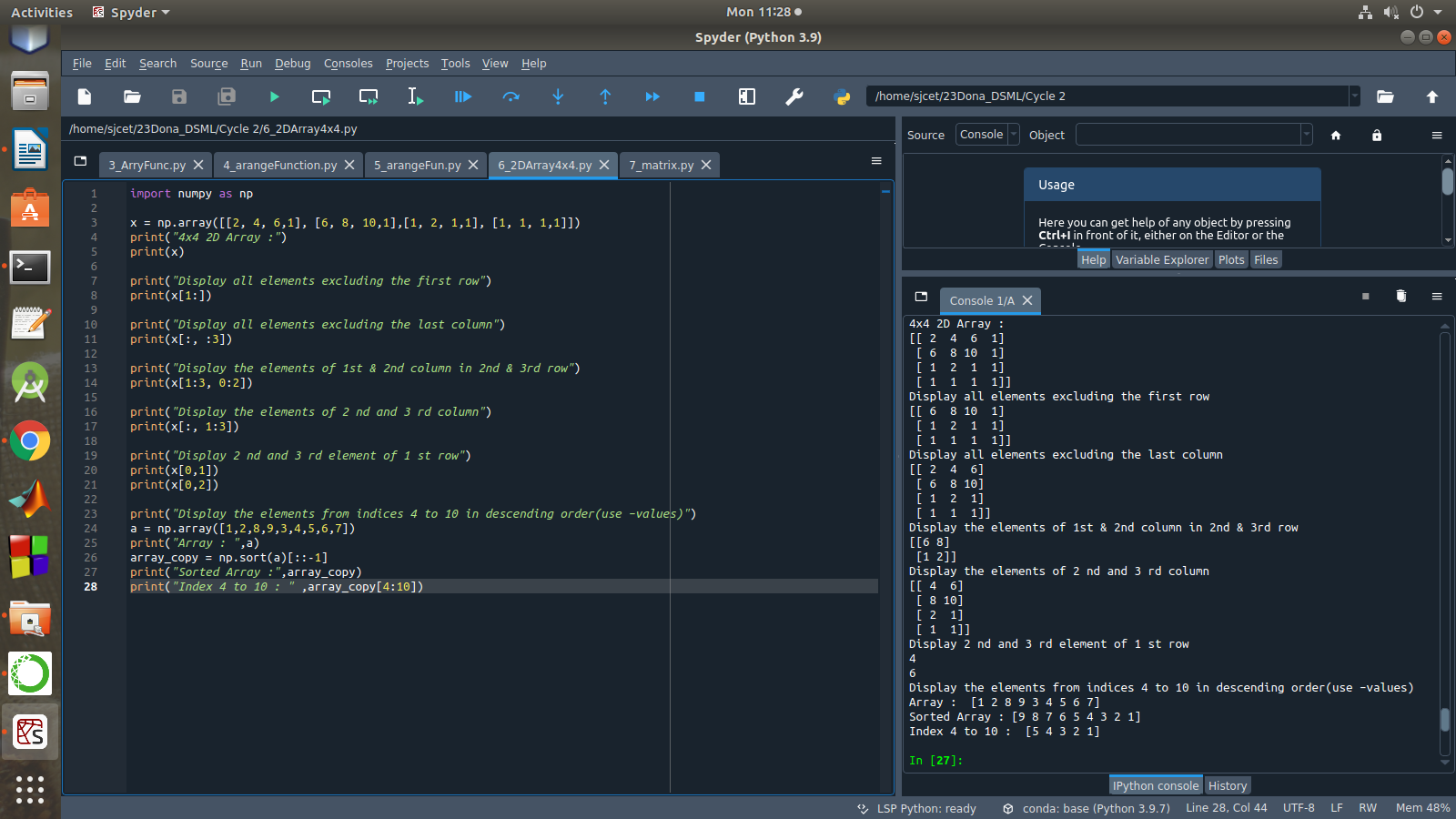
a = np.array([1,2,8,9,3,4,5,6,7])

print("Array : ",a)

array\_copy = np.sort(a)[::-1]

print("Sorted Array :",array\_copy)

print("Index 4 to 10 : " ,array\_copy[4:10])



1. **Create two 2D arrays using array object and**
2. **Add the 2 matrices and print it**
3. **Subtract 2 matrices**
4. **Multiply the individual elements of matrix**
5. **Divide the elements of the matrices**
6. **Perform matrix multiplication**
7. **Display transpose of the matrix**
8. **Sum of diagonal elements of a matrix**

import numpy as np

M1 = np.array([[9, 2], [5, 8]])

M2 = np.array([[3, 4], [1, 4]])

print("First matrix \n ",M1)

print("Second matrix \n ",M2)

add = M1 + M2

print("Matrix addition\n",add)

sub = M1 - M2

print("Matrix Substract\n",sub)

mul = M1 \* M2

print("Multiply the individual elements of matrix\n",mul)

div = M1 / M2

print("Divide the elements of the matrices\n",div)

M3 = M1.dot(M2)

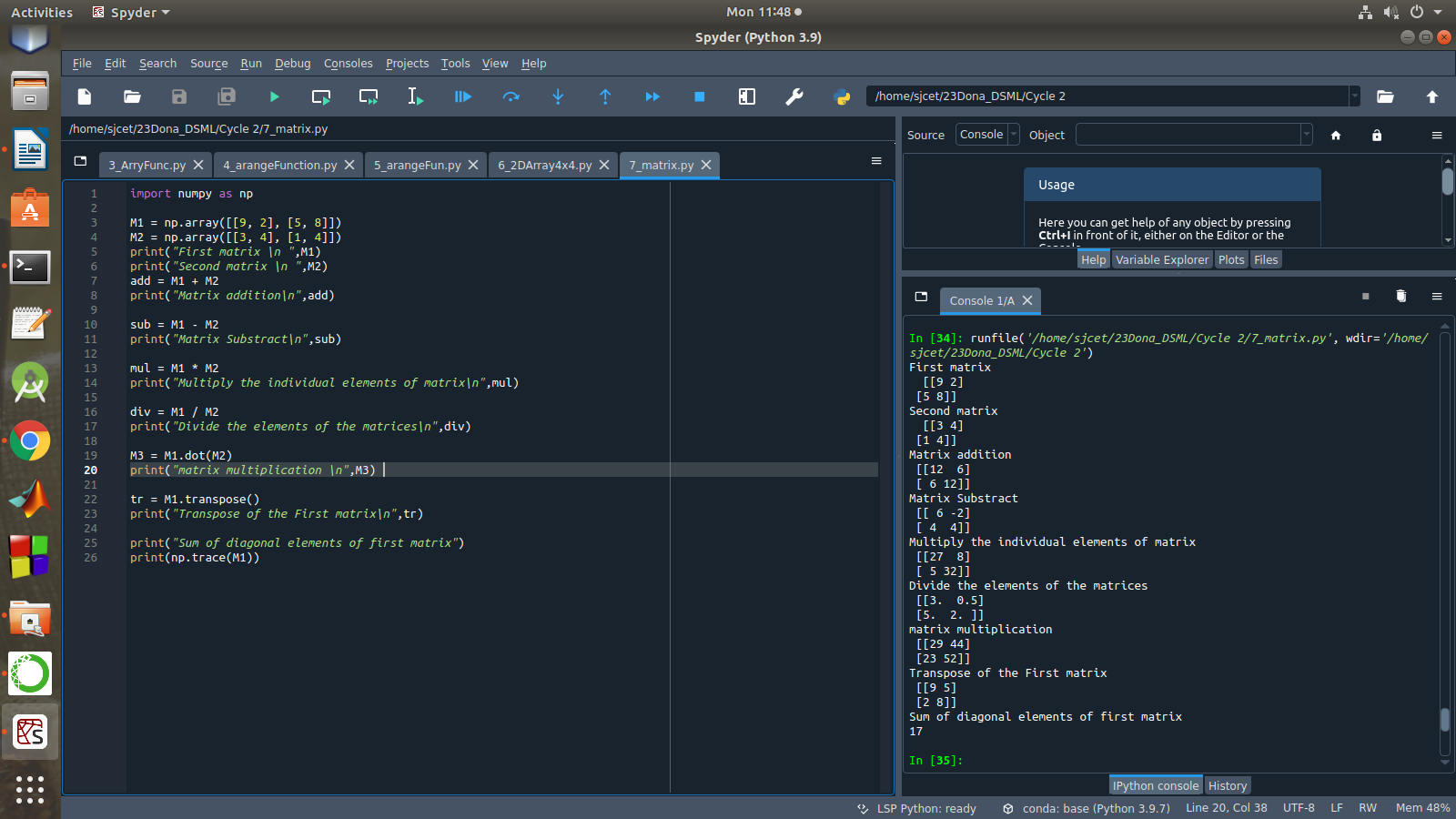
print("matrix multiplication \n",M3)

tr = M1.transpose()

print("Transpose of the First matrix\n",tr)

print("Sum of diagonal elements of first matrix")

print(np.trace(M1))



1. **Demonstrate the use of insert() function in 1D and 2D array.**

import numpy as np

arr1 = np.arange(10, 16)

print("1D ARRAY ")

print("The array is: ", arr1)

obj = 2

value = 40

arr = np.insert(arr1, obj, value, axis=None)

print("After inserting the new array is: \n",arr)

print("Shape of the new array is : ", np.shape(arr))

print("2D ARRAY ")

arr1 = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9), (50, 51, 52)])

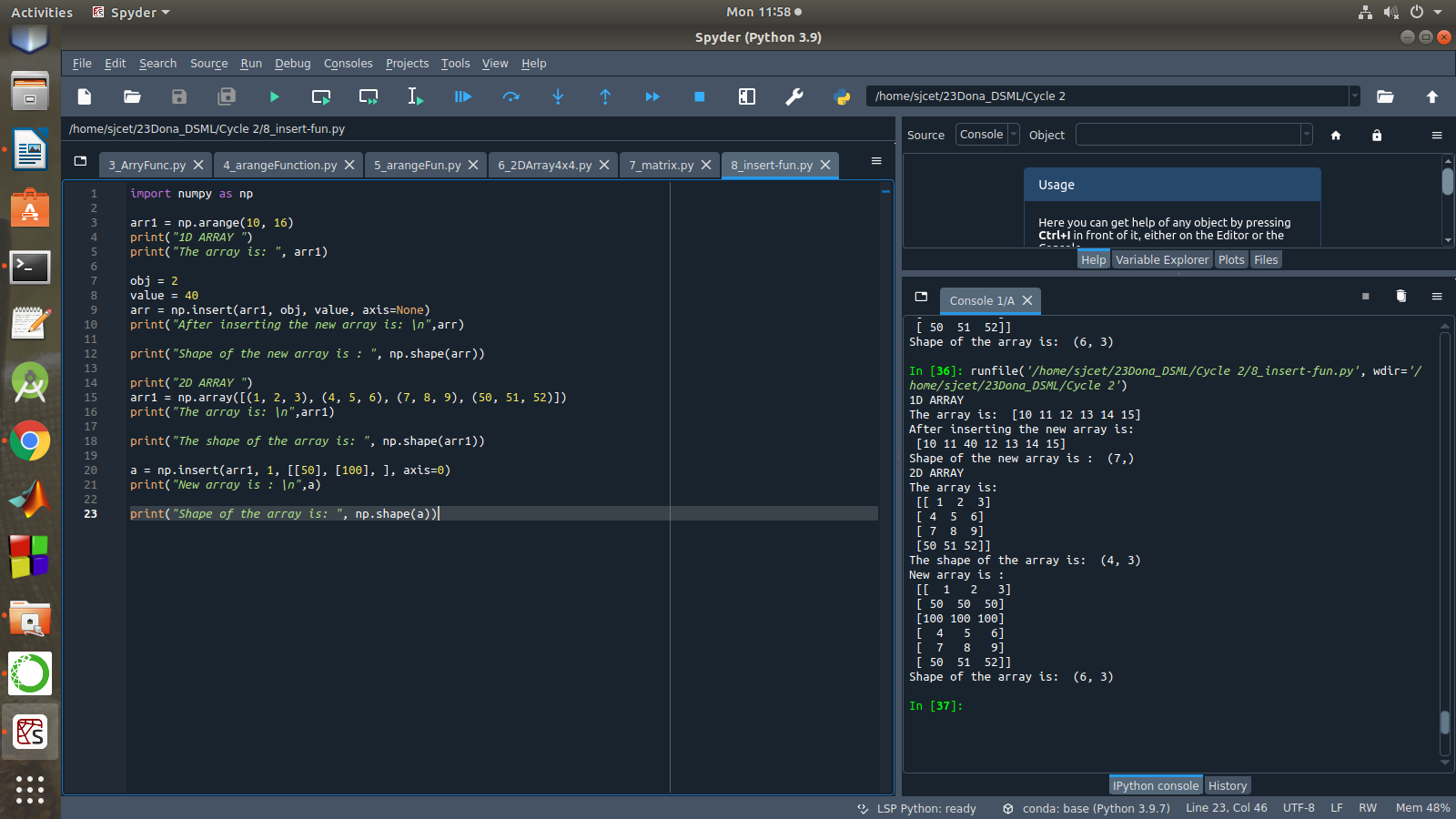
print("The array is: \n",arr1)

print("The shape of the array is: ", np.shape(arr1))

a = np.insert(arr1, 1, [[50], [100], ], axis=0)

print("New array is : \n",a)

print("Shape of the array is: ", np.shape(a))



1. **Demonstrate the use of diag() function in 1D and 2D array.**

import numpy as np

a= np.array([[3,6,7,8]])

b=np.array([[3,6,8,7], [4,2,1,0],[3,1,3,3],[1,1,2,2]])

print("1d array : ",a)

print("2D array \n",b)

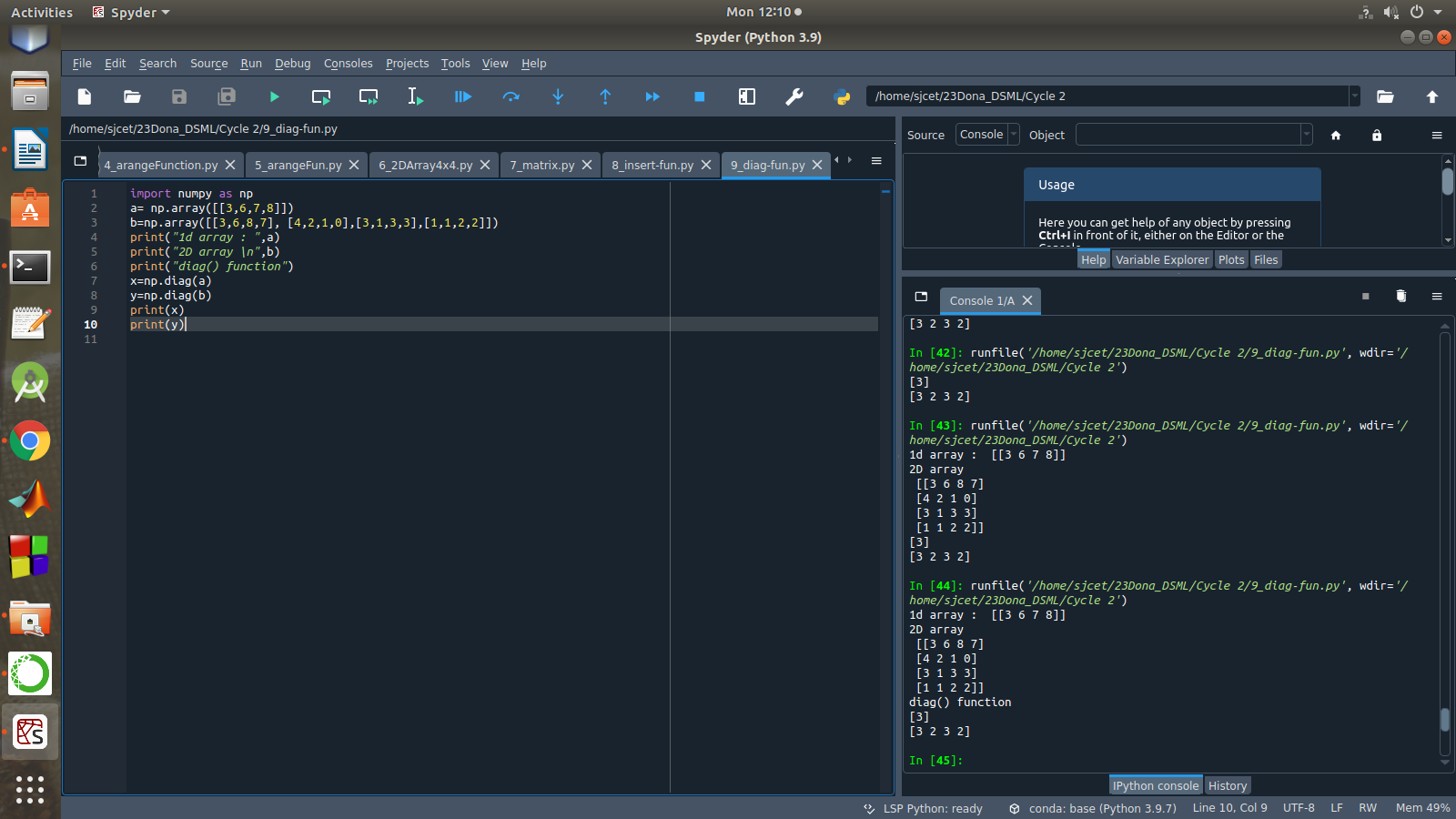
print("diag() function")

x=np.diag(a)

y=np.diag(b)

print(x)

print(y)



1. **Demonstarte the use of append() function in 1D and 2D array.**

import numpy as np

a = np.array([1,2,3])

b = np.array([[1,2,3],[4,5,6]])

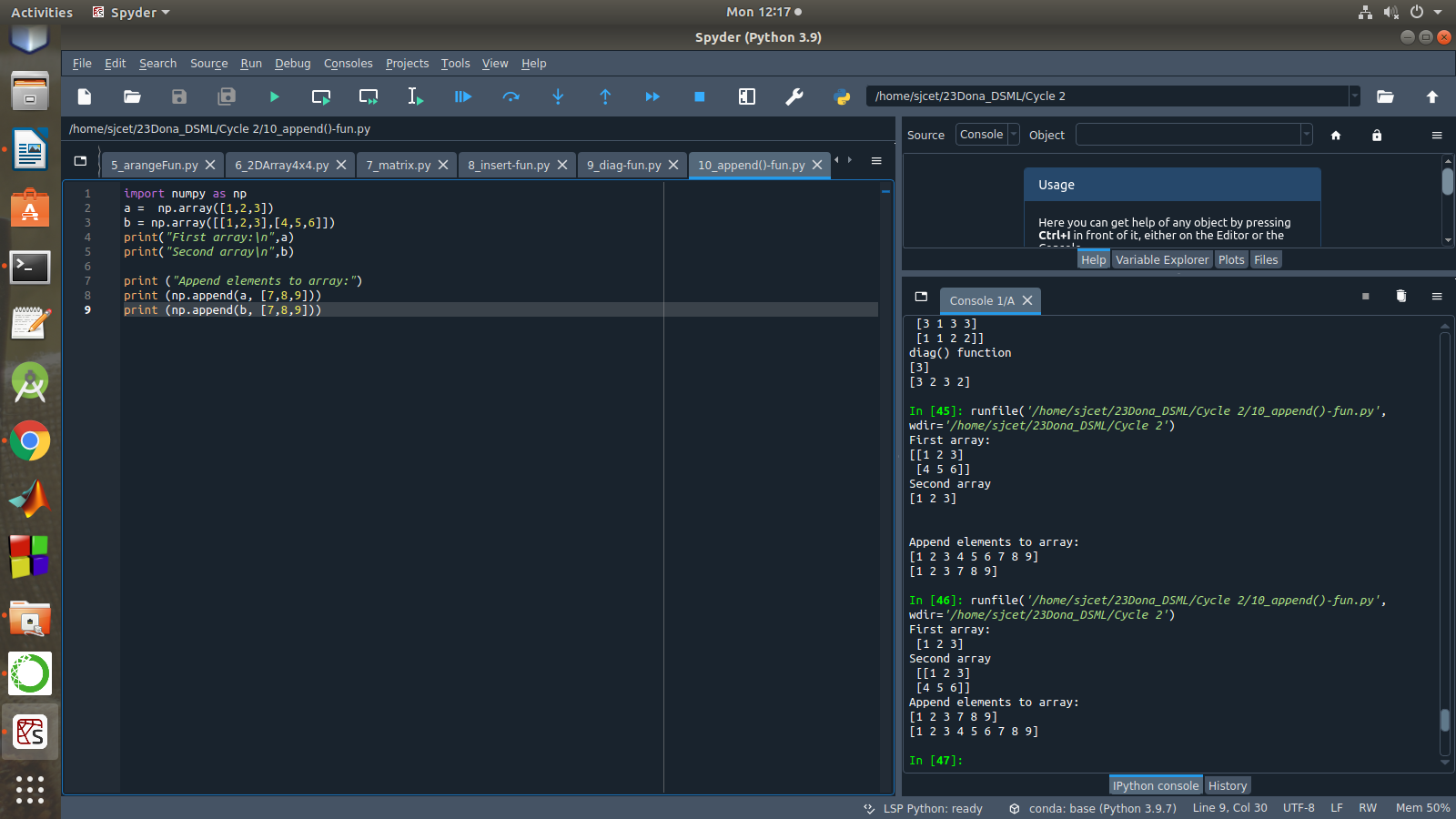
print("First array:\n",a)

print("Second array\n",b)

print ("Append elements to array:")

print (np.append(a, [7,8,9]))

print (np.append(b, [7,8,9]))



1. **Demonstarte the use of sum() function in 1D and 2D array.**

import numpy as np

a=np.array([4,5])

b=np.array([[1,2,3],[4,5,6]])

print("1D array\n",a)

print("2D array\n",b)

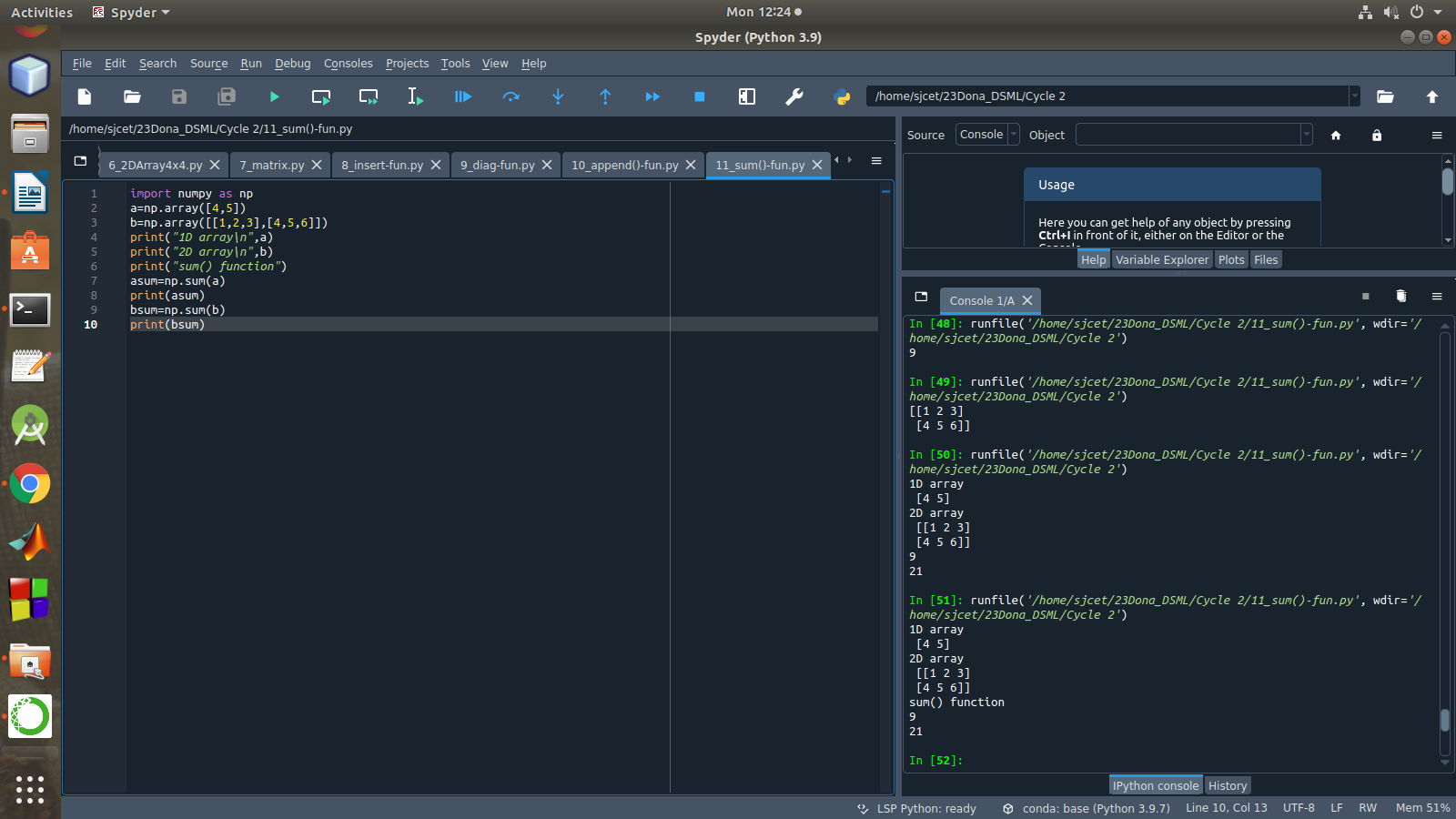
print("sum() function")

asum=np.sum(a)

print(asum)

bsum=np.sum(b)

print(bsum)



**PART 2**

1. **Create a square matrix with random integer values(use randint()) and use appropriate functions to find:**

**i) inverse**

**ii) rank of matrix**

**iii)** **Determinant**

**iv) transform matrix into 1D array**

**v) eigen values and vectors**

import numpy as np

import numpy as nf

from numpy.linalg import eig

mat = np.random.randint(10, size=(3, 3))

array = nf.random.randint(10, size=(3, 3))

print("Square matrix \n",mat)

M\_inverse = np.linalg.inv(mat)

print("Inverse of the matrix\n",M\_inverse)

rank = np.linalg.matrix\_rank(mat)

print("Rank of the given Matrix \n",rank)

det= np.linalg.det(mat)

print("Determinant of the given Matrix \n",det)

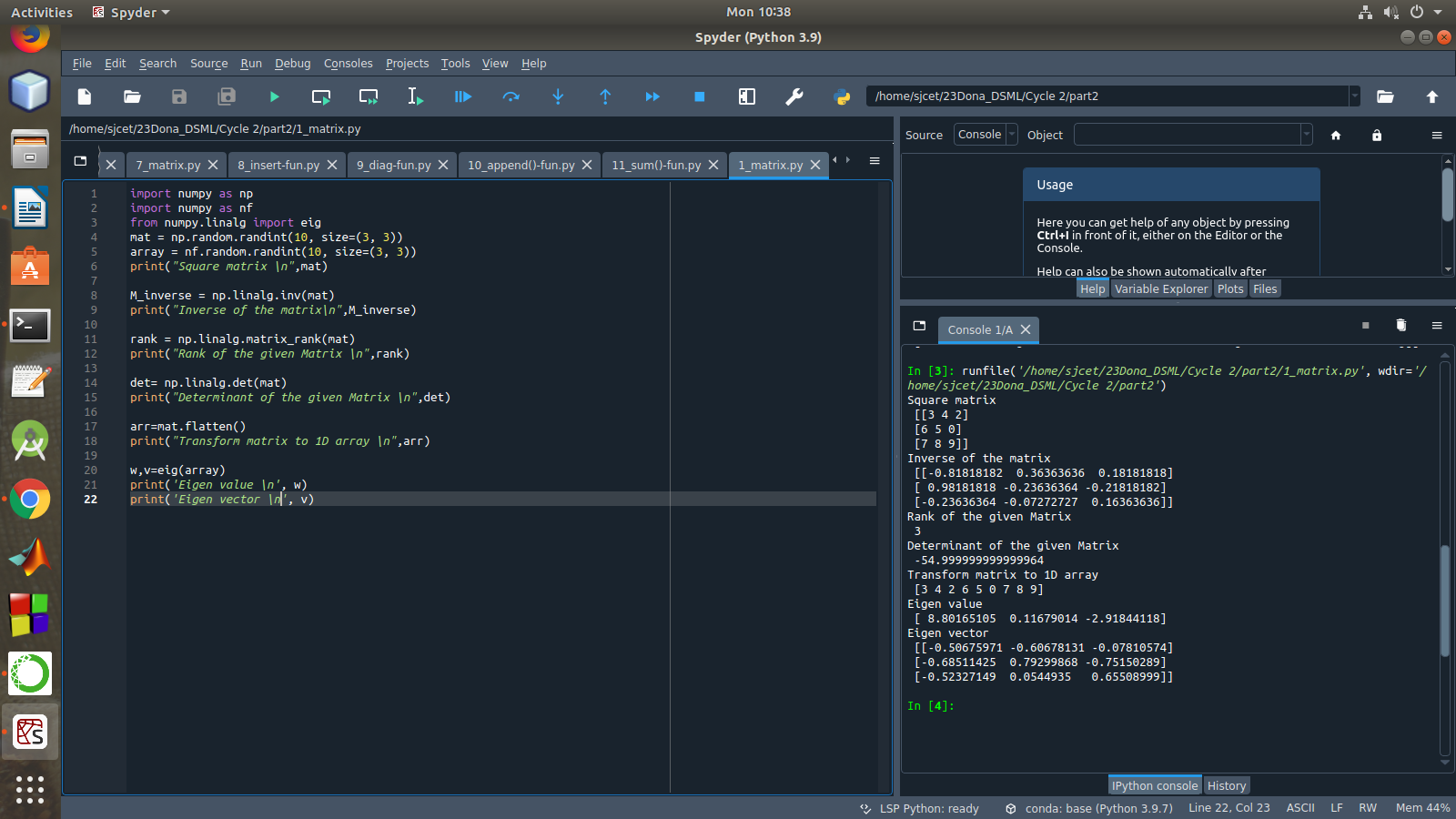
arr=mat.flatten()

print("Transform matrix to 1D array \n",arr)

w,v=eig(array)

print('Eigen value \n', w)

print('Eigen vector \n', v)



**2. Create a matrix X with suitable rows and columns**

**i) Display the cube of each element of the matrix using different methods**

**(use multiply(), \*, power(),\*\*)**

**ii) Display identity matrix of the given square matrix**

**iii) Display each element of the matrix to different powers.**

**iv) Create a matrix Y with same dimension as X and perform the operation X2+2Y**

import numpy as np

mat =np.array([[1, 2, 3],[3,2,4],[2,2,1]])

print("Matrix is....\n",mat)

print("Cubes using \*")

print(mat\*mat\*mat)

print("Cubes using \*\*")

print(mat\*\*3)

print("Cubes using multiply()")

print(np.multiply(mat,(mat\*mat)))

print("Cubes using power()")

print(np.power(mat,3))

print(pow(mat, 3))

b = np.identity(3, dtype = int)

print("Identity matrix:\n", b)

out = np.power(mat, mat)

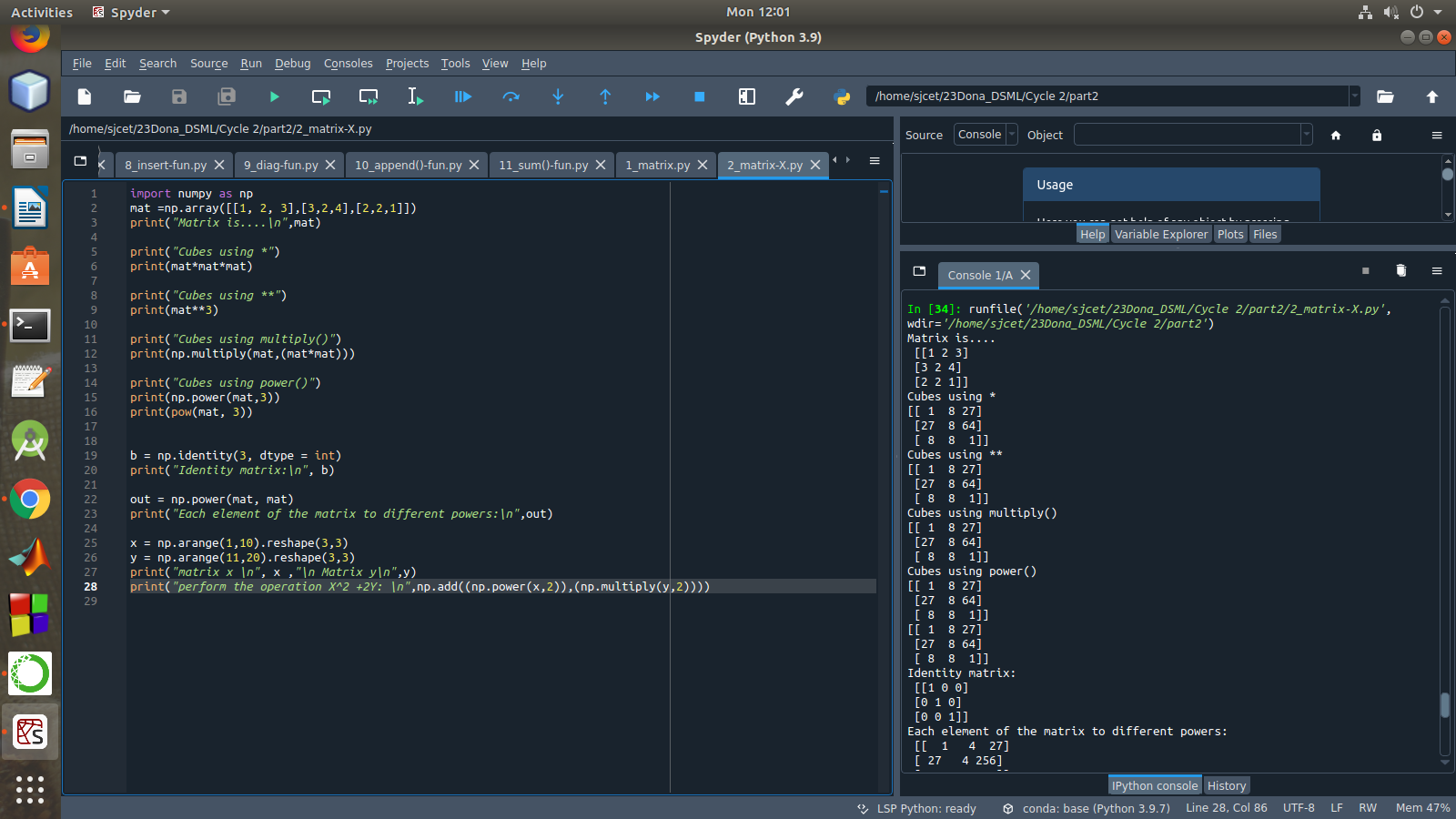
print("Each element of the matrix to different powers:\n",out)

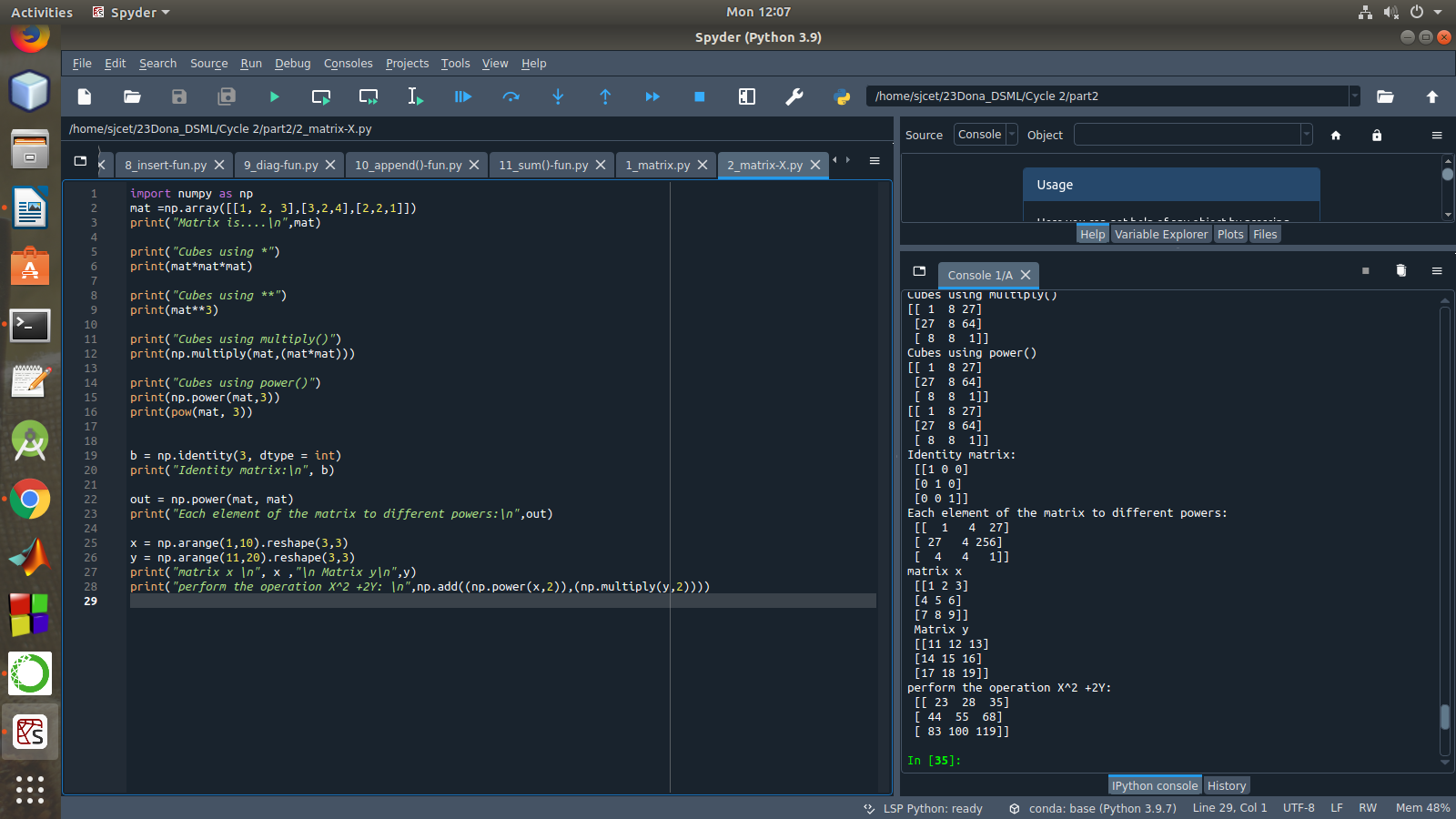
x = np.arange(1,10).reshape(3,3)

y = np.arange(11,20).reshape(3,3)

print("matrix x \n", x ,"\n Matrix y\n",y)

print("perform the operation X^2 +2Y: \n",np.add((np.power(x,2)),(np.multiply(y,2))))





3. **Multiply a matrix with a submatrix of another matrix and replace the same in larger matrix.**



import numpy as np

mat = np.array([[6, 1, 1, 4],

[1, 2, 5, 2],

[1, 5, 7, 3],

[3, 2, 4, 1]])

print("Original Matrix....\n",mat)

sub = mat[1:3, 1:3]

print("Sub matrix....\n",sub)

mat2 = np.array([[1, 4],

[3, 2]])

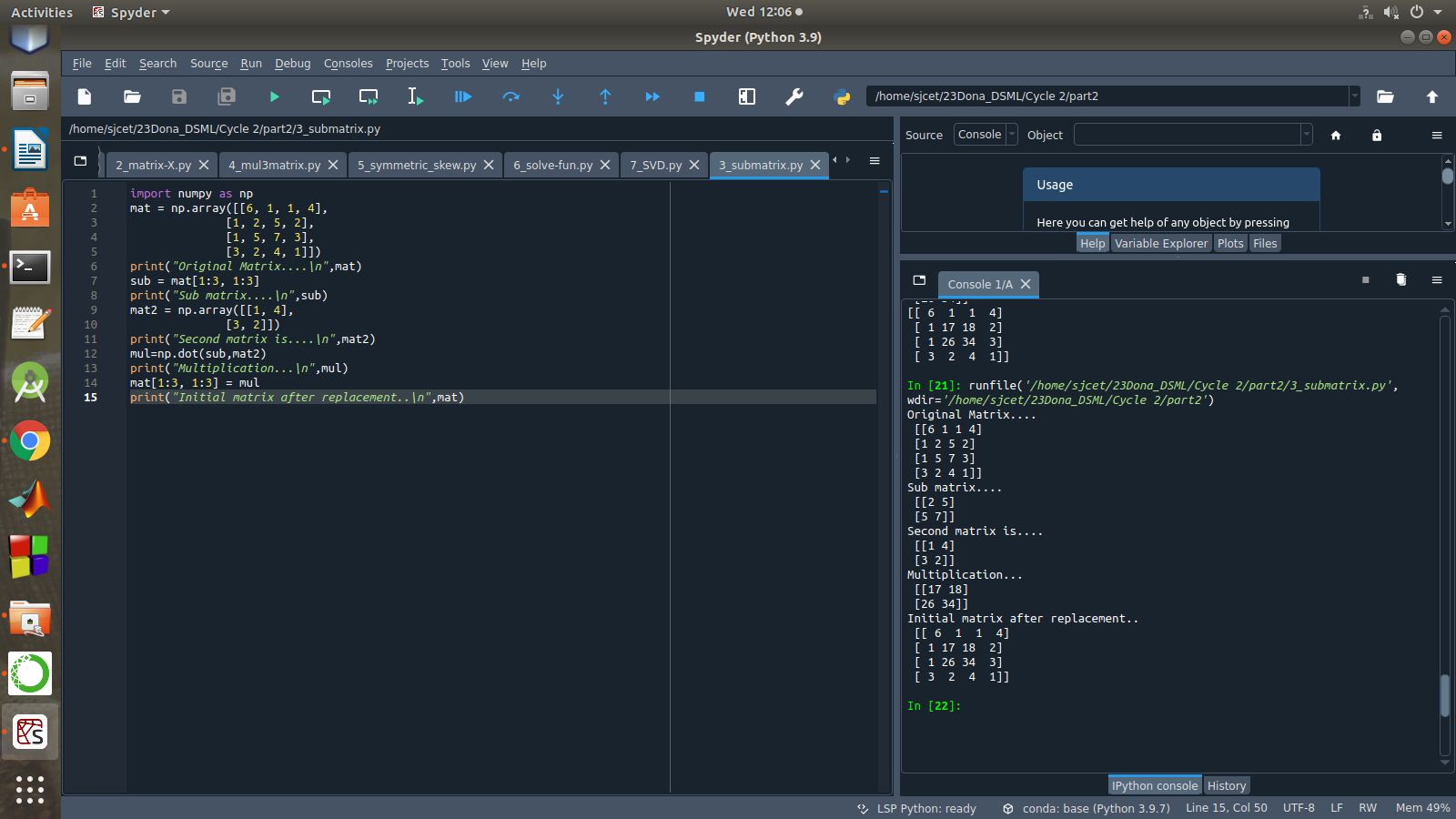
print("Second matrix is....\n",mat2)

mul=np.dot(sub,mat2)

print("Multiplication...\n",mul)

mat[1:3, 1:3] = mul

print("Initial matrix after replacement..\n",mat)



**4. Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of the 3 matrices.**

import numpy as np

M1 = np.array([[3, 6], [4, 2]])

M2 = np.array([[9, 2], [1, 2]])

M3=np.array([[2,4],[3,1]])

Mul = M1.dot(M2)

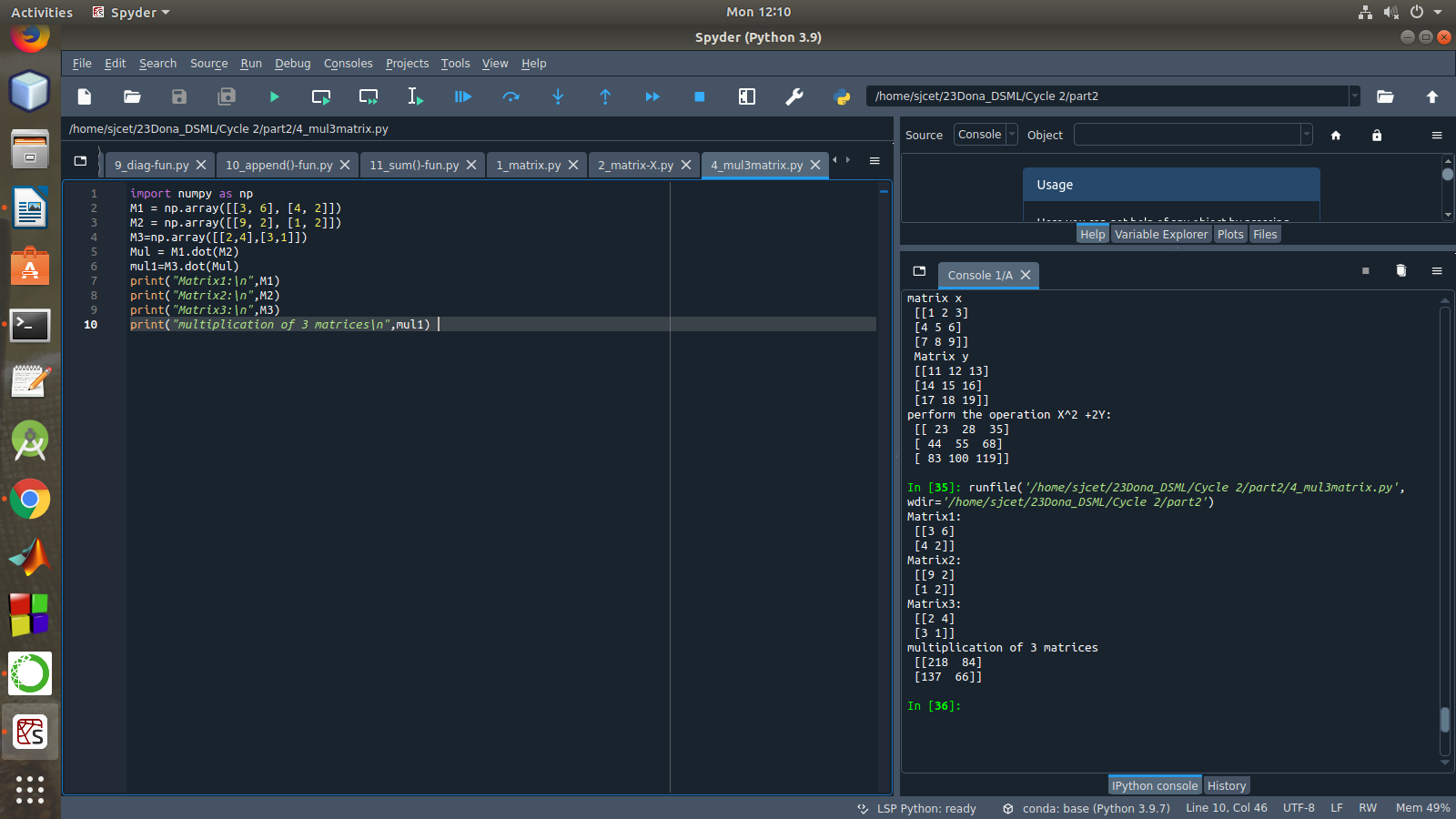
mul1=M3.dot(Mul)

print("Matrix1:\n",M1)

print("Matrix2:\n",M2)

print("Matrix3:\n",M3)

print("multiplication of 3 matrices\n",mul1)



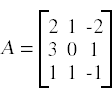
**5. Write a program to check whether given matrix is symmetric or Skew Symmetric.**

Solving systems of equations with numpy

One of the more common problems in linear algebra is solving a matrix-vector equation.

Here is an example. We seek the vector x that solves the equation

A X = b

where  

And X=A-1 b.

**Numpy provides a function called solve for solving such eauations.**

import numpy as np

A = np.array([[6, 1, 1],

[1, -2, 5],

[1, 5, 7]])

print("Original Matrix\n",A)

inv=np.transpose(A)

print ("Transpose matrix\n",inv)

neg=np.negative(A)

comparison = A == inv

comparison1 = inv== neg

equal\_arrays = comparison.all()

skew=comparison1.all()

if equal\_arrays :

print("Symmetric")

else:

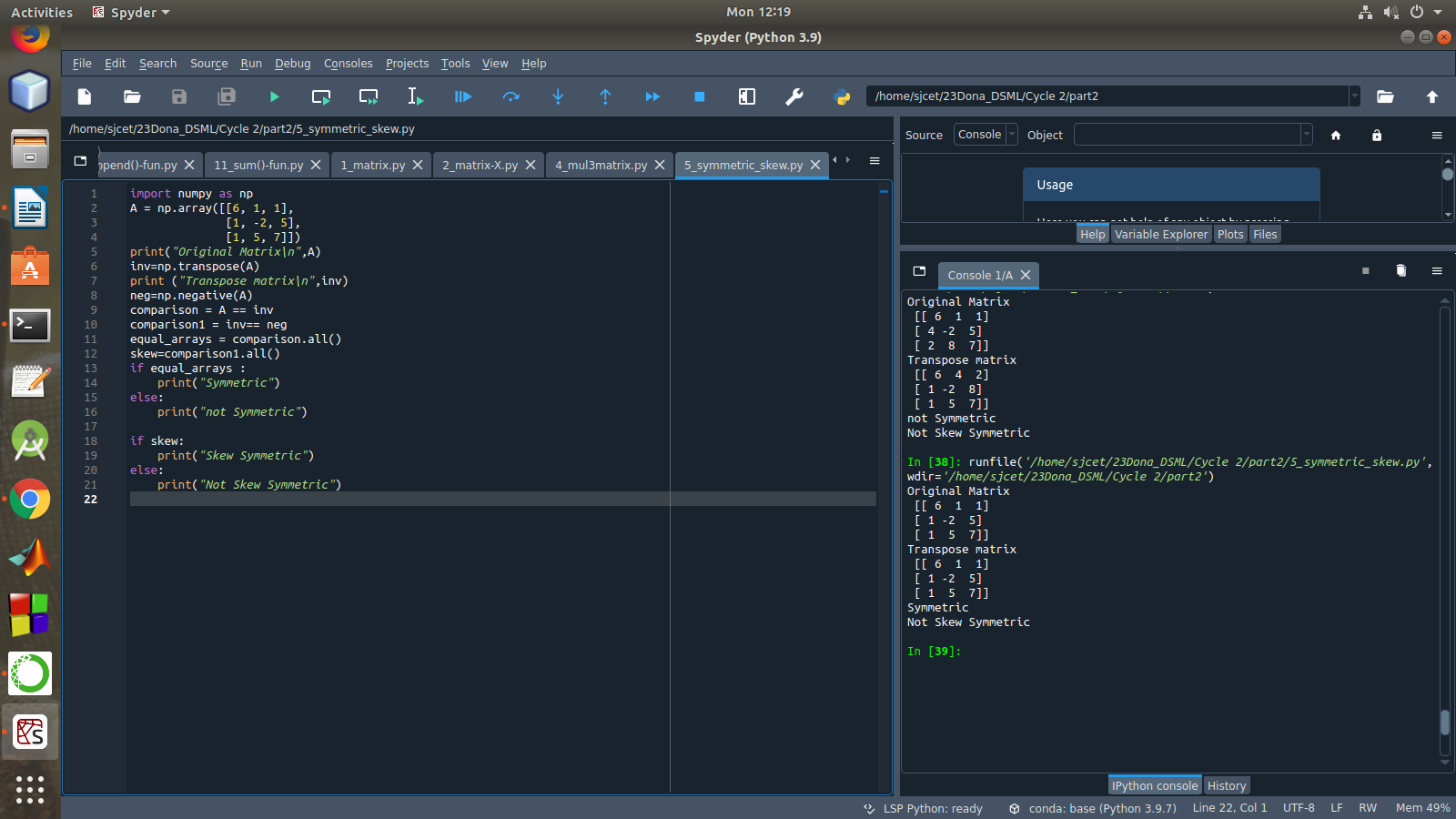
print("not Symmetric")

if skew:

print("Skew Symmetric")

else:

print("Not Skew Symmetric")



6. **Write a program to find out the value of X using solve(), given A and b as above**

import numpy as np

A = np.array([[2, 1, -2],

[3, 0, 1],

[1, 1, -1]])

b=np.array([[3],

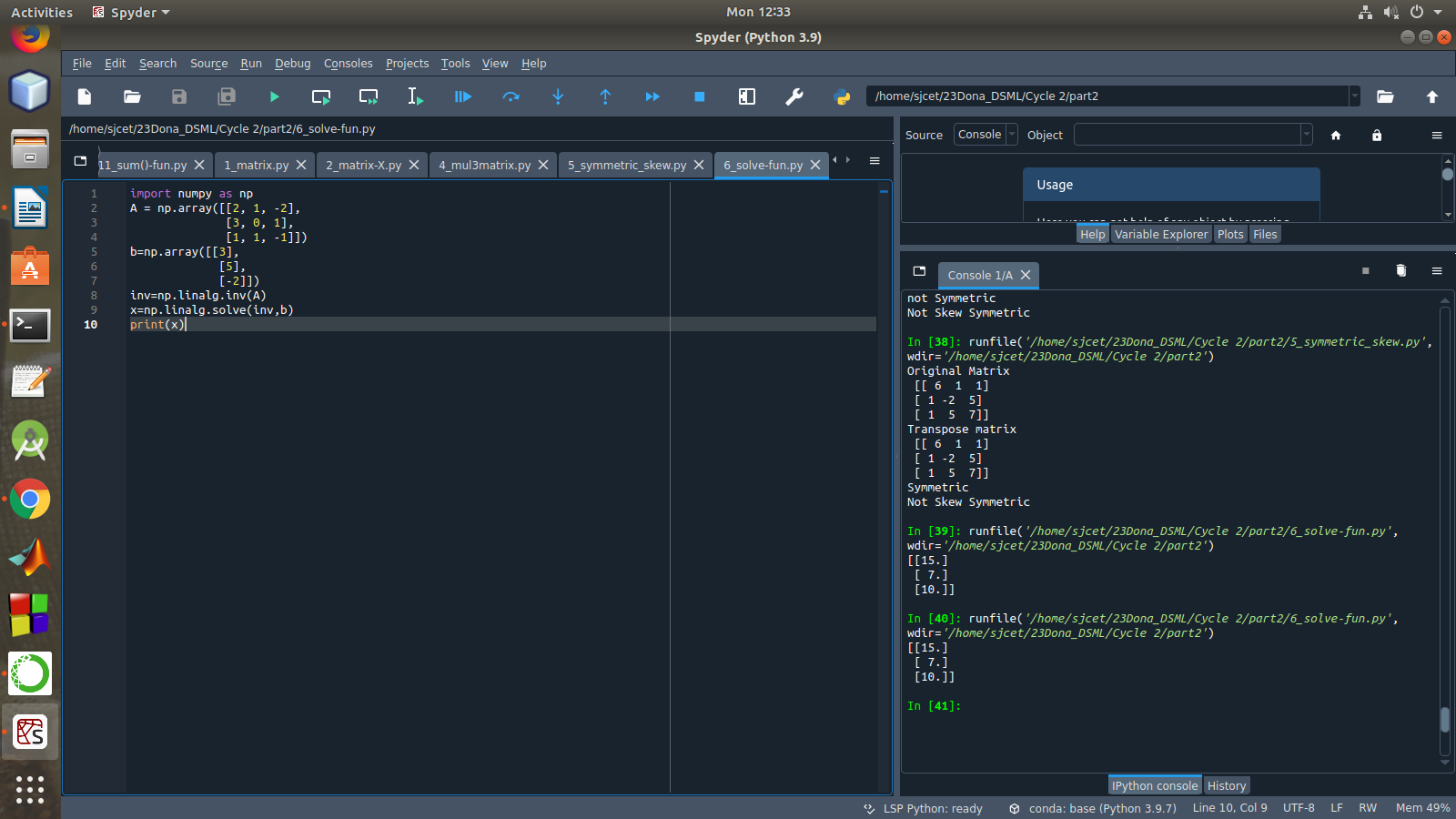
[5],

[-2]])

inv=np.linalg.inv(A)

x=np.linalg.solve(inv,b)

print(x)



7. **Write a program to perform the SVD of a given matrix. Also reconstruct the given matrix from the 3 matrices obtained after performing SVD.**

from numpy import array

from scipy.linalg import svd

from numpy import diag

from numpy import dot

from numpy import zeros

# define a matrix

A = array([[1, 2], [3, 4], [5, 6]])

print(A)

# SVD

U, s, VT = svd(A)

print("first" ,U)

print("second",s)

print("3rd" ,VT)

Sigma = zeros((A.shape[0], A.shape[1]))

# populate Sigma with n x n diagonal matrix

Sigma[:A.shape[1], :A.shape[1]] = diag(s)

# reconstruct matrix

B = U.dot(Sigma.dot(VT))

print(B)

