1.Program to Print all non-Prime Numbers in an Interval.

**CODE**

#Take the input from the user:

lower = int(input("Enter lower range: "))

upper = int(input("Enter upper range: "))

for num in range(lower,upper + 1):

if num > 1:

for i in range(2,num):

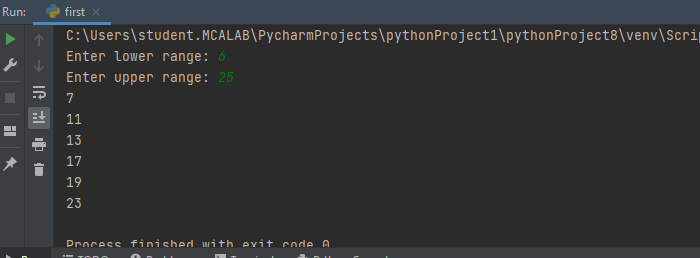
if (num % i) == 0:

break

else:

print(num)

**OUTPUT**



2. Program to print the first N Fibonacci numbers.

**CODE**

n = int (input("Enter the number of terms needed in the Fibonacci series: "))

if (n<0):

print ("Enter a positive number")

else:

f1, f2 = 0, 1

if n == 1:

print (f1)

elif n == 2:

print (f1,f2)

else:

print (f1,f2, end = ' ')

for i in range (3, n+1):

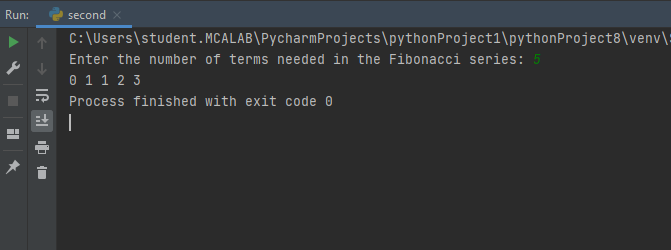
f3 = f1 + f2

print (f3, end = ' ')

f1 = f2

f2 = f3

**OUTPUT**



3. Given sides of a triangle, write a program to check whether given triangle is an isosceles, equilateral or scalene.

**CODE**

print("Input lengths of the triangle sides: ")

x = int(input("x: ")

y = int(input("y: "))

z = int(input("z: "))

if x == y == z:

print("Equilateral triangle")

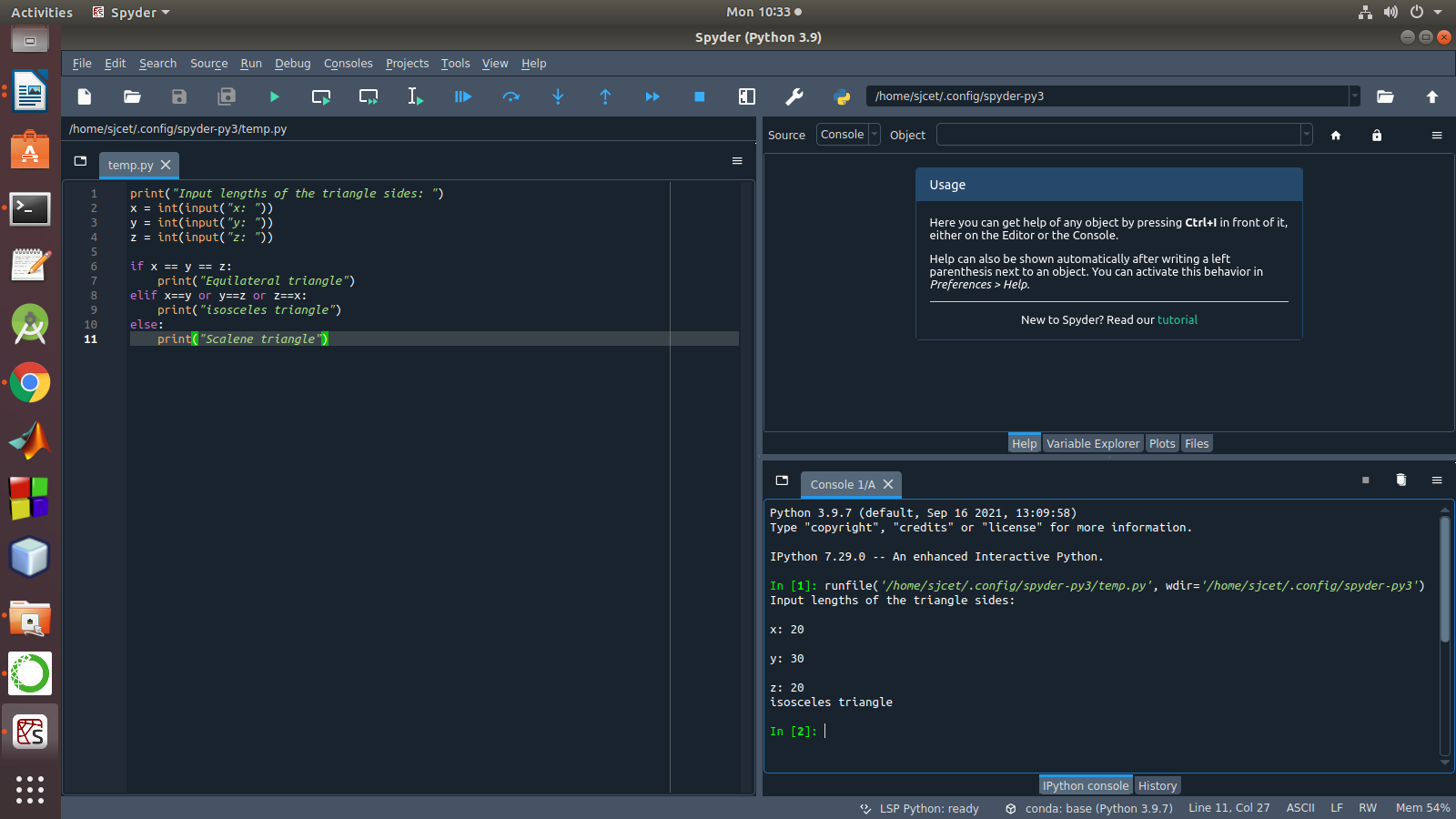
elif x==y or y==z or z==x:

print("isosceles triangle")

else:

print("Scalene triangle")

**OUTPUT**

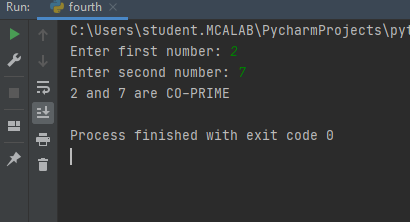


4. Program to check whether given pair of number is coprime

**CODE**

def are\_coprime(a,b):  
  
 hcf = 1  
  
 for i in range(1, a+1):  
 if a%i==0 and b%i==0:  
 hcf = i  
  
 return hcf == 1  
  
# Reading two numbers  
first = int(input('Enter first number: '))  
second = int(input('Enter second number: '))  
  
if are\_coprime(first, second):  
 print('%d and %d are CO-PRIME' %(first, second))  
else:  
 print('%d and %d are NOT CO-PRIME' %(first, second))

**OUTPUT**



5. Program to find the roots of a quadratic equation(rounded to 2 decimal places)

**CODE**

from math import sqrt\*x + c")

a = float(input("a: "))

b = float(input("b: "))

c = float(input("c: "))

r = b\*\*2 - 4\*a\*c

if r > 0:

num\_roots = 2

x1 = (((-b) + sqrt(r))/(2\*a))

x2 = (((-b) - sqrt(r))/(2\*a))

print("There are 2 roots: %f and %f" % (x1, x2))

elif r == 0:

num\_roots = 1

x = (-b) / 2\*a

print("There is one root: ", x)

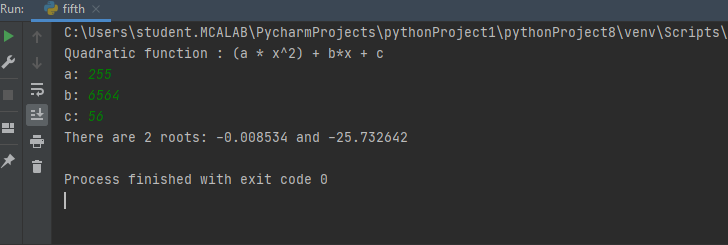
else:

num\_roots = 0

print("No roots, discriminant < 0.")

exit()

**OUTPUT**



6. Program to check whether a given number is perfect number or not(sum of factors=number

**CODE**

n = int(input("Enter any number: "))

sum1 = 0

for i in range(1, n):

if(n % i == 0):

sum1 = sum1 + i

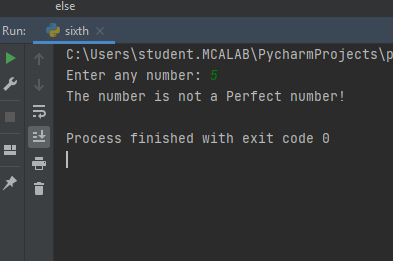
if (sum1 == n):

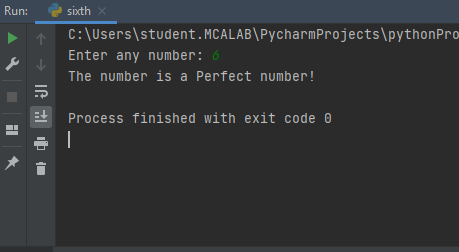
print("The number is a Perfect number!")

else:

print("The number is not a Perfect number!")

**OUTPUT**





7. Program to display amstrong numbers upto 1000

**CODE**

lower = int(input("Enter lower range: "))

upper = int(input("Enter upper range: "))

for num in range(lower,upper + 1):

sum = 0

temp = num

while temp > 0:

digit = temp % 10

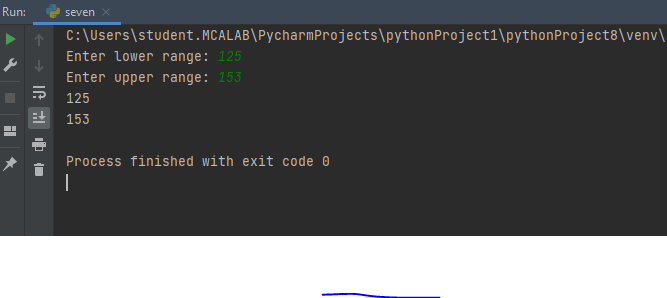
sum += digit \*\* 3

temp //= 10

if num == sum:

print(num)

**OUTPUT**



8. Store and display the days of a week as a List, Tuple, Dictionary, Set. Also demonstrate different ways to store values in each of them. Display its type also.

**CODE**

list = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"]

print(type(list))

print(list)

tuple = ("Sun","Mon","Tue","Wed","Thu","Fri","Sat")

print(type(tuple))

print(tuple)

set = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}

print(type(set))

print(set)

dict = {

"d1" : "Sun",

"d2" : "Mon",

"d3" : "Tue",

"d4" : "Wed",

"d5" : "Thu",

"d6" : "Fri",

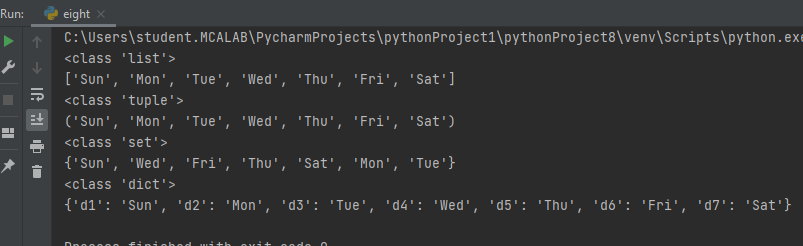
"d7" : "Sat"

}

print(type(dict))

print(dict)

**OUTPUT**



9. Write a program to add elements of given 2 lists.

**CODE**

lt1 = [5, 10, 15, 20, 25, 30]

lt2 = [2, 4, 6, 8, 10, 12]

# print the original list element

print(" Python Original list 1: " + str(lt1))

print("Python Original list 2: " + str(lt2))

# use naive method to add two list.

res\_lt = [] # declaration of the list

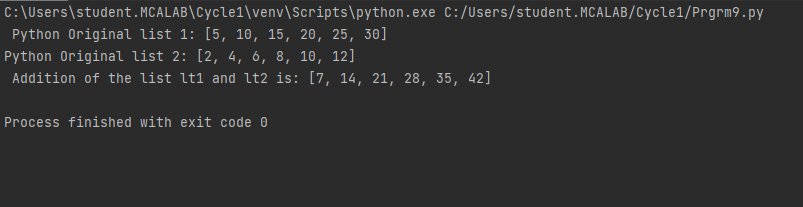
for x in range(0, len(lt1)):

res\_lt.append(lt1[x] + lt2[x])

# Display the sum of two list in Python

print(" Addition of the list lt1 and lt2 is: " + str(res\_lt))

**OUTPUT**



10. Write a program to find the sum of 2 matrices using nested List.

**CODE**

matOne = []

print("Enter Elements for First Matrix: ")

for i in range(3):

matOne.append([])

for j in range(3):

num = int(input())

matOne[i].append(num)

matTwo = []

print("Enter Elements for Second Matrix: ")

for i in range(3):

matTwo.append([])

for j in range(3):

num = int(input())

matTwo[i].append(num)

matThree = []

for i in range(3):

matThree.append([])

for j in range(3):

matThree[i].append(matOne[i][j]+matTwo[i][j])

print("\nAddition Result of Two Given Matrix is:")

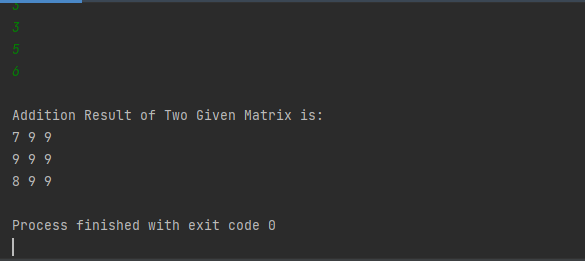
for i in range(3):

for j in range(3):

print(matThree[i][j], end=" ")

print()

**OUTPUT**



11. Write a program to perform bubble sort on a given set of elements.

**CODE**

def bubble\_sort(list1):

# Outer loop for traverse the entire list

for i in range(0, len(list1) - 1):

for j in range(len(list1) - 1):

if (list1[j] > list1[j + 1]):

temp = list1[j]

list1[j] = list1[j + 1]

list1[j + 1] = temp

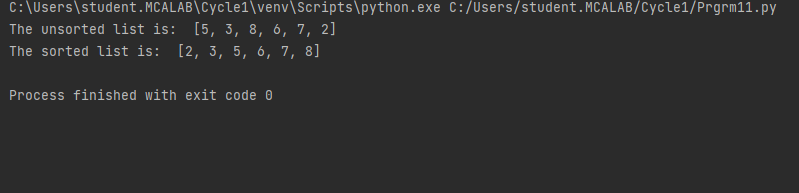
return list1

list1 = [5, 3, 8, 6, 7, 2]

print("The unsorted list is: ", list1)

# Calling the bubble sort function

print("The sorted list is: ", bubble\_sort(list1))

**OUTPUT**

12. Program to find the count of each vowel in a string(use dictionary).

**CODE**

def Check\_Vow(string, vowels):

# The term "casefold" has been used to refer to a method of ignoring cases.

string = string.casefold()

count = {}.fromkeys(vowels, 0)

# To count the vowels

for character in string:

if character in count:

count[character] += 1

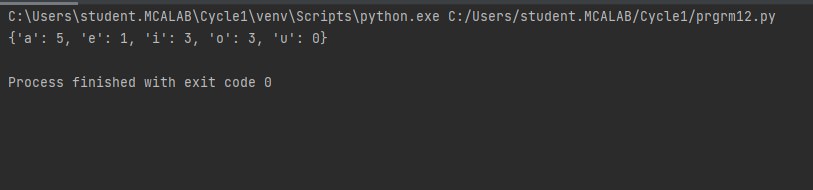
return count

vowels = 'aeiou'

string = "Hi, myself lisna jos and i am from kottayam,"

print (Check\_Vow(string, vowels))

**OUTPUT**



13. Write a Python program that accept a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive.

**CODE**

def repeat\_times(n):

s = 0

n\_str = str(n)

while (n > 0):

n -= sum([int(i) for i in list(n\_str)])

n\_str = list(str(n))

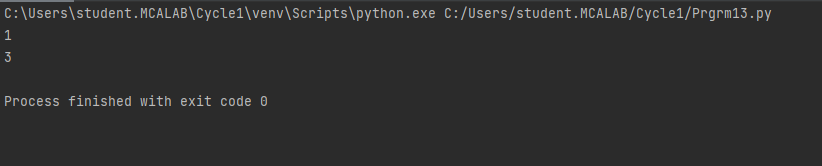
s += 1

return s

print(repeat\_times(9))

print(repeat\_times(21))

**OUTPUT**



14. Write a Python program that accepts a 10 digit mobile number, and find the digits which are absent in a given mobile number.

**CODE**

def absent\_digits(n):

all\_nums = set([0,1,2,3,4,5,6,7,8,9])

n = set([int(i) for i in n])

n = n.symmetric\_difference(all\_nums)

n = sorted(n)

return n

print(absent\_digits([9,5,2,6,0,1,4,6,8,4]))

**OUTPUT**



**CYCLE-2**

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

**CODE**

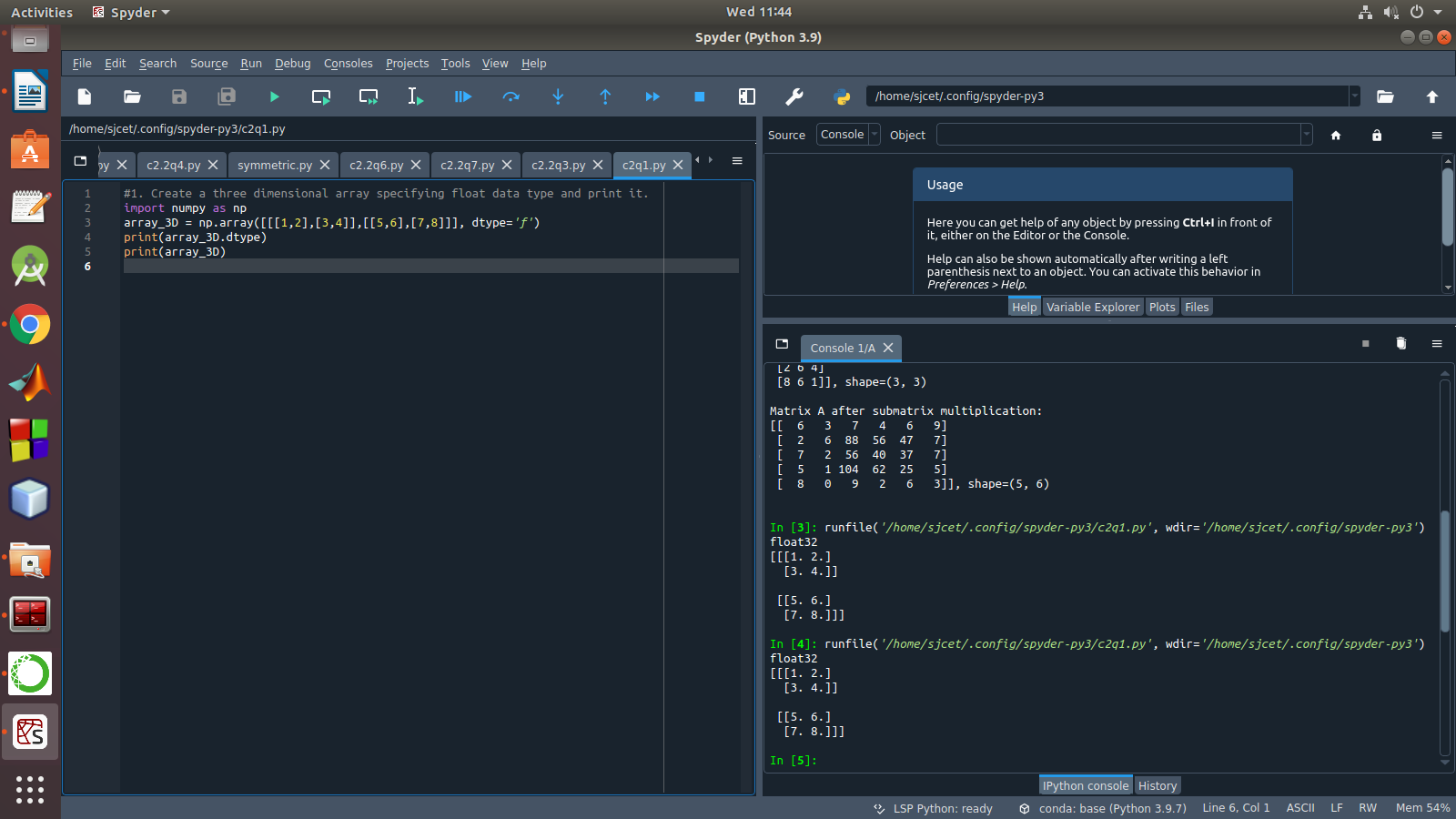
import numpy as np

array\_3D = np.array([[[1,2],[3,4]],[[5,6],[7,8]]], dtype='f')

print(array\_3D.dtype)

print(array\_3D)

**OUTPUT**

****

2. Create a 2 dimensional array (2X3) with elements belonging to complex data

type and print it. Also display

a. the no: of rows and columns

b. dimension of an array

c. reshape the same array to 3X2

**CODE**

import numpy as np

array\_2d=np.array([[complex(1,2),complex(2,3),complex(3,4)],[complex(4,5),complex(5,6),complex(6,7)]])

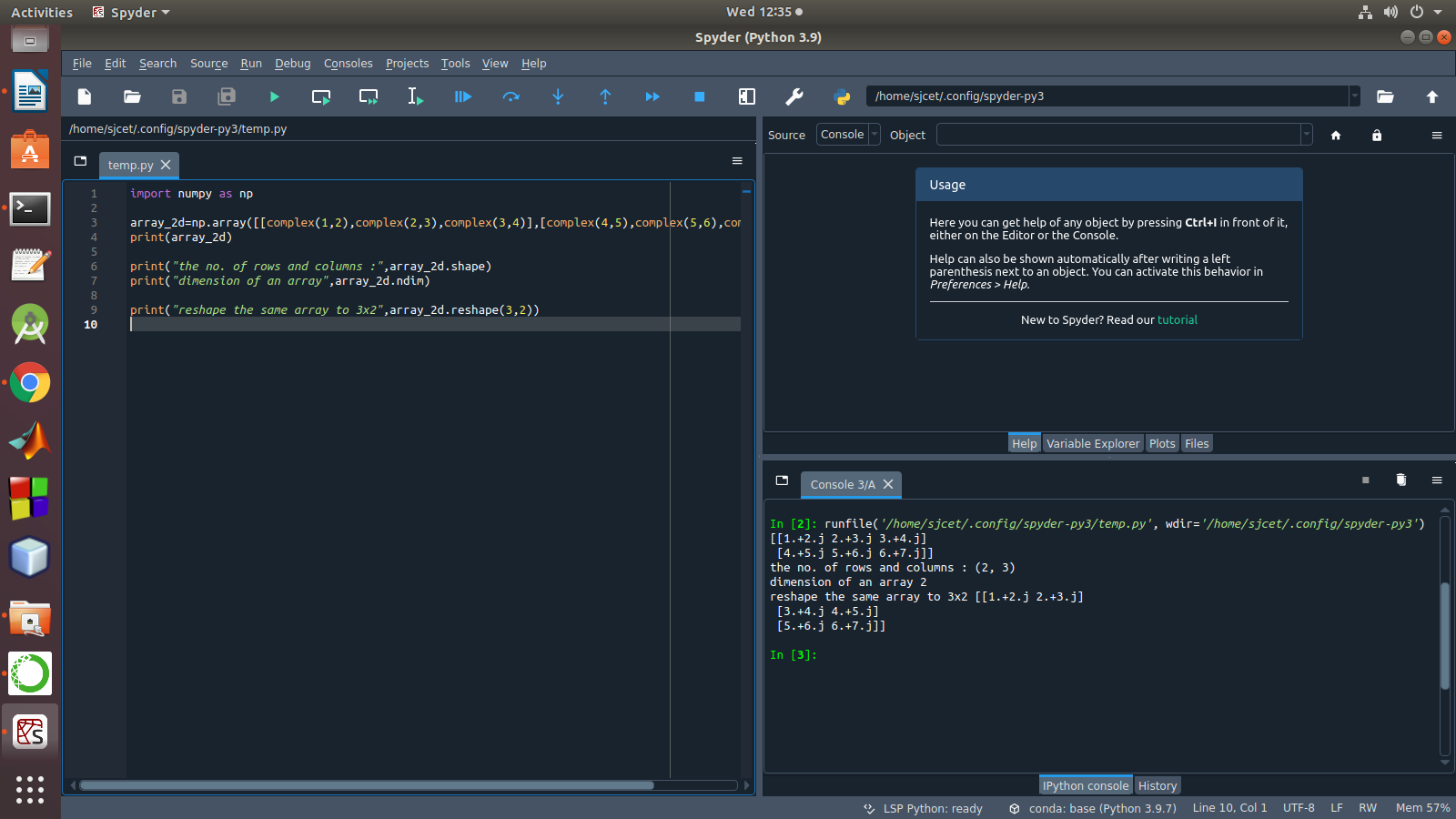
print(array\_2d)

print("the no. of rows and columns :",array\_2d.shape)

print("dimension of an array",array\_2d.ndim)

print("reshape the same array to 3x2",array\_2d.reshape(3,2))

**OUTPUT**

****

3. Familiarize with the functions to create

a) an uninitialized array

b) array with all elements as 1,

c) all elements as 0

**CODE**

import numpy as np

x=np.empty([2, 2])

print(x)

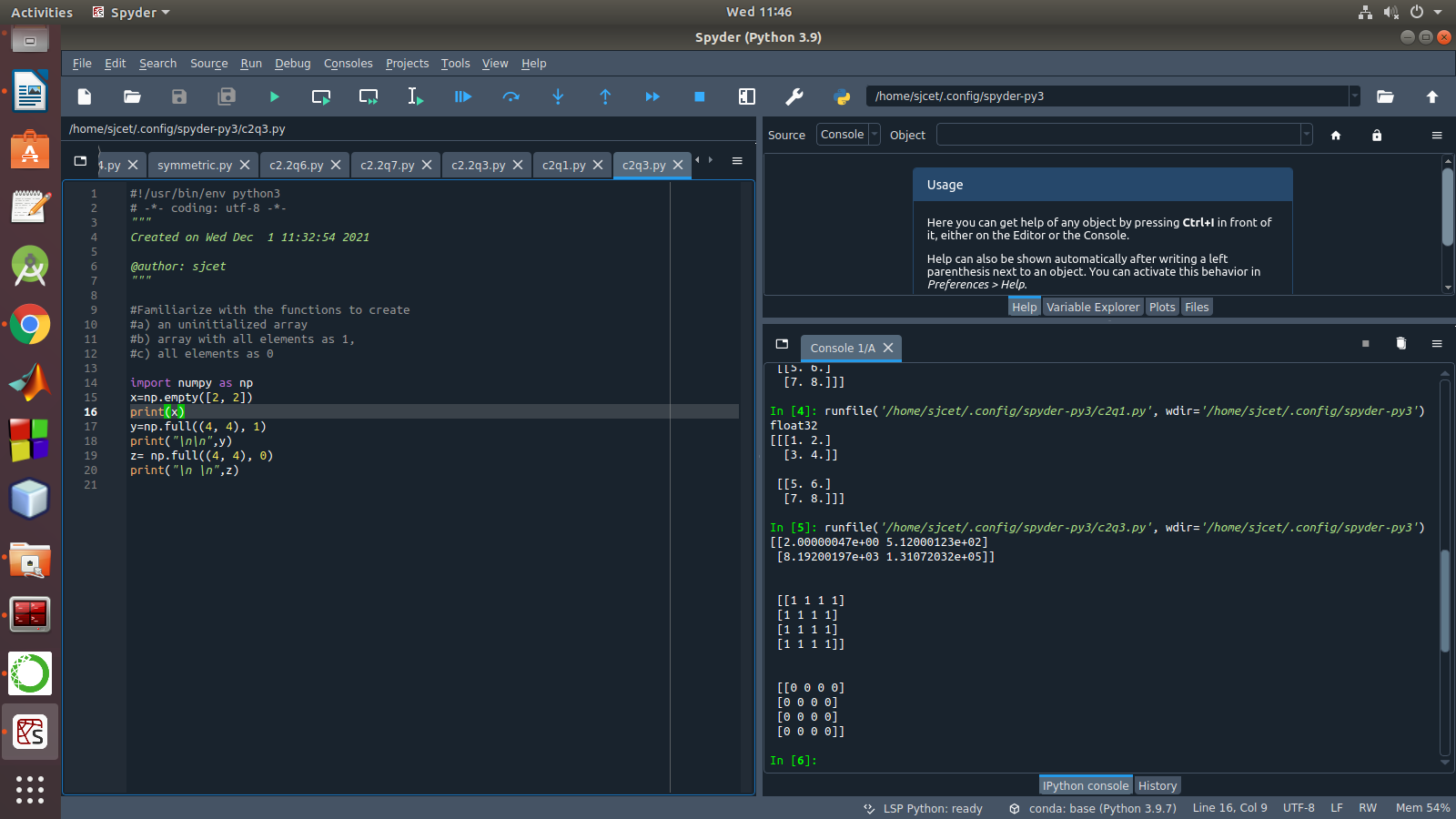
y=np.full((4, 4), 1)

print("\n\n",y)

z= np.full((4, 4), 0)

print("\n \n",z)

**OUTPUT**



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

Display

a. First 4 elements

b. Last 6 elements

c. Elements from index 2 to 7

**CODE**

import numpy as np

arr = np.arange(1, 11)

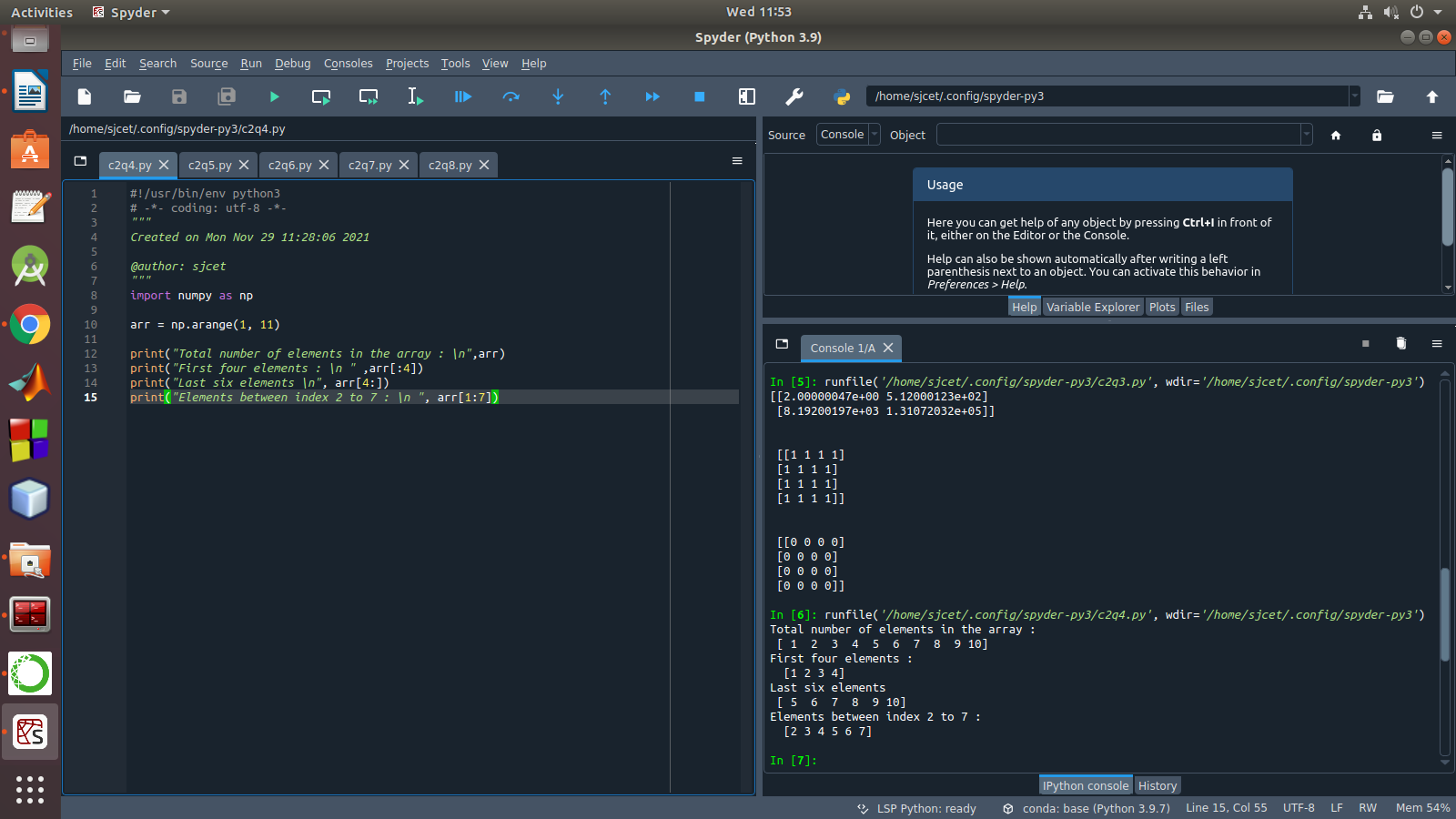
print("Total number of elements in the array : \n",arr)

print("First four elements : \n " ,arr[:4])

print("Last six elements \n", arr[4:])

print("Elements between index 2 to 7 : \n ", arr[1:7])

**OUTPUT**

****

5. Create an 1D array with arange containing first 15 even numbers as elements

a. Elements from index 2 to 8 with step 2(also demonstrate the same

using slice function)

b. Last 3 elements of the array using negative index

c. Alternate elements of the array

d. Display the last 3 alternate elements

**CODE**

import numpy as np

arr = np.arange(0,31,2)

print("First 15 even numbers are : \n", arr)

print ( "Elements from index 2 to 8 are : \n ", arr[2:8])

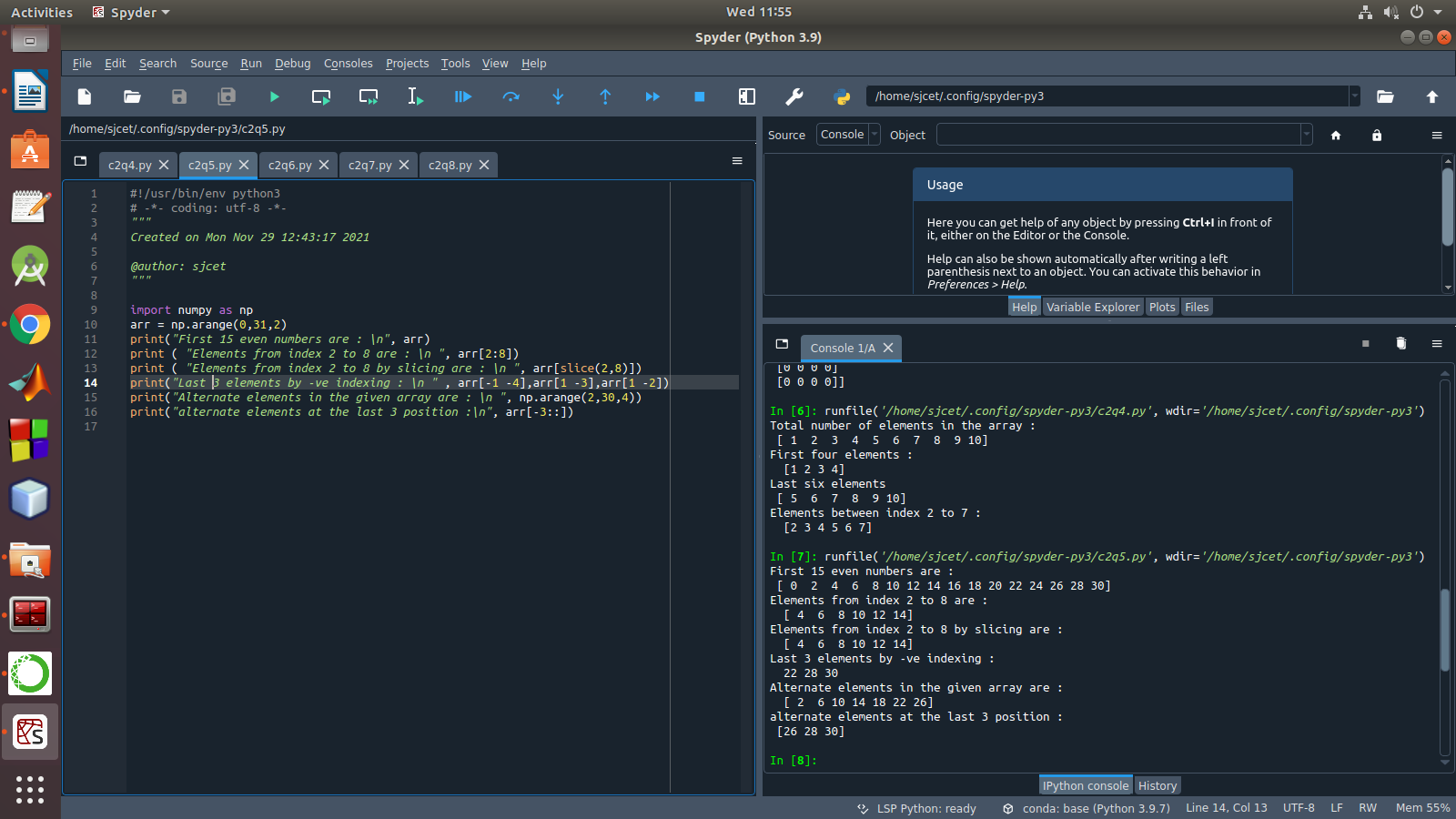
print ( "Elements from index 2 to 8 by slicing are : \n ", arr[slice(2,8)])

print("Last 3 elements by -ve indexing : \n " , arr[-1 -4],arr[1 -3],arr[1 -2])

print("Alternate elements in the given array are : \n ", np.arange(2,30,4))

print("alternate elements at the last 3 position :\n", arr[-3::])

**OUTPUT**

****

6. Create a 2 Dimensional array with 4 rows and 4 columns.

a. Display all elements excluding the first row

b. Display all elements excluding the last column

c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row

d. Display the elements of 2 nd and 3 rd column

e. Display 2 nd and 3 rd element of 1 st row

f. Display the elements from indices 4 to 10 in descending order(use

–values)

**CODE**

import numpy as np

array\_2d=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])

print(array\_2d)

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

print(array\_2d[1:4,:])

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

print(array\_2d[:,0:3])

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

print(array\_2d[1:3,1:3])

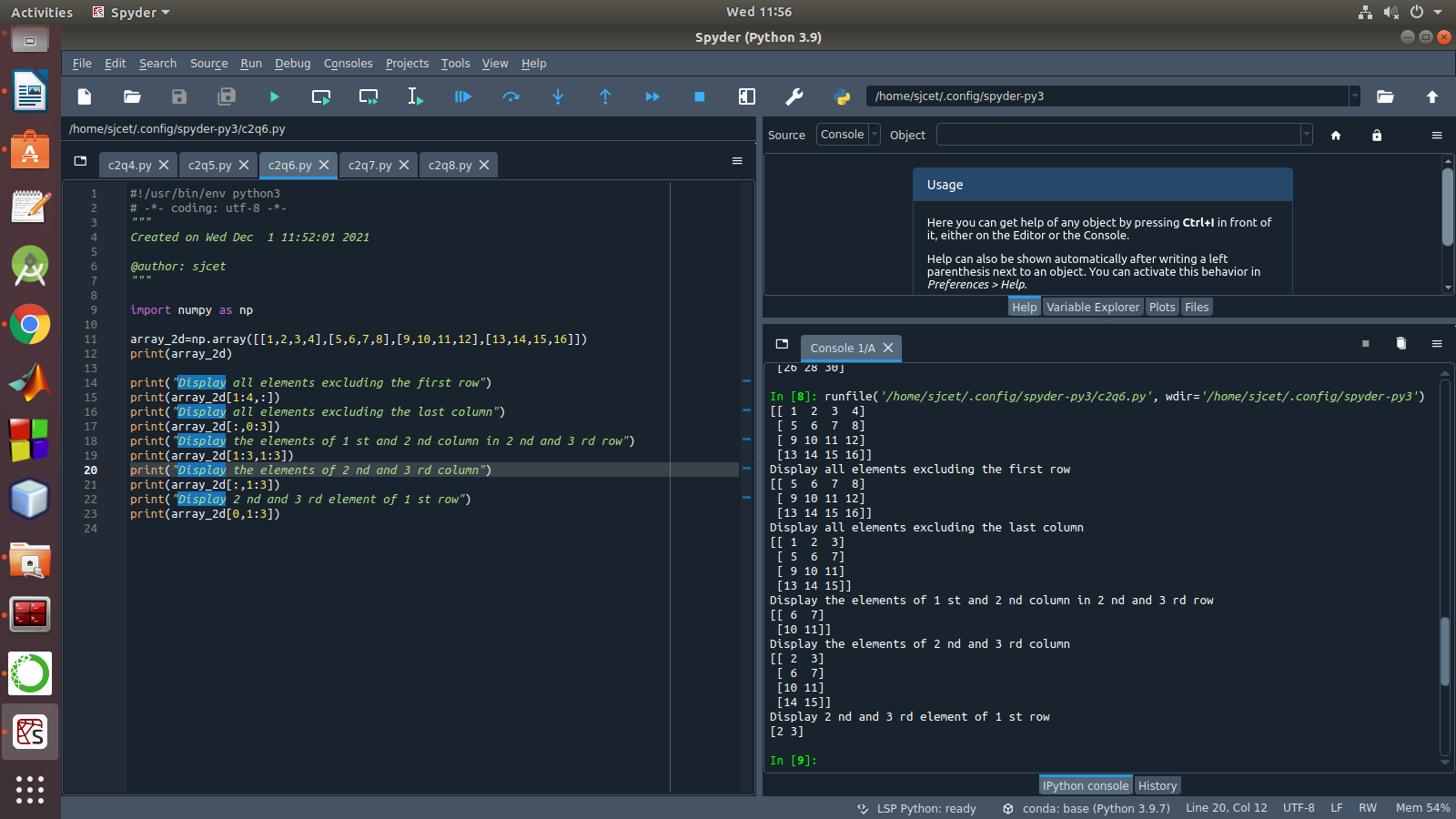
print("Display the elements of 2 nd and 3 rd column")

print(array\_2d[:,1:3])

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

print(array\_2d[0,1:3])

**OUTPUT**

****

7. Create two 2D arrays using array object and

a. Add the 2 matrices and print it

b. Subtract 2 matrices

c. Multiply the individual elements of matrix

d. Divide the elements of the matrices

e. Perform matrix multiplication

f. Display transpose of the matrix

g. Sum of diagonal elements of a matrix

**CODE**

import numpy as np

M1 = np.array([[3, 6], [14, 21]])

M2 = np.array([[9, 27], [11, 22]])

M3 = M1 + M2

print("Matrix addition")

print(M3)

M1 = np.array([[3, 6], [14, 21]])

M2 = np.array([[9, 27], [11, 22]])

M3 = M1 - M2

print("Matrix Substract")

print(M3)

M1 = np.array([[3, 6], [14, 21]])

M2 = np.array([[9, 27], [11, 22]])

M3 = M1 / M2

print("Divide the elements of the matrices")

print(M3)

M1 = np.array([[3, 6], [5, -10]])

M2 = np.array([[9, -18], [11, 22]])

M3 = M1 \* M2

print("Multiply the individual elements of matrix")

print(M3)

M1 = np.array([[3, 6], [5, -10]])

M2 = np.array([[9, -18], [11, 22]])

M3 = M1.dot(M2)

print("matrix multiplication")

print(M3)

M1 = np.array([[3, 6, 9], [5, -10, 15], [4,8,12]])

M2 = M1.transpose()

print("Transpose of the matrix")

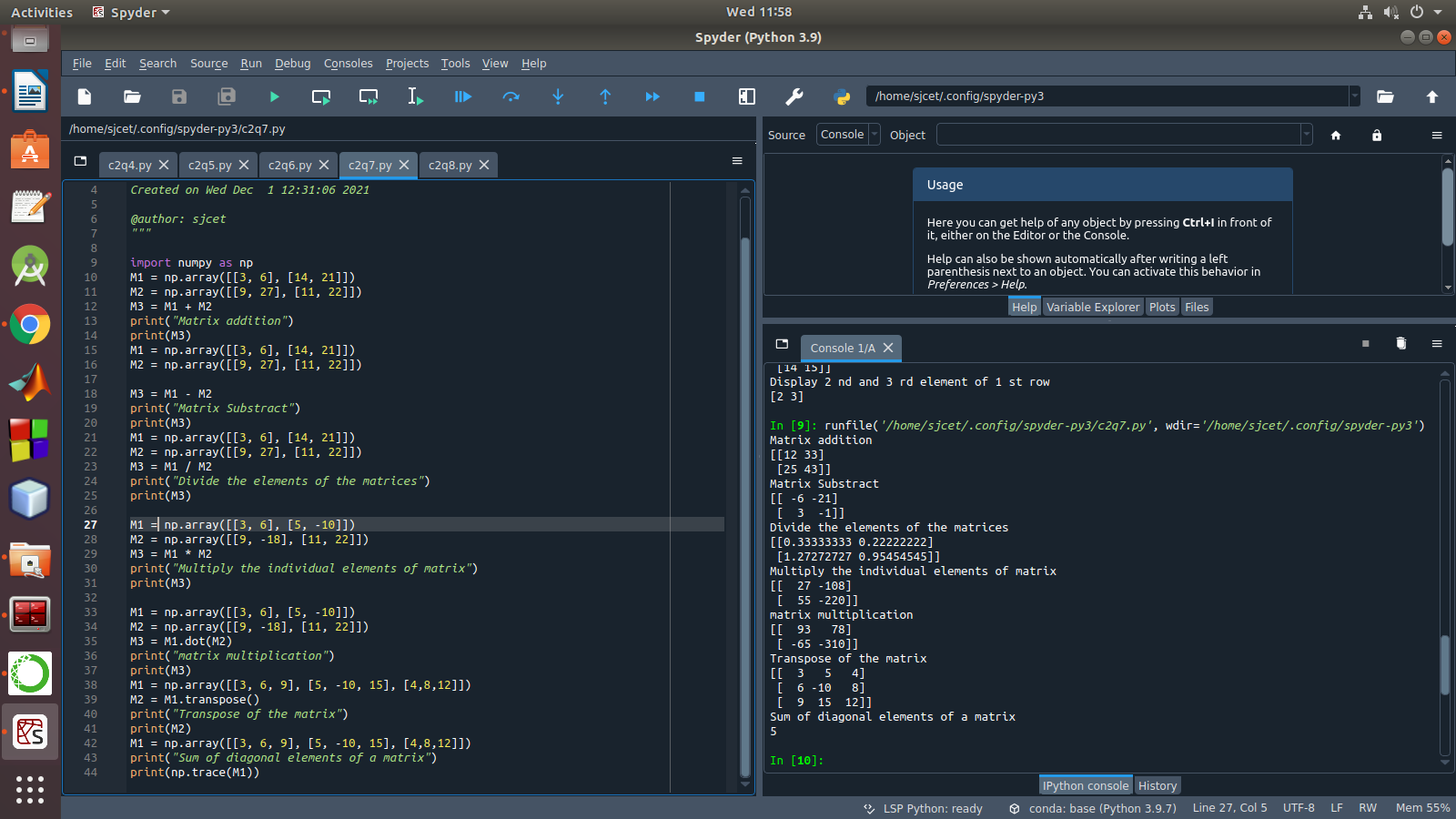
print(M2)

M1 = np.array([[3, 6, 9], [5, -10, 15], [4,8,12]])

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

print(np.trace(M1))

**OUTPUT**

****

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

**CODE**

list = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"]

print(type(list))

print(list)

tuple = ("Sun","Mon","Tue","Wed","Thu","Fri","Sat")

print(type(tuple))

print(tuple)

set = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}

print(type(set))

print(set)

dict = {

"d1" : "Sun",

"d2" : "Mon",

"d3" : "Tue",

"d4" : "Wed",

"d5" : "Thu",

"d6" : "Fri",

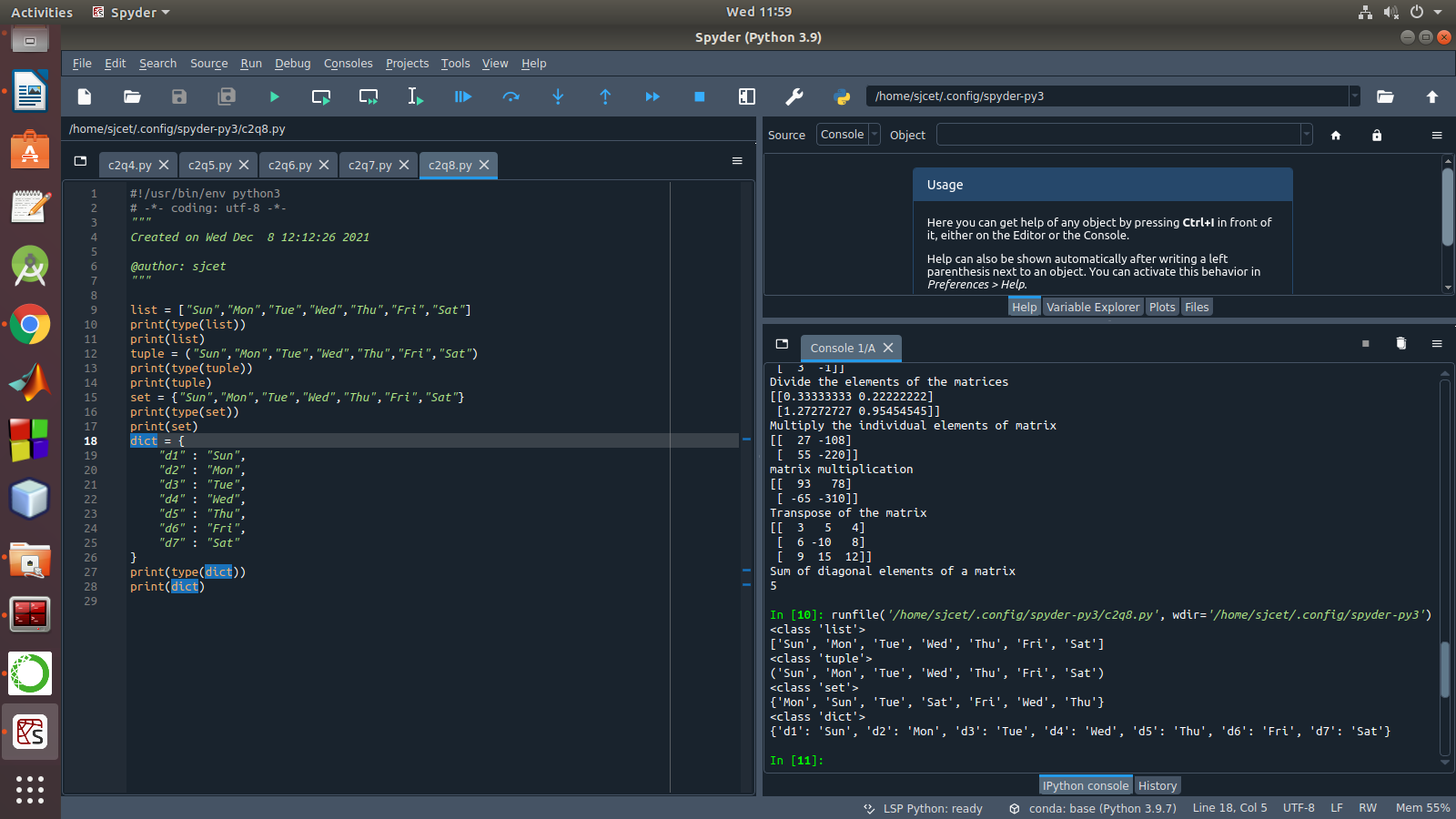
"d7" : "Sat"

}

print(type(dict))

print(dict)

**OUTPUT**

****

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

**CODE**

# initialize the Python lists

lt1 = [5, 10, 15, 20, 25, 30]

lt2 = [2, 4, 6, 8, 10, 12]

# print the original list element

print ( " Python Original list 1: " + str (lt1))

print ( "Python Original list 2: " + str (lt2))

# use naive method to add two list.

res\_lt = [] # declaration of the list

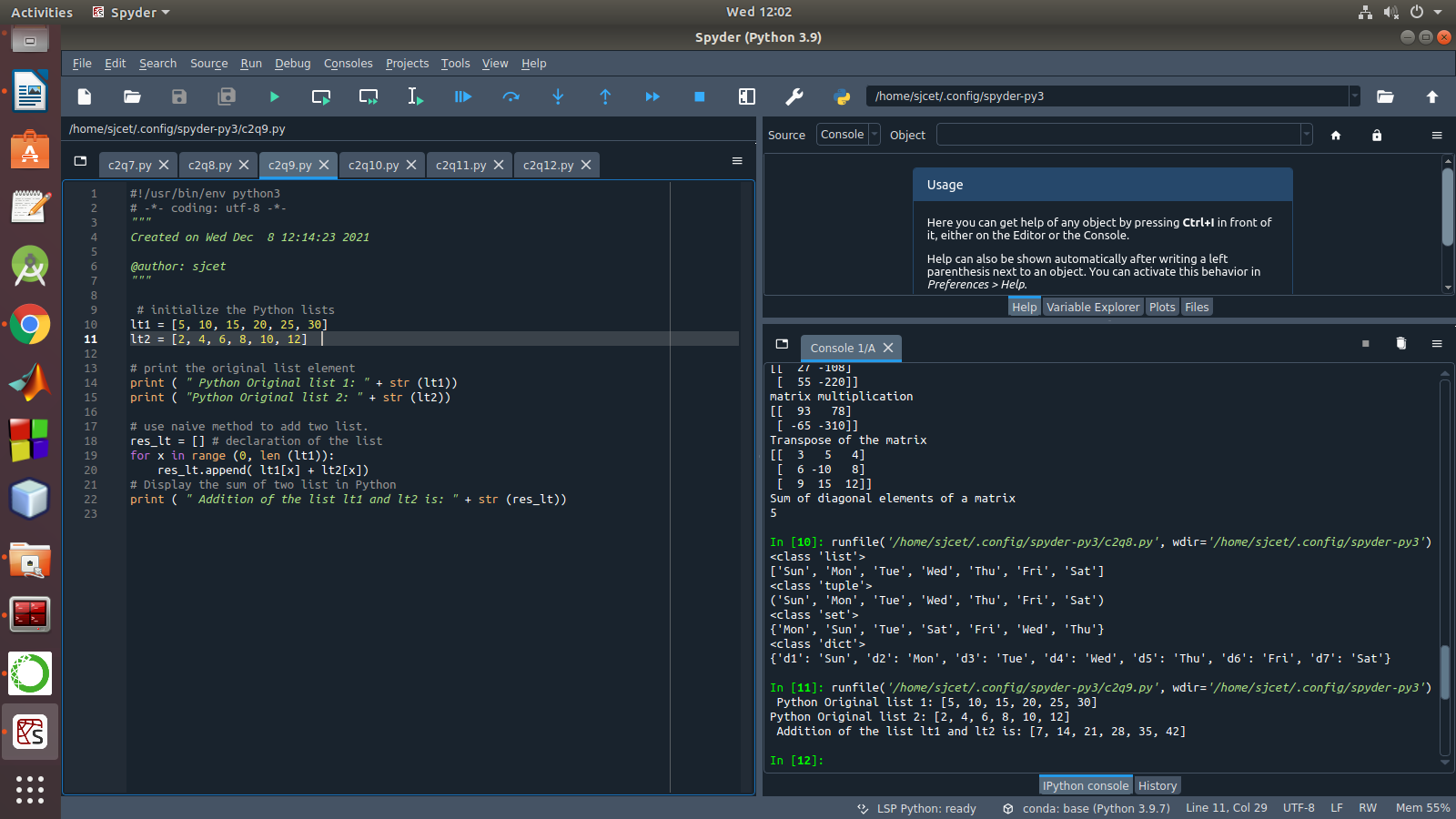
for x in range (0, len (lt1)):

res\_lt.append( lt1[x] + lt2[x])

# Display the sum of two list in Python

print ( " Addition of the list lt1 and lt2 is: " + str (res\_lt))

**OUTPUT**

****

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

**CODE**

matOne = []

print("Enter Elements for First Matrix: ")

for i in range(3):

matOne.append([])

for j in range(3):

num = int(input())

matOne[i].append(num)

matTwo = []

print("Enter Elements for Second Matrix: ")

for i in range(3):

matTwo.append([])

for j in range(3):

num = int(input())

matTwo[i].append(num)

matThree = []

for i in range(3)

matThree.append([])

for j in range(3):

matThree[i].append(matOne[i][j]+matTwo[i][j])

print("\nAddition Result of Two Given Matrix is:")

for i in range(3):

for j in range(3):

print(matThree[i][j], end=" ")

print()

**OUTPUT**

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

**CODE**

**OUTPUT**

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

**CODE**

import numpy as np

array=np.random.randint(5,size=(2,2));

print("random array is \n ", array)

inverse=np.linalg.inv(array)

print("Inverse of the given matrix is \n", inverse)

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

print("Rank of the matrix is \n", rank)

determinant=np.linalg.det(array)

print("Determinant of the given matrix is \n", determinant)

oned=array.flatten()

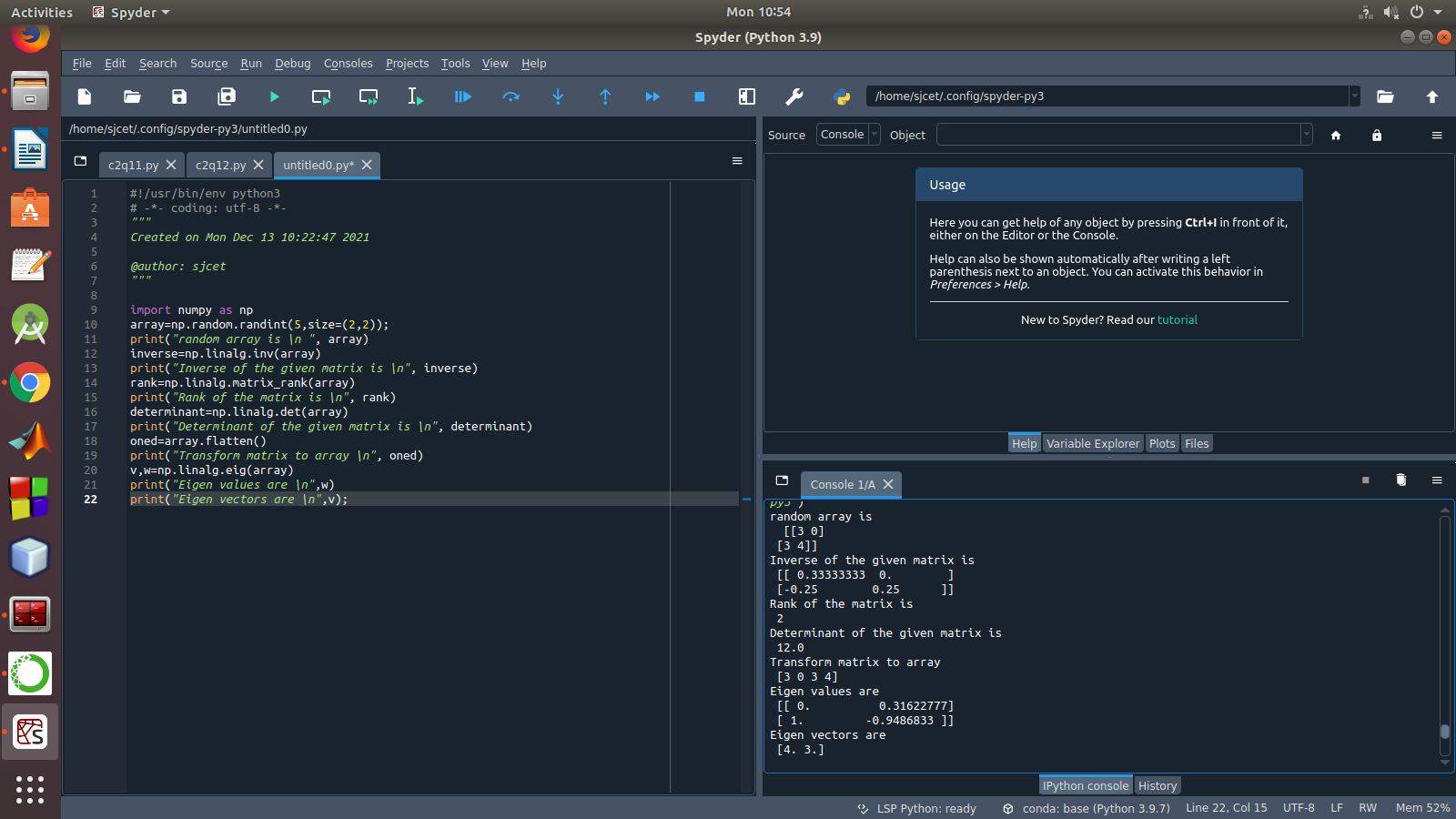
print("Transform matrix to array \n", oned)

v,w=np.linalg.eig(array)

print("Eigen values are \n",w)

print("Eigen vectors are \n",v);

**OUTPUT**



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 X 2 +2

**CODE**

import numpy as np

x=np.array([[4,5,8],

[7,6,2],

[1,2,5]])

print(x)

cube=np.power(x,3)

print("cube of the given matrix using power() \n",cube)

cube=np.multiply(x,(x\*x))

print("cube using multiply fn is \n", cube)

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

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

out=np.power(x,x)

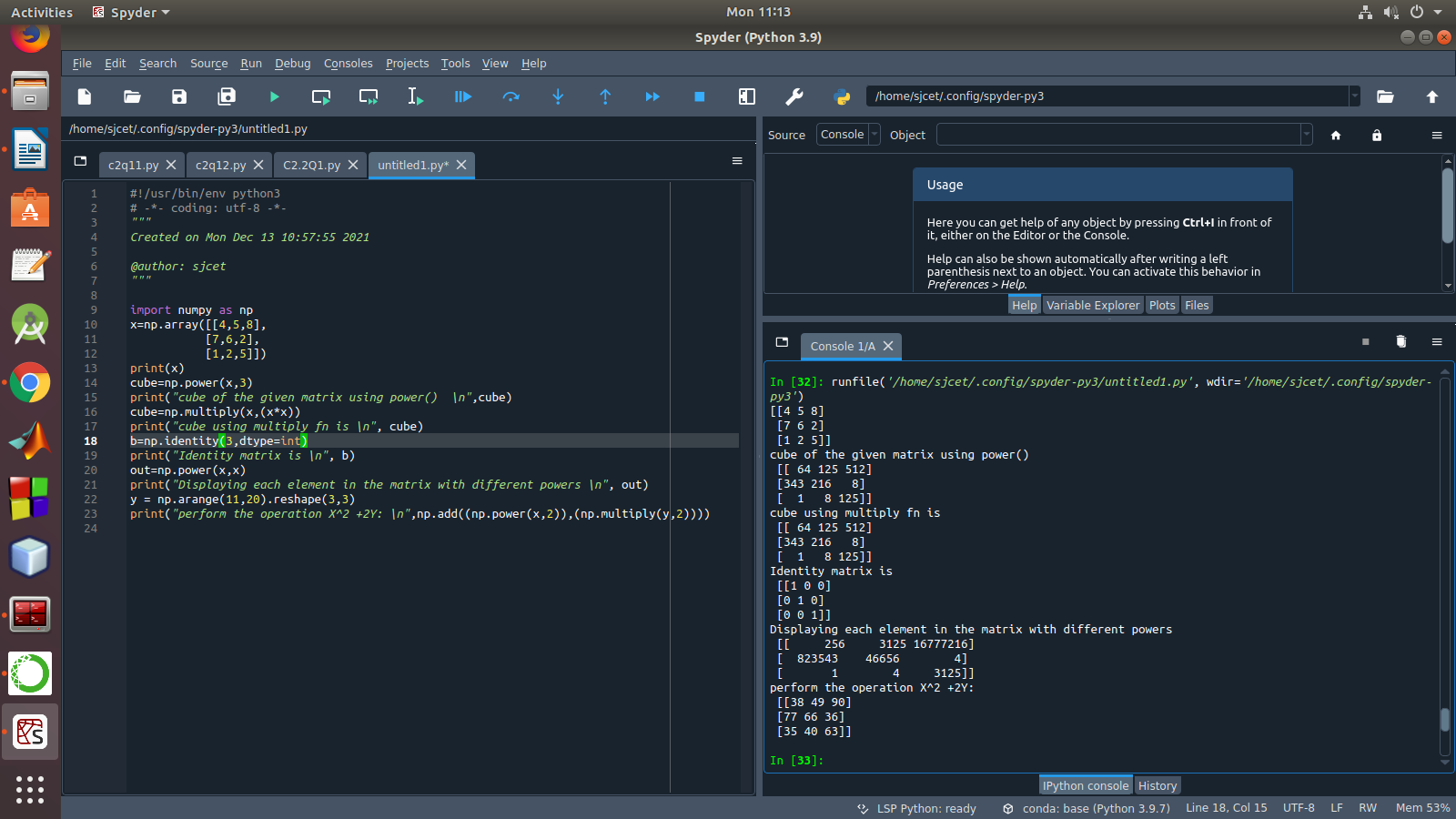
print("Displaying each element in the matrix with different powers \n", out)

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

print("perform the operation X^2 +2Y: \

n",np.add((np.power(x,2)),(np.multiply(y,2))))

**OUTPUT**



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

matrix.

**CODE**

import numpy as np

np.random.seed(42)

A = np.random.randint(0, 10, size=(5,6))

B = np.random.randint(0, 10, size=(3,3))

print("Matrix A:\n{}, shape={}\n".format(A, A.shape))

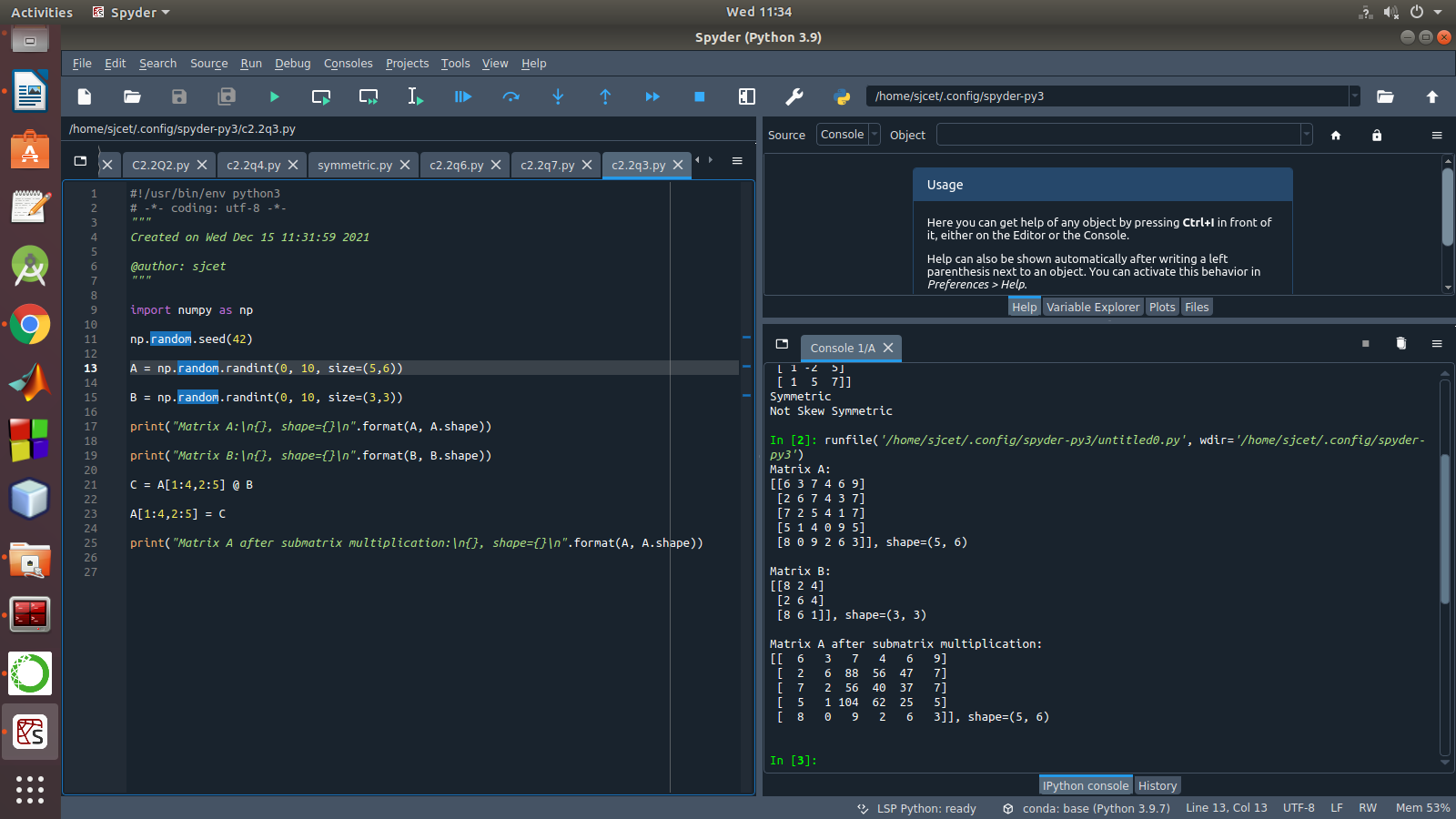
print("Matrix B:\n{}, shape={}\n".format(B, B.shape))

C = A[1:4,2:5] @ B

A[1:4,2:5] = C

print("Matrix A after submatrix multiplication:\n{}, shape={}\n".format(A, A.shape))

**OUTPUT**



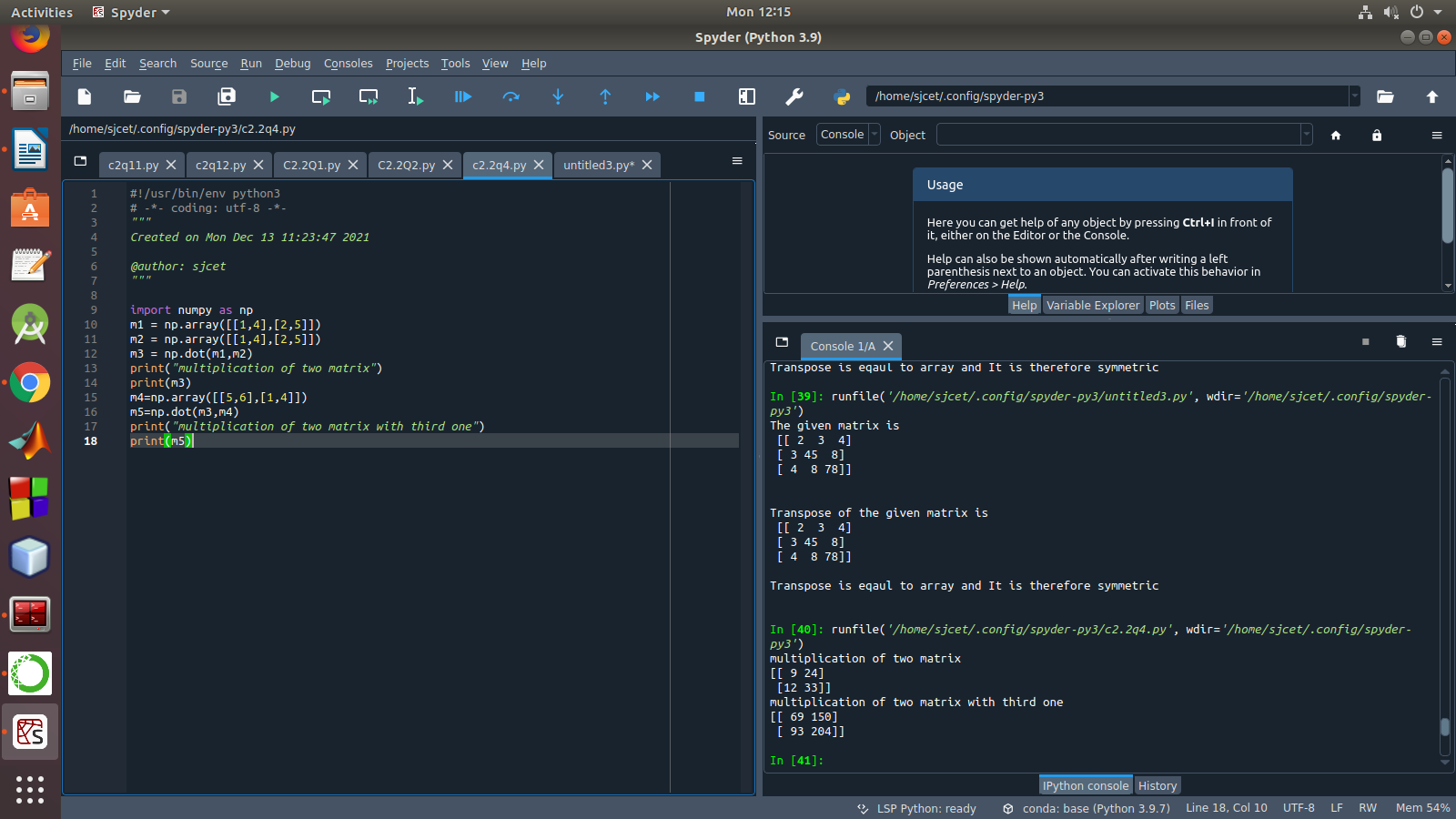
4. Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of

the 3 matrices.

**CODE**

mport numpy as np  
m1 = np.array([[1,4],[2,5]])  
m2 = np.array([[1,4],[2,5]])  
m3 = np.dot(m1,m2)   
print("multiplication of two matrix")  
print(m3)  
m4=np.array([[5,6],[1,4]])  
m5=np.dot(m3,m4)  
print("multiplication of two matrix with third one")  
print(m5)

**OUTPUT**



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

CODE

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")

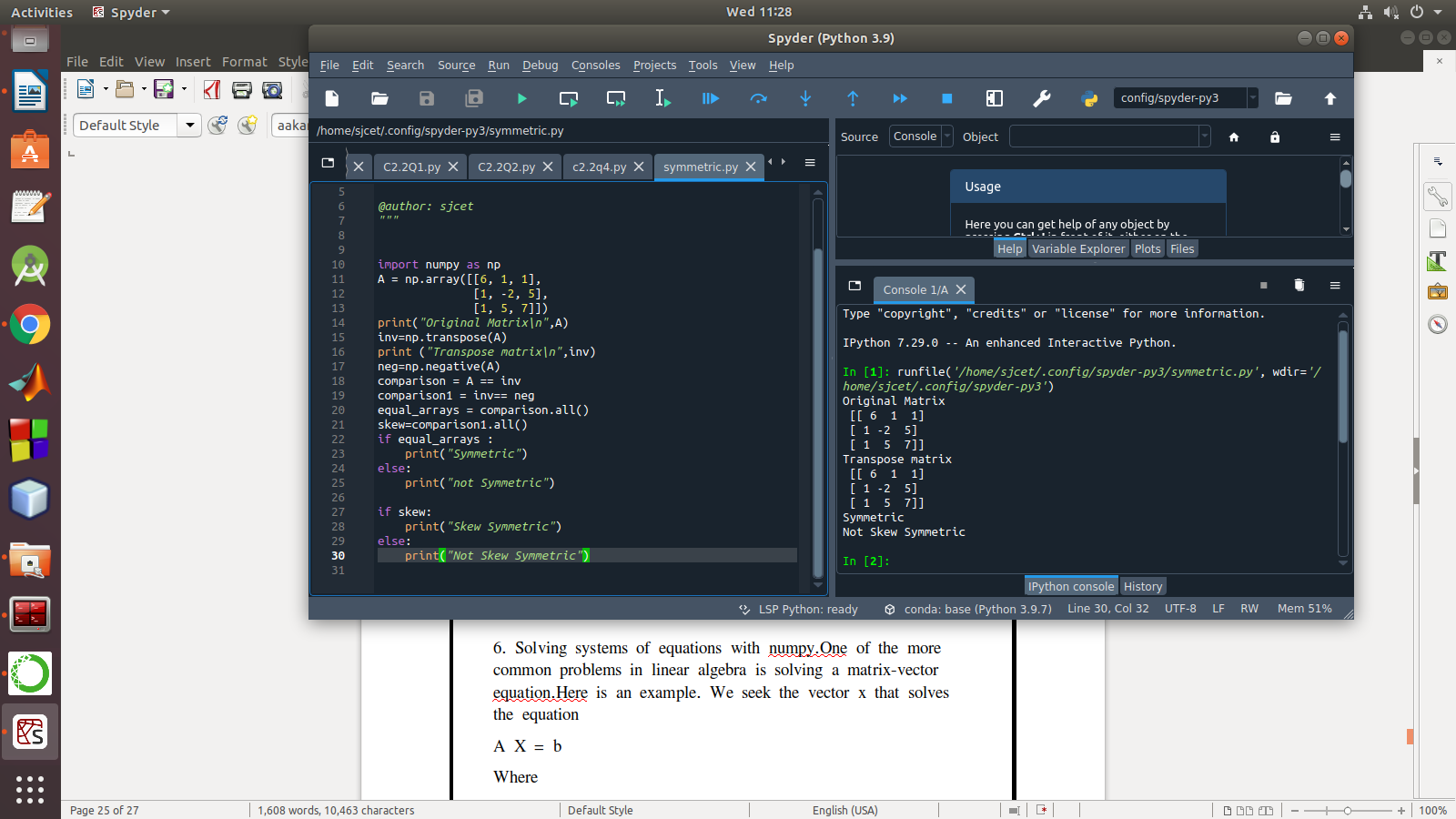
if skew:

print("Skew Symmetric")

else:

print("Not Skew Symmetric")

**OUTPUT**



6. 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.

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

**CODE**

import numpy as np

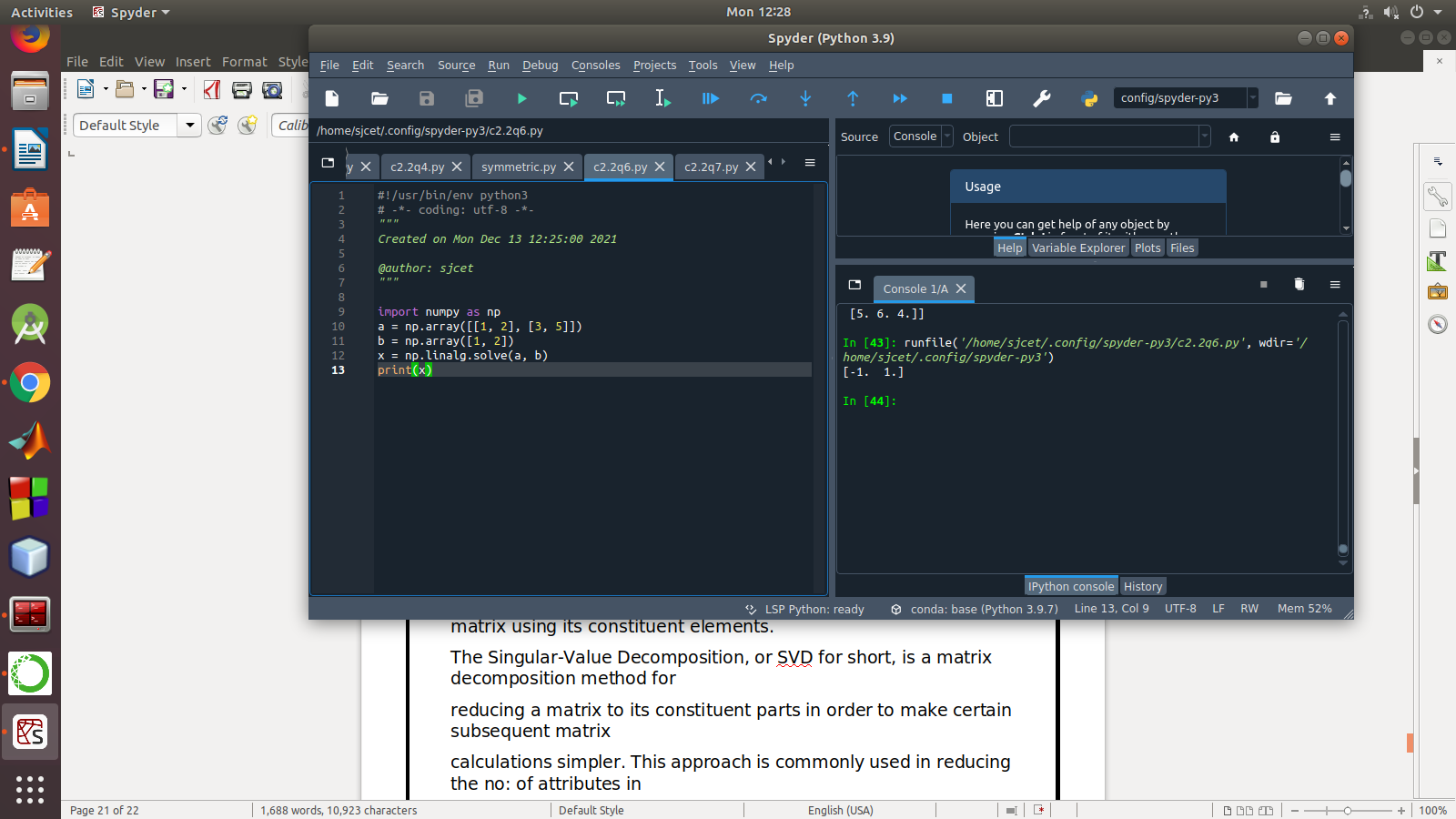
a = np.array([[1, 2], [3, 5]])

b = np.array([1, 2])

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

print(x)

**OUTPUT**



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.

**CODE**

# Singular-value decomposition

from numpy import array

from scipy.linalg import svd

from numpy import dot

from numpy import diag

# define a matrix

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

print(A)

# SVD

U, s, VT = svd(A)

print(U)

print(s)

print(VT)

Sigma=diag(s)

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

print(B)

**OUTPUT**

