Maths Lab Assignents

|  |
| --- |
| Set 1  import numpy as np  list1 = [1, 2, 3] list2 = [1, 2, 3]  #defining vectors vector1 = np.array(list1) print("First Vector: "+str(vector1)) vector2 = np.array(list2) print("Second vector: " + str(vector2)) print("\n")  #adding both vectors addition = vector1 + vector2 print("Vector Addition: " + str(addition))  #subtracting both the vector subtraction = vector1 + vector2 print("Vector Subtraction: "+ str(subtraction))  #multiplying both the vector multiplication= vector1 \* vector2 print("Vector Multiplication: "+str(multiplication))  #dividing both the vector division = vector1 / vector2 print("Vector Division: "+ str(division))  #finding dot product dot\_product = vector1.dot(vector2) print("Dot Product: "+str(dot\_product))  #vector scalar multiplication vector=np.array([1,2]) scalar=np.array([3]) vsm=np.multiply(vector,scalar) print("Vector scalar multiplication: "+str(vsm))  **Output**  First Vector: [1 2 3]  Second vector: [1 2 3]  Vector Addition: [ 2 4 6]  Vector Subtraction: [ 0 0 0]  Vector Multiplication: [ 1 4 9]  Vector Division: [1. 1. 1.]  Dot Product:14  Vector scalar multiplication: [3 6] |

|  |
| --- |
| Set 2  #Calculate L1,L2,max norms of a vector import numpy as np  from numpy.linalg import norm arr = np.array([1, 2, 3, 4, 5]) print(arr)  norm\_l1 = norm(arr,1) #l1 norm of the vector print("l1 norm:", norm\_l1)  norm\_l2 = norm(arr) #l2 norm of the vector print("l2 norm: ", norm\_l2)  max\_norm = norm(arr, np.inf) print("maxnorm: ", max\_norm)  **Output**  [1 2 3 4 5]  l1 norm: 15.0  l2 norm: 7.416198487095663  maxnorm: 5.0 |

|  |
| --- |
| #Define a matrix, add, subtract, multiply, divide and hadamard product of the matrices import numpy as np  a = np.matrix([[1, 6],[2, 3]]) b = np.matrix([[5, 1],[3, 2]])  c = a+b #Add print("Sum \n",c)  c = a-b #Subtract print("Difference \n",c)  c = a\*b #multiply print("Multiply \n",c)  c = a/b # divide print("Divide \n",c)  c = np.multiply(a, b) #Hadamard product print("Hadamard product: \n",c)  **Output**  Sum  [[6 7]  [5 5]]  Difference  [[-4 5]  [-1 1]]  Multiply  [[23 13]  [19 8]]  Divide  [[0.2 6. ]  [0.66666667 1.5 ]]  Hadamard product:  [[5 6]  [6 6]] |

|  |
| --- |
| #vector multiplication import numpy as np  A = np.array([[1, 2], [3, 4]])  v = np.array([5, 6])  w = np.dot(A, v) print(w)  **Output**  [17 39] |

|  |
| --- |
| # scalar multiplication import numpy as np   a = np.array([[1, 2],[3, 4]]) s = 2  c = s\*a print(c)  **Output**  [[ 2 4]  [ 6 8]] |

|  |
| --- |
| #define a 3x3 matrix, extract the main diagonal,create a diagonal matrix from the extracted vector import numpy as np   a = np.matrix([[1,2,3],[4,5,6],[7,8,9]]) print("Original vector:\n",a)  dia\_a = np.diag(a) print("Diagonal of the matrix: \n",dia\_a)  d = np.diag(dia\_a) print("Diagonal matrix from the vector: \n",d)  **Output //didnt got correct answer**  Original vector:  [[1 2 3]  [4 5 6]  [7 8 9]]  Diagonal of the matrix:  [1 5 9]  Diagonal matrix from the vector:  [[1 0 0]  [0 5 0]  [0 0 9]] |

|  |
| --- |
| #identity matrix of order 4 import numpy as np   b = np.identity(4, dtype=int) print(b)  **Output**  [[1 0 0 0]  [0 1 0 0]  [0 0 1 0]  [0 0 0 1]] |

|  |
| --- |
| #Transpose of a matrix import numpy as np  Rows = int(input("Enter the number of rows here: ")) Columns = int(input("Enter the number of columns here: "))  print("Enter the entries in a single line (separated by space):")  entries = list(map(int, input().split()))  matrix = np.array(entries).reshape(Rows, Columns) print("Original matrix:\n", matrix)  tr = matrix.transpose() print("Matrix after transpose: \n",tr)  **Output**  Enter the number of rows here: 3  Enter the number of columns here: 3  Enter the entries in a single line (separated by space):  1 2 3 4 5 6 7 8 9  Original matrix:  [[1 2 3]  [4 5 6]  [7 8 9]]  Matrix after transpose:  [[1 4 7]  [2 5 8]  [3 6 9]] |

|  |
| --- |
| #inverse of a matrix  import numpy as np  def calculate\_inverse(matrix):  try:  inverse = np.linalg.inv(matrix)  return inverse  except np.linalg.LinAlgError as e:  if 'Singular matrix' in str(e):  print("Error: The matrix is singular and cannot be inverted.")  else:  print("Error:", str(e))  # Get matrix dimensions R = int(input("Enter the number of rows: ")) C = int(input("Enter the number of columns: "))  print("Enter the entries in a single line (separated by space):") entries = list(map(int, input().split())) a = np.array(entries).reshape(R, C)  print("Original matrix:\n", a)  b = calculate\_inverse(a) if b is not None:  print("Inverse matrix:\n", b)  **Output**  Enter the number of rows: 3  Enter the number of columns: 3  Enter the entries in a single line (separated by space):  1 2 3 0 1 4 5 6 0  Original matrix:  [[1 2 3]  [0 1 4]  [5 6 0]]  Inverse matrix:  [[-24. 18. 5.]  [ 20. -15. -4.]  [ -5. 4. 1.]] |

|  |
| --- |
| #determinant of a matrix import numpy as np  R = int(input("Enter the number of rows: ")) C = int(input("Enter the number of columns: "))  print("Enter the entries in a single line (separated by space):")  entries = list(map(int, input().split()))  a = np.array(entries).reshape(R, C) print("Original matrix:\n", a)  b = np.linalg.det(a) print("Determinant:\n",b)  **Output**  Enter the number of rows: 3  Enter the number of columns: 3  Enter the entries in a single line (separated by space):  4 -3 5 1 0 3 -1 5 2  Original matrix:  [[ 4 -3 5]  [ 1 0 3]  [-1 5 2]]  Determinant:  -20.000000000000007 |

Set 3

|  |
| --- |
| # 1. Create an orthogonal matrix and check Q^TQ=QQ^T=I import numpy as np  def create\_orthogonal\_matrix(n):  A = np.random.rand(n, n)    Q, \_ = np.linalg.qr(A)    return Q  def check\_orthogonal(Q):  Qt = np.transpose(Q)  identity = np.identity(Q.shape[0])    result1 = np.allclose(np.dot(Qt, Q), identity)  result2 = np.allclose(np.dot(Q, Qt), identity)    return result1, result2 matrix\_size = int(input("Enter the size of the matrix: "))  orthogonal\_matrix = create\_orthogonal\_matrix(matrix\_size) print(orthogonal\_matrix) result\_QtQ, result\_QQt = check\_orthogonal(orthogonal\_matrix)  if result\_QtQ and result\_QQt:  print("Q^TQ=QQ^T=I , The matrix is orthogonal.") else:  print("The matrix is not orthogonal.") |

Output:

Enter the size of the matrix: 5

[[-0.57683159 -0.33186696 -0.17065852 0.57056382 -0.44995803]

[-0.46532675 -0.25545407 0.21699663 -0.77146976 -0.27561043]

[-0.37559559 0.59356305 -0.68370975 -0.18621216 0.06690937]

[-0.34192984 0.62432091 0.67329756 0.19822989 -0.02612848]

[-0.43903106 -0.28725223 0.05476773 0.07294708 0.84641426]]

Q^TQ=QQ^T=I , The matrix is orthogonal.

|  |
| --- |
| # 2. print rank of a matrix import numpy as np  R = int(input("Enter the number of rows: ")) C = int(input("Enter the number of columns: "))  print("Enter the entries in a single line (separated by space):")  entries = list(map(int, input().split())) matrix = np.array(entries).reshape(R, C)   print("Matrix:\n", matrix)  a = np.linalg.matrix\_rank(matrix) print("The rank of the matrix is: ", a) |

Output:

Enter the number of rows: 3

Enter the number of columns: 3

Enter the entries in a single line (separated by space):

1 2 3 4 5 6 7 8 9

Matrix:

[[1 2 3]

[4 5 6]

[7 8 9]]

The rank of the matrix is: 2

|  |
| --- |
| # 3. calculate sparsity of a matrix import numpy as np  R = int(input("Enter the number of rows: ")) C = int(input("Enter the number of columns: "))  print("Enter the entries in a single line (separated by space):")  entries = list(map(int, input().split())) A = np.array(entries).reshape(R, C)  zero\_count = np.count\_nonzero(A == 0) total\_elements = A.size sparsity = zero\_count / total\_elements percent=sparsity\*100  print("Number of zero elements: ", zero\_count) print("Total number of elements: ", total\_elements) print("Sparsity: ", sparsity) print("Sparsity percentage: ",percent,"%") |

Output:

Enter the number of rows: 3

Enter the number of columns: 3

Enter the entries in a single line (separated by space):

3 2 5 0 4 0 5 0 1

Number of zero elements: 3

Total number of elements: 9

Sparsity: 0.3333333333333333

Sparsity percentage: 33.33333333333333 %

|  |
| --- |
| # 4. print eigenvalues and eigenvectors of a matrix  import numpy as np from numpy.linalg import eig  a = np.array([[2, 2, 4],  [1, 3, 5],  [2, 3, 4]]) w,v = eig(a) print("Eigenvalue: ", w) print("Eigenvector: ", v) |

Output:

Eigenvalue: [ 8.80916362 0.92620912 -0.73537273]

Eigenvector: [[-0.52799324 -0.77557092 -0.36272811]

[-0.604391 0.62277013 -0.7103262 ]

[-0.59660259 -0.10318482 0.60321224]]

|  |
| --- |
| # 5. Calculate eigenvalues and eigenvectors of a matrix and reconstruct the matrix import numpy as np  A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) print("The matrix A is:") print(A) w, v = np.linalg.eig(A) # w: eigenvalues, v: eigenvectors  print("The eigenvalues of A are:") print(w) print("The eigenvectors of A are:") print(v) # Reconstruct the matrix using the eigenvalues and eigenvectors B = v @ np.diag(w) @ np.linalg.inv(v) print("The reconstructed matrix A is:") print(B) |

Output:

The matrix A is:

[[1 2 3]

[4 5 6]

[7 8 9]]

The eigenvalues of A are:

[ 1.61168440e+01 -1.11684397e+00 -9.75918483e-16]

The eigenvectors of A are:

[[-0.23197069 -0.78583024 0.40824829]

[-0.52532209 -0.08675134 -0.81649658]

[-0.8186735 0.61232756 0.40824829]]

The reconstructed matrix A is:

[[1. 2. 3.]

[4. 5. 6.]

[7. 8. 9.]]

|  |
| --- |
| # 6. define a 5x2 matrix dataset, # split it into x and y components and plot the dataset as scatterplot  import numpy as np import matplotlib.pyplot as plt  dataset = np.array([[6, 1],  [4, 7],  [4, 2],  [2, 8],  [9, 5]])  x = dataset[:, 0] y = dataset[:, 1]   plt.scatter(x, y) plt.xlabel('x') plt.ylabel('y') plt.title('Scatter plot of 5x2 matrix dataset') plt.show()  Output: |

