**Name :Vikas Yadav Subject: LINEAR ALGEBRA USING PYTHON**

**Class: SYBSC(COMPUTER SCIENCE) Exam Seat no: 90**

**Academic Year : 2019-2020**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Date | TITLE | Pg No | SIGN |
| 1 | **26/11/19** | **Write a program which demonstrate the following:**  **i)Addition of two complex numbers.**  **ii) Displaying the conjugate of a complex number.**  **iii) Plotting a set of complex numbers.**  **iv) Creating a new plot by rotating the given number by a degree 90, 180, 270 degrees and also by scaling b a number a=1/2, a=1/3, a=2 etc.** |  |  |
| 2 | **03/12/19** | **Write a program to do the following-**  **⦁ Enter a vector u as a n-list.**  **⦁ Enter another vector v as a n-list.**  **⦁ Find the vector au+bv for different values of a and b.**  **⦁ Find the dot product of u and v** |  |  |
| 3 | **10/12/2019** | **Write a program to do the following-**  **Enter two distinct faces as vectors u and v.**  **⦁ Find a new face as a liner combination of u and v i.e. au + bv for a and b in R.**  **⦁ Average Face Value** |  |  |
| 4 | **17/12/19** | **Write a program to do the following:**  **i) Enter an r by c matrix M(r and c being positive integers).**  **ii) Display M in matrix format.**  **iii) Display the row and columns of the matrix M.**  **iv) Find the scalar multiplication of M for a given scalar.**  **v) Find the transpose of the matrix M.** |  |  |
| 5 | **24/12/2019** | **Write a program to do the following-**  **i) Find the vector-matrix multiplication of a r by c matrix M with an c- vector u.**  **ii) Find the matrix- matrix product of M with a c by p matrix N.** |  |  |
| 6 | **07/01/2019** | **Write a program to enter a matrix and check if it is invertible. If the inverse exists, find the inverse.** |  |  |
| 7 | **14/01/2020** | **Write a program to do the following:**  **(i)Enter a positive number N and find numbers a and b such that**  **a2 – b2 = N**  **(ii) Find the gcd of two numbers using Euclid’s algorithm** |  |  |
| 8 | **21/01/2020** | **i) Enter a vector b and find the projection of b orthogonal to a given vector u.**  **ii) Find the projection of b orthogonal to a set of given vectors**. |  |  |

**Linear Algebra using Python**

**Practical 1**

**Aim: Write a program which demonstrate the following:**

**Addition of two complex numbers.**

**Displaying the Conjugate of two complex numbers.**

**Plotting a set of Complex numbers.**

**Creating a new plot by rotating the given number by a degree 90,180,270 degrees and also by a Scaling b a number a=1/2,a=1/3,a=2,etc.**

**Source Code:**

**import matplotlib.pyplot as plt**

**S={3+3j,4+3j,2+1j,2.5+1j,3+1j,3.25+1j}**

**print("Select operation")**

**print("1. Addition of two complex numbers")**

**print("2. Plot points from set of complex numbers")**

**print("3. Translation")**

**print("4.Scaling")**

**print("5.Rotation")**

**print("6. Exit")**

**while True:**

**ch=int(input("Enter the choice of operation"))**

**if ch==1:**

**c1=complex(input("Enter complex number 1"))**

**c2=complex(input("Enter complex number 2"))**

**print("Addition of two complex numbers (c1+c2) is:",c1+c2)**

**elif ch==2:**

**S1={x for x in S}**

**X=[x.real for x in S1]**

**Y=[x.imag for x in S1]**

**plt.plot(X,Y,'ro')**

**plt.axis([-6,6,-6,6])**

**plt.show()**

**elif ch==3:**

**c1=complex(input("Enter Translation in complex number format"))**

**S1={x+c1 for x in S}**

**print("Translation of number is:",S1)**

**elif ch==4:**

**scale=float(input("Enter scale point like (0.5) for 1/2"))**

**S1={x\*scale for x in S}**

**print("Scaling of number is",S1)**

**elif ch==5:**

**angle=int(input("enter angle of rotation 90/180/270"))**

**if angle==90:**

**S1={x\*1j for x in S}**

**print("Rotation of no is:",S1)**

**elif angle==180:**

**S1={x\*-1 for x in S}**

**print("Rotation of no is:",S1)**

**elif angle==270:**

**S1={x\*1j\*-1 for x in S}**

**print("Rotation of no is:",S1)**

**else:**

**print("Invalid angle,Enter only 90/180/270 degree")**

**else:**

**break**

**Output:**

**Select operation**

**1. Addition of two complex numbers**

**2. Plot points from set of complex numbers**

**3. Translation**

**4.Scaling**

**5.Rotation**

**6. Exit**

**Enter the choice of operation1**

**Enter complex number 11+2j**

**Enter complex number 22+1j**

**Addition of two complex numbers (c1+c2) is: (3+3j)**

**Enter the choice of operation2**

**Enter the choice of operation3**

**Enter Translation in complex number format2+3j**

**Translation of number is: {(4+4j), (4.5+4j), (5.25+4j), (5+4j), (5+6j), (6+6j)}**

**Enter the choice of operation4**

**Enter scale point like (0.5) for 1/20.5**

**Scaling of number is {(1+0.5j), (1.25+0.5j), (1.625+0.5j), (1.5+0.5j), (1.5+1.5j), (2+1.5j)}**

**Enter the choice of operation5**

**enter angle of rotation 90/180/270180**

**Rotation of no is: {(-4-3j), (-3-3j), (-3-1j), (-2-1j), (-2.5-1j), (-3.25-1j)}**

**=========================================================**

**Practical 2**

**Aim: Write a program to do the following:**

**i)Enter the vector u as a n-list.**

**ii)Enter another vector v as a n-list.**

**iii)Find a vector au + bv for different values of a and b.**

**iv)Find the dot product of u and v.**

**Source Code:**

**def addvec(x,y):**

**return [x[i]+y[i] for i in range(len(x))]**

**def subvec(x,y):**

**return [x[i]-y[i] for i in range(len(x))]**

**def scalermul(x,p):**

**return [p\*x[i] for i in range(len(x))]**

**def dotprod(x,y):**

**return sum([x[i]\*y[i] for i in range(len(x))])**

**v=[]**

**u=[]**

**n=int(input(“Enter the Number you want to add in vector:”))**

**print(“Enter Elements of vector u:”)**

**for i in range(n):**

**elem=int(input(“Enter the Element:”))**

**u.append(elem)**

**print(“Vector u=”,u)**

**print(“Enter Elements of vector v:”)**

**for i in range(n):**

**elem=int(input(“Enter the Element:”))**

**v.append(elem)**

**print(“Vector v=”,v)**

**while True:**

**print(“\n Select Vector Operations:”)**

**print(“1:Addition”)**

**print(“2:Subtraction”)**

**print(“3:Scalar Multiplication”)**

**print(“4:Dot Product”)**

**print(“5:Exit”)**

**ch=int(input(“Enter your Choice:”))**

**if ch==1:**

**print(“Addition of vector u and v is (u+v)”,addvec(u,v))**

**elif ch==2:**

**print(“Subtraction of vector u and v is (u-v)”,subvec(u,v))**

**elif ch==3:**

**print(“To perform Scalar multiplication au”)**

**a=int(input(“Enter the value of a”))**

**print(“Scalar multiple of vector au is”,scalermul(u,a))**

**elif ch==4:**

**print(“Dot product of vector u and v is (u.v)”,dotprod(u,v))**

**else:**

**break**

**Output:**

**Enter the Number you want to add in vector:2**

**Enter Elements of vector u:**

**Enter the Element:12**

**Vector u= [12]**

**Enter the Element:13**

**Vector u= [12, 13]**

**Enter Elements of vector v:**

**Enter the Element:14**

**Vector v= [14]**

**Enter the Element:15**

**Vector v= [14, 15]**

**Select Vector Operations:**

**1:Addition**

**2:Subtraction**

**3:Scalar Multiplication**

**4:Dot Product**

**5:Exit**

**Enter your Choice:1**

**Addition of vector u and v is (u+v) [26, 28]**

**Select Vector Operations:**

**1:Addition**

**2:Subtraction**

**3:Scalar Multiplication**

**4:Dot Product**

**5:Exit**

**Enter your Choice:2**

**Subtraction of vector u and v is (u-v) [-2, -2]**

**Select Vector Operations:**

**1:Addition**

**2:Subtraction**

**3:Scalar Multiplication**

**4:Dot Product**

**5:Exit**

**Enter your Choice:3**

**To perform Scalar multiplication au**

**Enter the value of a2**

**Scalar multiple of vector au is [24, 26]**

**Select Vector Operations:**

**1:Addition**

**2:Subtraction**

**3:Scalar Multiplication**

**4:Dot Product**

**5:Exit**

**Enter your Choice:4**

**Dot product of vector u and v is (u.v) 363**

**Select Vector Operations:**

**1:Addition**

**2:Subtraction**

**3:Scalar Multiplication**

**4:Dot Product**

**5:Exit**

**Enter your Choice:5**

**==================================================================**

**Practical 3(a)**

**Aim: Write a program to do the following-**

**Enter two distinct faces as vectors u and v.**

**Find a new face as a liner combination of u and v i.e. au + bv for a and b in R.**

**Source Code:**

**def scalermul(x,p):**

**return[p\*x[i] for i in range(len(x))]**

**def lin\_comb(vlist,clist):**

**s=[(scalermul(vlist[i],clist[i])) for i in range(len(vlist))]**

**l=[]**

**for j in range(len(s[0])):**

**su=0**

**for i in range(len(s)):**

**su=su+s[i][j]**

**l.append(su)**

**return l**

**print(lin\_comb([[1,2,3],[0,1,4],[2,-1,1]],[2,1,3]))**

**print(lin\_comb([[1,1],[2,3]],[-6,4]))**

**Output:**

**[8, 2, 13]**

**[2, 6]**

**Practical 3(b)**

**Aim: Average face value.**

**Source Code:**

**l=int(input('Enter length of vector'))**

**u=[]**

**v=[]**

**c=[]**

**print('enter elements of vector u')**

**for i in range(l):**

**n=int(input('enter no'))**

**u.append(n)**

**print('enter elements of vector v')**

**for i in range(l):**

**n=int(input('enter no'))**

**v.append(n)**

**print('enter elements of coeficient')**

**c1=int(input('enter coeficent1'))**

**c2=int(input('enter coeficient2'))**

**newface=[c1\*u[i]+c2\*v[i] for i in range(len(u))]**

**print('New Face of u and v=',newface)**

**avgface=[(u[i]+v[i]/2) for i in range(len(u))]**

**print('average face of u and v =', avgface)**

**Output:**

**Enter length of vector2**

**enter elements of vector u**

**enter no3**

**enter no2**

**enter elements of vector v**

**enter no3**

**enter no2**

**enter elements of coeficient**

**enter coeficent11**

**enter coeficient22**

**New Face of u and v= [9, 6]**

**average face of u and v = [4.5, 3.0]**

**Practical 4**

**Aim: Write a program to do the following:**

**i)Enter an r by c matrix M(r and c being positive integers).**

**ii)Display M in matrix format.**

**iii)Display the number of rows and columns of matrix M.**

**iv)Find the scalar multiple of M for a given scalar.**

**v)Find the transpose of matrix M.**

**Source Code:**

**global r,c**

**def printmatrix(A):**

**print("The entered matrix M is:")**

**for i in range(r):**

**print(A[i])**

**def printrows(A):**

**print("Rows of Matrix:")**

**for i in range(r):**

**print("Row %d="%i,A[i])**

**def printcolumns(A):**

**print("Columns of Matrix:")**

**for j in range(c):**

**print("Columns %d="%j)**

**for i in range(r):**

**print(A[i][j])**

**print("\n")**

**def scalarmul(A,s):**

**N=[[s\*A[i][j] for j in range(c)] for i in range(r)]**

**print("The scalar multiplication s\*M=")**

**printmatrix(N)**

**def transpose(A):**

**T=[[A[i][j] for i in range(r)] for j in range(c)]**

**print("Transpose of Matrix=")**

**for j in range(c):**

**print(T[j])**

**print("Enter the Dimensions of matrix:")**

**r=int(input("Enter the number of rows:"))**

**c=int(input("Enter the number of columns:"))**

**M=[]**

**for i in range(r):**

**print("Enter the Elements of row",i)**

**M.append([])**

**for j in range(c):**

**n=int(input("Enter the number:"))**

**M[i].append(n)**

**while True:**

**print("\n Select Operation:")**

**print("1:Display Matrix")**

**print("2:Display rows of Matrix")**

**print("3:Display columns of Matrix")**

**print("4:Scalar Multiplication of Matrix")**

**print("5:Transpose of Matrix")**

**print("6:Exit")**

**ch=int(input("Enter the Choice for Operation:"))**

**if ch==1:**

**printmatrix(M)**

**elif ch==2:**

**printrows(M)**

**elif ch==3:**

**printcolumns(M)**

**elif ch==4:**

**sc=int(input("Enter the Scalar value:"))**

**scalarmul(M,sc)**

**elif ch==5:**

**transpose(M)**

**elif ch==6:**

**break;**

**Output:**

**Enter the Dimensions of matrix:**

**Enter the number of rows:2**

**Enter the number of columns:2**

**Enter the Elements of row 0**

**Enter the number:3**

**Enter the number:4**

**Enter the Elements of row 1**

**Enter the number:5**

**Enter the number:6**

**Select Operation:**

**1:Display Matrix**

**2:Display rows of Matrix**

**3:Display columns of Matrix**

**4:Scalar Multiplication of Matrix**

**5:Transpose of Matrix**

**6:Exit**

**Enter the Choice for Operation:1**

**The entered matrix M is:**

**[3, 4]**

**[5, 6]**

**Select Operation:**

**1:Display Matrix**

**2:Display rows of Matrix**

**3:Display columns of Matrix**

**4:Scalar Multiplication of Matrix**

**5:Transpose of Matrix**

**6:Exit**

**Enter the Choice for Operation:2**

**Rows of Matrix:**

**Row 0= [3, 4]**

**Row 1= [5, 6]**

**Select Operation:**

**1:Display Matrix**

**2:Display rows of Matrix**

**3:Display columns of Matrix**

**4:Scalar Multiplication of Matrix**

**5:Transpose of Matrix**

**6:Exit**

**Enter the Choice for Operation:3**

**Columns of Matrix:**

**Columns 0=**

**3**

**5**

**Columns 1=**

**4**

**6**

**Select Operation:**

**1:Display Matrix**

**2:Display rows of Matrix**

**3:Display columns of Matrix**

**4:Scalar Multiplication of Matrix**

**5:Transpose of Matrix**

**6:Exit**

**Enter the Choice for Operation:4**

**Enter the Scalar value:2**

**The scalar multiplication s\*M=**

**The entered matrix M is:**

**[6, 8]**

**[10, 12]**

**Select Operation:**

**1:Display Matrix**

**2:Display rows of Matrix**

**3:Display columns of Matrix**

**4:Scalar Multiplication of Matrix**

**5:Transpose of Matrix**

**6:Exit**

**Enter the Choice for Operation:5**

**Transpose of Matrix=**

**[3, 5]**

**[4, 6]**

**Select Operation:**

**1:Display Matrix**

**2:Display rows of Matrix**

**3:Display columns of Matrix**

**4:Scalar Multiplication of Matrix**

**5:Transpose of Matrix**

**6:Exit**

**Enter the Choice for Operation:6**

**>>>**

**==================================================================**

**Practical 5**

**Aim: Write a program to do the following**

**Find a vector matrix multiplication of a Rby C matrix M with an column vector v.**

**Find the matrix-matrix product of M with a with a Column by Row matrix N.**

**Source Code:**

**global r1,c1,r2,c2**

**def printmatrix(A):**

**for i in range(len(A)):**

**print(A[i])**

**def matrixadd(A,B):**

**C=[[A[i][j]+B[i][j] for j in range(len(B[0])) ]for i in range(len(A))]**

**print("Addition of 2 Matrix")**

**printmatrix(C)**

**def matrixmul(A,B):**

**C=[[sum(A[i][k]\*B[k][j] for k in range(len(B)))for j in range(len(B[0]))]for i in range(len(A))]**

**print("Multiplication of 2 matrix")**

**printmatrix(C)**

**def matrixvecmul(A,v):**

**C=[sum(A[i][j]\*v[j] for j in range(len(v)))for i in range(len(A))]**

**print("Matrix vector Multiplication(Ml\*v)")**

**printmatrix(C)**

**def vecmatrixmul(v,A):**

**C=[sum(v[j]\*A[i][j]for j in range(len(v)))for i in range(len(A))]**

**print("Vector matrix multiplication(v\*Ml)")**

**printmatrix(C)**

**print("Enter the Dimensions of Matrix1")**

**r1=int(input("Enter the number of rows:"))**

**c1=int(input("Enter the number of columns:"))**

**M=[]**

**for i in range(r1):**

**print("Enter elements of row",i)**

**M.append([])**

**for j in range(c1):**

**n=int(input("Enter number"))**

**M[i].append(n)**

**print("The entered matrix M1 is:")**

**printmatrix(M)**

**print("Enter the Dimensions of Matrix2")**

**r2=int(input("Enter the number of rows:"))**

**c2=int(input("Enter the number of columns:"))**

**N=[]**

**for i in range(r2):**

**print("Enter elements of row",i)**

**N.append([])**

**for j in range(c2):**

**n=int(input("Enter number"))**

**N[i].append(n)**

**print("The entered matrix M2 is:")**

**printmatrix(N)**

**print("Select Operation:")**

**print("1: Matrix Addition")**

**print("2:Matrix Multiplication")**

**print("3:Matrix vector multiplication")**

**print("4:Vector Matrix multiplication")**

**print("5:Exit")**

**while True:**

**ch=int(input("Enter the choice of operation"))**

**if ch==1:**

**if(r1,c1)==(r2,c2):**

**matrixadd(M,N)**

**else:**

**print("Invalid matrix :To multiply 2 matrices matrix1 column=matrix2 row")**

**elif ch==2:**

**if c1==r2:**

**matrixmul(M,N)**

**else:**

**print("Invalid matrix :To multiply 2 matrices matrix1 column=matrix2 row")**

**elif ch==3:**

**s=input("Enter elements of vector separated by spaces")**

**v=[int(x) for x in s.split()]**

**print(len(v))**

**if len(v)!=c1:**

**print("Invalid vector add vector of %d elements (Columns of M1)"%c1)**

**else:**

**matrixvecmul(M,v)**

**elif ch==4:**

**s=input("Enter elements of vector separated by spaces")**

**v=[int(x) for x in s.split()]**

**if len(v)!=r1:**

**print("Invalid vector add vector of %d elements (Rows of M1)"%r1)**

**else:**

**vecmatrixmul(v,M)**

**elif ch==5:**

**break**

**else:**

**break**

**Output*:***

**Enter the Dimensions of Matrix1**

**Enter the number of rows:2**

**Enter the number of columns:2**

**Enter elements of row 0**

**Enter number1**

**Enter number2**

**Enter elements of row 1**

**Enter number3**

**Enter number4**

**The entered matrix M1 is:**

**[1, 2]**

**[3, 4]**

**Enter the Dimensions of Matrix2**

**Enter the number of rows:2**

**Enter the number of columns:2**

**Enter elements of row 0**

**Enter number5**

**Enter number6**

**Enter elements of row 1**

**Enter number7**

**Enter number8**

**The entered matrix M2 is:**

**[5, 6]**

**[7, 8]**

**Select Operation:**

**1: Matrix Addition**

**2:Matrix Multiplication**

**3:Matrix vector multiplication**

**4:Vector Matrix multiplication**

**5:Exit**

**Enter the choice of operation1**

**Addition of 2 Matrix**

**[6, 8]**

**[10, 12]**

**Enter the choice of operation2**

**Multiplication of 2 matrix**

**[19, 22]**

**[43, 50]**

**Enter the choice of operation4**

**Enter elements of vector separated by spaces 1 2**

**Vector matrix multiplication(v\*Ml)**

**5**

**11**

**Enter the choice of operation3**

**Enter elements of vector separated by spaces3 4**

**2**

**Matrix vector Multiplication(Ml\*v)**

**11**

**25**

**Enter the choice of operation5**

**===============================================================**

**Practical 6**

**Aim: Write a program to enter a matrix and check if it is invertible if the inverse exists find the inverse.**

**Source Code:**

**def printmatrix(A):**

**for i in range(len(A)):**

**print(A[i])**

**def transpose(A):**

**T=[[A[i][j]for i in range(len(A))]for j in range(len(A))]**

**return T**

**c=int(input("Enter the number of rows and columns of square matrix"))**

**r=c**

**M=[]**

**for i in range(r):**

**print("Enter the elementsof row:",i)**

**M.append([])**

**for j in range(c):**

**n=int(input("Enter number"))**

**M[i].append(n)**

**print("the entered Matrix M1 is")**

**printmatrix(M)**

**determinant=0**

**if r==2:**

**determinant=M[0][0]\*M[1][1]-M[0][1]\*M[1][0]**

**print("Determinant:",determinant)**

**if determinant==0:**

**print("Matrix not invertible")**

**else:**

**print("Matrix is invertible")**

**CFM=[]**

**for i in range(2):**

**CFM.append([])**

**CFM[0].append(M[1][1])**

**CFM[0].append(-(M[0][1]))**

**CFM[0].append(-(M[1][0]))**

**CFM[0].append(M[0][0])**

**print("Cofactor matrix")**

**printmatrix(CFM)**

**MI=[]**

**for i in range(1):**

**MI.append([])**

**for j in range(2):**

**MI[i].append(CFM[i][j]/determinant)**

**print("Inverse of A matrix M is:")**

**printmatrix(MI)**

**else:**

**for i in range(3):**

**determinant=determinant+(M[0][i]\*M[1][(i+1)%3]\*M[2][(i+2)%3]-M[1][(i+2)%3]\*M[2][(i+1)%3])**

**print("Determinant",determinant)**

**if determinant==0:**

**print("Matrix is not invertible")**

**else:**

**print("Matrix is invertible")**

**CFM=[]**

**for i in range(3):**

**CFM.append([])**

**for j in range(3):**

**v=(M[(i+1)%3][(j+1)%3]\*M[(i+2)%3][(j+2)%3])-(M[(i+1)%3][(j+2)%3]\*M[(i+2)%3][(j+1)%3])**

**CFM[i].append(v)**

**print("Cofactor matrix")**

**printmatrix(CFM)**

**AdjM=transpose(CFM)**

**MI=[]**

**for i in range(3):**

**for j in range(3):**

**MI(i).append(AdjM[i][j]/determinant)**

**print("inverse of A,Matrix Mis:")**

**printmatrix(MI)**

**Output:**

**Enter the number of rows and columns of square matrix2**

**Enter the elements of row 0**

**Enter number12**

**Enter number13**

**Enter the elementsof row 1)**

**Enter number14**

**Enter number15**

**the entered Matrix M1 is**

**[12, 13]**

**[14, 15]**

**Determinant: -2)**

**Matrix is invertible**

**Cofactor matrix**

**[15, -13, -14, 12]**

**Inverse of A matrix M is:**

**[-8, 6]**

**Cofactor matrix**

**[15, -13, -14, 12, 15, -13, -14, 12]**

**[]**

**Inverse of A matrix M is:**

**[-8, 6]**

**>>>**

**Practical 7**

**Aim: Write a Positive Number N and find numbers a and b such that a2-b2=N**

**Find the GCD of two Numbers using Euclid’s Algorithm.**

**Source Code:**

**from math import \***

**pf=[]**

**n=int(input("Enter Number:"))**

**x=n**

**while n%2==0:**

**pf.append(2)**

**n=n/2**

**i=3**

**while i<=sqrt(n):**

**while n%i==0:**

**pf.append(i)**

**n=n/i**

**i=i+2**

**if n>2:**

**pf.append(n)**

**print("Prime Factors of",x,"area",pf)**

**pf1=set (pf)**

**nf=1**

**for f in pf1:**

**cnt=0**

**for f1 in pf:**

**if f==f1:**

**cnt+=1**

**nf\*=cnt+1**

**print("Number of factors",x,"=",nf)**

**print("number of positive integral solutions=",nf/2)**

**Output:**

**Enter Number:5**

**Prime Factors of 5 area [5]**

**Number of factors 5 = 2**

**number of positive integral solutions= 1.0**

**==================================================================**

**Practical 8**

**Aim: Write a program to do the following**

**Enter a vector b and find the projection of b orthogonal to a given vector u.**

**To find the projection of b orthogonal to a set of given vectors.**

**Source Code:**

**def dot(x,y):**

**return sum([x[i]\*y[i] for i in range(len(x))])**

**def scalar(a,v):**

**return [a\*v[i] for i in range(len(v))]**

**def sub(u,v):**

**return [u[i]-v[i] for i in range(len(v))]**

**def project\_along(b,v):**

**sigma=(dot(b,v)/dot(v,v))if dot(v,v)!=0 else 0**

**return scalar(sigma,v)**

**def project\_orthogonal(b,v):return sub(b,project\_along(b,v))**

**def project\_orthogonalvectorset(b,s):**

**for i in range(len(s)):**

**v=s[i]**

**b=project\_orthogonal(b,v)**

**return(b)**

**print(project\_orthogonal([5,-5,2],[8,-2,2]))**

**Output:**

**[-1.0, -3.5, 0.5]**