Q1:

import math as m

import numpy as np

def vector(vec, xvec, yvec, zvec):

    magnitude = np.linalg.norm(vec)

    print("Magnitude of Vector A =", magnitude)

    return m.acos(np.dot(vec, xvec) / magnitude), m.acos(np.dot(vec, yvec) / magnitude), m.acos(np.dot(vec, zvec) / magnitude)

ax = float(input("Enter Axi: "))

ay = float(input("Enter Ayj: "))

az = float(input("Enter Azk: "))

A = np.array([ax, ay, az])

B = np.array([1, 0, 0])

C = np.array([0, 1, 0])

D = np.array([0, 0, 1])

a, b, g = vector(A, B, C, D)

print("Alpha =", str(m.degrees(a)) + "\u00b0")

print("Beta =", str(m.degrees(b)) + "\u00b0")

print("Gamma =", str(m.degrees(g)) + "\u00b0")

import math as m

import numpy as np

def vector(vec, xvec, yvec, zvec):

    magnitude = np.linalg.norm(vec)

    print("Magnitude of Vector A =", magnitude)

    return m.acos(np.dot(vec, xvec) / magnitude), m.acos(np.dot(vec, yvec) / magnitude), m.acos(np.dot(vec, zvec) / magnitude)

ax = float(input("Enter Axi: "))

ay = float(input("Enter Ayj: "))

az = float(input("Enter Azk: "))

A = np.array([ax, ay, az])

B = np.array([1, 0, 0])

C = np.array([0, 1, 0])

D = np.array([0, 0, 1])

a, b, g = vector(A, B, C, D)

print("Alpha =", str(m.degrees(a)) + "\u00b0")

print("Beta =", str(m.degrees(b)) + "\u00b0")

print("Gamma =", str(m.degrees(g)) + "\u00b0")

Q2-3:

import numpy as np

import math as m

import matplotlib.pyplot as plt

def dampedOscillation( Xm, origAngFreq, phi, t, b):

    mass=1.0

    displacement=Xm\*m.exp((-b\*t)/(2\*mass))\*m.cos(origAngFreq\*t+phi)

    if b<2\*mass\*origAngFreq:

        condition="under damped"

    elif b==2\*mass\*origAngFreq:

        condition="critical damped"

    else:

        condition="over damped"

    return displacement, condition

Xm=float(input("Enter amplitude 'Xm': "))

origAngFreq = float(input("Enter angular frequency 'w': "))

phi=float(input("Enter Phase constant '\u03C6': "))

t=float(input("Enter time 't': "))

b=float(input("Enter damping coefficient 'b': "))

X,condition=dampedOscillation( Xm, origAngFreq, phi, t, b)

print("displacement =", X, "and the oscillation is", condition)

timeValues=np.linspace(0, 10, 1000)

displacementValues=[dampedOscillation( Xm, origAngFreq, phi, t, b)[0] for t in timeValues]

plt.plot(timeValues,displacementValues,label='Displacement')

plt.xlabel("Time")

plt.ylabel("Displacement")

plt.title("Damped Oscillation")

plt.legend()

plt.grid(True)

plt.show()

Q4:

import sympy as sp

import math as m

import numpy as np

import matplotlib.pyplot as plt

sp.init\_printing()

t = sp.symbols('t')

position = 3\*sp.sin(t)-2\*sp.cos(t)+5\*t\*\*3

velocity = sp.diff(position,t)

acceleration = sp.diff(position,t,2)

print("Expression for displacement:", position)

print("Expression for velocity:", velocity)

print("Expression for acceleration:", acceleration)

time\_values = np.linspace(0,10,100)

position\_values = [position.evalf(subs={t: val}) for val in time\_values]

velocity\_values = [velocity.evalf(subs={t: val}) for val in time\_values]

acceleration\_values = [acceleration.evalf(subs={t: val}) for val in time\_values]

plt.plot(time\_values, position\_values, label='Displacement')

plt.plot(time\_values, velocity\_values, label='Velocity')

plt.plot(time\_values, acceleration\_values, label='Acceleration')

plt.xlabel("Time (s)")

plt.ylabel("Value")

plt.title("Graph of Displacement, Velocity, and Acceleration vs Time")

plt.legend()

plt.grid(True)

plt.show()

Q5:

import math as m

magr=7.3

angle=m.radians(30)

rx=magr\*m.cos(angle)

ry=magr\*m.sin(angle)

print("a)")

print("X component of Vector R =", (rx))

print("b)")

print("Y component of Vector R =", (ry))

Q6:

import math as m

import numpy as np

import matplotlib.pyplot as plt

mag=10.0

angleA=m.radians(30)

angleB=m.radians(180-105)

Rx=mag\*m.cos(angleA)-mag\*m.cos(angleB)

Ry=mag\*m.sin(angleA)+mag\*m.sin(angleB)

R=np.array([Rx,Ry])

print("a)")

print("X-component of R =",Rx)

print("Y-component of R =",Ry)

magR=np.linalg.norm(R)

angleX=m.degrees(m.acos(Rx/magR))

print("b)")

print("Magnitude of R =",magR)

print("c)")

print("Angle R makes with the x-axis : ",angleX,"\u00b0")

Q7:

import math as m

import numpy as np

def degrees(angle\_rad):

    return angle\_rad\*180/m.pi

r1x=50\*(m.cos(30))+50\*(m.cos(195))+50\*(m.cos(315))

r1y=50\*(m.sin(30))+50\*(m.sin(195))+50\*(m.sin(315))

magr1=m.sqrt(pow(r1x,2)+pow(r1y,2))

thetar1=m.atan2(r1y,r1x)

print("i)")

print("magnitude and angle of vector a + b + c is", magr1,",",degrees(thetar1),"\u00b0")

r2x = 50\*(m.cos(30))-50\*(m.cos(195))+50\*(m.cos(315))

r2y=50\*(m.sin(30))-50\*(m.sin(195))+50\*(m.sin(315))

magr2=m.sqrt(pow(r2x,2)+pow(r2y,2))

thetar2=m.atan2(r2y,r2x)

print("ii)")

print("magnitude and angle of vector a - b + c is", magr2,",",degrees(thetar2),"\u00b0")

dx=50\*(m.cos(30))+50\*(m.cos(195))-50\*(m.cos(315))

dy=50\*(m.sin(30))+50\*(m.sin(195))-50\*(m.sin(315))

magd=m.sqrt(pow(dx,2)+pow(dy,2))

thetad=m.atan2(dy,dx)

print("iii)")

print("magnitude and angle of vector d is", magd,",",degrees(thetad),"\u00b0")

Q8:

import numpy as np

def Angle(s):

    cosx=s[0]/(np.linalg.norm(s))

    x=np.arccos(cosx)

    anglex=np.degrees(x)

    cosy=s[1]/(np.linalg.norm(s))

    y=np.arccos(cosy)

    angley=np.degrees(y)

    cosz=s[2]/(np.linalg.norm(s))

    z=np.arccos(cosz)

    anglez=np.degrees(z)

    print('The angle of Vector A with X-axis is : ',anglex)

    print('The angle of Vector A with Y-axis is : ',angley)

    print('The angle of Vector A with Z-axis is : ',anglez)

A=[2,-3,5]

Angle(A)