```
%pip install scipy --upgrade
%matplotlib inline
import numpy as np
import math
import scipy.optimize as opt
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (1.13.0)
Requirement already satisfied: numpy<2.3,>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from scipy) (1.25.2)
```

Q1

```
def objective1(g):
    return -(4*(1-0.75**g[0])+1/(1+g[0]))

x0 = np.array([0])
res = opt.minimize(objective1,x0)

print("Maximized x value:",res.x)
print("Maximum value of the original function:",-res.fun)
```

```
Maximized x value: [22.41016967]
Maximum value of the original function: 4.036375420390952
```

Q2

(a)

```
def objective2(g):
    sx=12.17
    sm=24.47
    so=37.72
    ss=8.66
    cxm=0.158
    cxo=0.078
    cxs=0.579
    cmo=0.241
    cms=0.302
    cos=0.282

    retVal=np.sqrt(((sx**2)*(g[0]**2))+((sm**2)*(g[1]**2))+((so**2)*(g[2]**2))
    +((ss**2)*(g[3]**2))+(2*sx*sm*cxm*g[0]*g[1])+(2*sx*so*cxo*g[0]*g[2])+(2*sx*ss*cxs*g[0]+g[3])
    +(2*sm*so*cmo*g[1]*g[2])+(2*sm*ss*cms*g[1]*g[3])+(2*so*ss*cos*g[2]*g[3]))
    return (retVal)


def constr1(g):
    return g[0]+g[1]+g[2]+g[3]
def constr2(g):
    return 14.67*g[0]+33.2*g[1]+59.36*g[2]+11.91*g[3]

x0 = np.array([0,0,0,0])
cons1 = opt.NonlinearConstraint(constr1, ub=1, lb=0)
cons2 = opt.NonlinearConstraint(constr2, ub=30, lb=30)
res = opt.minimize(objective2, x0,constraints=[cons1, cons2])

print("x value:", res.x)
print("Minimun value of the original function:", res.fun)
```

```
x value: [0.07560987 0.26933979 0.25599732 0.39905302]
Minimun value of the original function: 15.021714127606021
```

(b)

```
def objective3(g):
    return -(14.67*g[0]+33.2*g[1]+59.36*g[2]+11.91*g[3])
def constr1(g):
    return g[0]+g[1]+g[2]+g[3]
def constr2(g):
    sx=12.17
    sm=24.47
    so=37.72
    ss=8.66
    cxm=0.158
    cxo=0.078
    cxs=0.579
    cmo=0.241
    cms=0.302
    cos=0.282

    retVal=np.sqrt(((sx**2)*(g[0]**2))+((sm**2)*(g[1]**2))+((so**2)*(g[2]**2))
    +((ss**2)*(g[3]**2))+(2*sx*sm*cxm*g[0]*g[1])+(2*sx*so*cxo*g[0]*g[2])+(2*sx*ss*cxs*g[0]+g[3])
    +(2*sm*so*cmo*g[1]*g[2])+(2*sm*ss*cms*g[1]*g[3])+(2*so*ss*cos*g[2]*g[3]))
    return (retVal)


x0 = np.array([0,0,0,0])
cons1 = opt.NonlinearConstraint(constr1, ub=1, lb=0)
cons2 = opt.NonlinearConstraint(constr2, ub=10, lb=10)
res = opt.minimize(objective3, x0,constraints=[cons1, cons2])

print("Maximized x value:", res.x)
print("Maximun value of the original function:", -res.fun)
```

```
Maximized x value: [-0.00852673  0.17289432  0.15249821  0.45257997]
Maximun value of the original function: 20.05752564475212
```

(c)

```
def objective4(g):
    return -(14.67*g[0]+33.2*g[1]+59.36*g[2]+11.91*g[3])
def constr1(g):
    return g[0]+g[1]+g[2]+g[3]
def constr2(g):
    sx=12.17
    sm=24.47
    so=37.72
    ss=8.66
    cxm=0.158
    cxo=0.078
    cxs=0.579
    cmo=0.241
    cms=0.302
    cos=0.282

    retVal=np.sqrt(((sx**2)*(g[0]**2))+((sm**2)*(g[1]**2))+((so**2)*(g[2]**2))
    +((ss**2)*(g[3]**2))+(2*sx*sm*cxm*g[0]*g[1])+(2*sx*so*cxo*g[0]*g[2])+(2*sx*ss*cxs*g[0]+g[3])
    +(2*sm*so*cmo*g[1]*g[2])+(2*sm*ss*cms*g[1]*g[3])+(2*so*ss*cos*g[2]*g[3]))
    return (retVal)

x0 = np.array([0,0,0,0])
risk_levels = np.linspace(8, 40, 50)
```

```
risk_levels = np.linspace(8, 40, 50)
returns = []
risks = []

for risk in risk_levels:
    cons2 = opt.NonlinearConstraint(constr2, ub=risk, lb=risk)
    cons1 = opt.NonlinearConstraint(constr1, ub=1, lb=1)
    res = opt.minimize(objective4, x0, constraints=[cons1, cons2], method='SLSQP')

    if res.success:
        returns.append(-res.fun)
        risks.append(risk)

# 繪製結果
plt.plot(risks, returns, 'b-o')
plt.title('Efficient Frontier')
plt.xlabel('Risk')
plt.ylabel('Return')
plt.grid(True)
plt.show()
```