1 HW 1, Problem 1 - Python Solution

Details of the problem are as follows: + Cash Requirements = [100, 500, 100, -600, -500, 200, 600, -900] + Financial Instruments + Loan: Unlimited, duration = 8 quarters, interest = 1% per quarter, compounded quarterly + Loan: Unlimited, duration = 2 quarters, interest = 1.8% per quarter, compounded quarterly + Loan: Unlimited, duration = 1 quarter, interest = 2.5% per quarter, compounded quarterly + Bank account: Unlimited, interest = 0.5% per month

[1]:
```
# import relevant modules
import numpy as np
from scipy.optimize import linprog
```

[2]:
```
liability = np.array([100.0,500.0,100.0,-600.0,-500.0,200.0,600.0,-900.0,0])
num_quarters = len(liability);
lt_int = 1;
xt_int = 1.8;
yt_int = 2.5;
rf_int = 0.5;
```

[3]:
```
nvars = 1 + (num_quarters-2) + (num_quarters-1) + num_quarters
objvec = np.zeros(nvars)  # zero vector of length nvars
objvec[nvars-1] = 1; # the last component is to be maximized  -- last component␣
 ↪is the last component of the z vector
```

[4]:
```
# Equality constraints
Aeq = np.zeros((num_quarters,nvars))
for t in range(num_quarters):
    # coefficients of l_t
    if (t==0):
        Aeq[t,t] = 1;  # L_1 coefficient in Q1
        Aeq[num_quarters-1,t] = -(1+lt_int/100)**8;  #L_1 coefficient in Q9

    # coefficients of x_t
    xoffset = 1
    if (t<num_quarters-2):
        Aeq[t,xoffset+t] = 1;  # x_t coefficient
    if (t>1):
        Aeq[t,t-1] = -(1+xt_int/100)**2;   # x_{t-2} coefficient

    # coefficients of y_t
    yoffset = num_quarters-1
    if (t<num_quarters-1):
        Aeq[t,yoffset+t] = 1;   # y_t coefficient
    if (t>0):
        Aeq[t,yoffset+t-1] = -(1+yt_int/100);   # y_{t-1} coefficient

    # coefficient of z_t
    zoffset = 1+(num_quarters-2)+(num_quarters-1)
    if (t>0):
```

```
        Aeq[t,zoffset+t-1] = 1+rf_int/100;      # z_{t-1} coefficient
    Aeq[t,zoffset+t] = -1;    # z_t coefficient

beq = liability
```

[5]:
```
# set up and solve the optimization problem
res = linprog(-objvec,A_eq=Aeq,b_eq=beq)    #By default, linprog solves␣
 ↪minimization problems.
```

[6]:
```
if (res.status == 0): # if optimization solved correctly
    var = res.x
    l = var[0]
    x = var[xoffset:num_quarters-1]
    y = var[yoffset:yoffset+(num_quarters-1)]
    z = var[zoffset:zoffset+num_quarters]
    print("objective value = ", -res.fun)
    print("l = ", l)
    print("x = ", x)
    print("y = ", y)
    print("z = ", z)
else:
    print(res.message)
```

```
objective value =  471.56304857588367
l =  399.8095748847105
x =  [2.26928213e-04 1.98691111e+02 8.48009287e-04 1.41402332e-04
 1.21501065e-04 1.93836963e-04 2.00719432e-04]
y =  [2.40078620e-04 2.87611412e-04 9.99996404e+01 1.93464789e-04
 2.71961434e-04 1.39133063e-04 1.12479648e-03 2.62096726e-04]
z =  [2.99810046e+02 2.67463775e-04 2.31112335e-04 2.91592546e+02
 7.93049806e+02 5.97014970e+02 1.12515390e-03 9.00000004e+02
 4.71563049e+02]
```