Code for week3 planning model

At the part, we will have a look at the codes on the course, many thing will be understood after your coding.

First of all, we need to import the libraries we need to make the work run

In [ ]:

```
from scipy.optimize import linprog as lp
import numpy as np
import pulp
```

Exercise 1

For the first question, we need to use the scipy as the tool for solving the Linear programming

In [ ]:

```
c1 = [-72, -64]
A1 = [[1, 1],
     [12, 8],
     [3, 0]]
b1 = [50, 480, 100]
pot1 = lp(c = c1, A_ub = A1, b_ub = b1, method = 'highs')
x1 = pot1.x
val1 = -pot1.fun
print("The best choice of X are {0}, the max value of the func is {1}".format(x1, val1))
```

```
The best choice of X are [20. 30.], the max value of the func is 3360.0
```

Exercise 2

On the base of the exercise 1, we can have a more complex question for solving, but the method is the same,Only the formula are more complicated.

In [ ]:

```
c2 = [-24, -16, -44, -32, 3, 3]
A2 = [[1, 0, 0, 0, 1, 0],
      [1.0/3, 1.0/4, 0, 0, 1.0/3, 1.0/4],
      [4, 2, 0, 0, 6, 4]]
b2 = [100, 50, 480]
Aeq2 = [[0, 0, 1, 0, -0.8, 0],
        [0, 0, 0, 1, 0, -0.75]]
beq2 = [0, 0]
pot2 = lp(c = c2, A_ub = A2, b_ub = b2, A_eq = Aeq2, b_eq = beq2, method = 'highs')
x2 = pot2.x
val2 = -pot2.fun
print("The best choice of X are {0}, the max value of the func is {1}".format(x2, val2))
```

```
The best choice of X are [  0.  168.   19.2  -0.   24.    0. ], the max value of the func is
3460.8
```

Exercise 3

For this part, we need to solve the problem integer programming.

- Integer programming
- 0-1 integer programming
- mixed integer programming

In [ ]:

```
prob3 = pulp.LpProblem("Integer_programming_problem", pulp.LpMaximize)

x1 = pulp.LpVariable('x1', lowBound = 0, cat = 'Continuous')
x2 = pulp.LpVariable('x2', lowBound = 0, cat = 'Integer')
x3 = pulp.LpVariable('x3', lowBound = 0, cat = 'Continuous')

prob3 += 2 * x1 + 3 * x2 + 4 * x3

prob3 += 1.5 * x1 + 3 * x2 + 5 * x3 <= 600
prob3 += 280 * x1 + 250 * x2 + 400 * x3 <= 60000

prob3.solve()

print(prob3.name)
print(prob3.objective)
print(prob3.variables())
print("Maximization Results:")
for variable in prob3.variables():
    print(variable.name, "=", variable.varValue)
print("Total Maximization: ", pulp.value(prob3.objective))
```

```
Integer_programming_problem
2*x1 + 3*x2 + 4*x3
[x1, x2, x3]
Maximization Results:
x1 = 64.0
x2 = 168.0
x3 = 0.0
Total Maximization:  632.0
```

In [ ]:

```
prob4 = pulp.LpProblem("0-1-Integer-programming", pulp.LpMaximize)


x1 = pulp.LpVariable('x1', lowBound = 0, cat = 'Integer')
x2 = pulp.LpVariable('x2', lowBound = 0, cat = 'Integer')
x3 = pulp.LpVariable('x3', lowBound = 0, cat = 'Integer')
y1 = pulp.LpVariable('y1', lowBound = 0, upBound = 1, cat = 'Integer')
y2 = pulp.LpVariable('y2', lowBound = 0, upBound = 1, cat = 'Integer')
y3 = pulp.LpVariable('y3', lowBound = 0, upBound = 1, cat = 'Integer')

prob4 += 2 * x1 + 3 * x2 + 4 * x3

M = 1000

prob4 += x1 >= 80 * y1
prob4 += x2 >= 80 * y2
prob4 += x3 >= 80 * y3
prob4 += x1 <= M * y1
prob4 += x2 <= M * y2
prob4 += x3 <= M * y3
prob4 += 1.5 * x1 + 3 * x2 + 5 * x3 <= 600
prob4 += 280 * x1 + 250 * x2 + 400 * x3 <= 60000

prob4.solve()

print(prob4.name)
print(prob4.objective)
print(prob4.variables())
for var in prob4.variables():
    print(var.name, "=", var.varValue)
print("Total Maximization: ", pulp.value(prob4.objective))
```

```
0-1-Integer-programming
2*x1 + 3*x2 + 4*x3
[x1, x2, x3, y1, y2, y3]
x1 = 80.0
x2 = 150.0
x3 = 0.0
y1 = 1.0
y2 = 1.0
y3 = 0.0
Total Maximization:  610.0

2*x1 + 3*x2 + 4*x3
[x1, x2, x3, y1, y2, y3]
x1 = 80.0
x2 = 150.0
x3 = 0.0
y1 = 1.0
y2 = 1.0
y3 = 0.0
Total Maximization:  610.0
```

Exercise 4

This part is for the assignment decision! we should focus on how the variable's matrix is defined, and finally use the `pulp` for solving

In [ ]:

```
rows = 5
n = 20

prob5 = pulp.LpProblem('assignment_solution', pulp.LpMinimize)

#define the target variables and grades
x = [[pulp.LpVariable(f'x_{i}_{j}', lowBound = 0, upBound = 1, cat = "Integer") for i in rang
grades = [[66.8, 75.6, 87, 58.6],
          [57.2, 66, 66.4, 53],
          [78, 67.8, 84.6, 59.4],
          [70, 74.2, 69.6, 57.2],
          [67.4, 71, 83.8, 62.4]]

## define the target function we wanna solve
tmp = x[0][0] * grades[0][0]
for i in range(rows):
    for j in range(n // rows):
        if i == 0 and j == 0:
            continue
        tmp += x[i][j] * grades[i][j]
prob5 += tmp

for j in range(n // rows):
    tmp = x[0][j]
    for i in range(1, rows):
        tmp += x[i][j]
    prob5 += tmp == 1
for i in range(rows):
    tmp = x[i][0]
    for j in range(1, n // rows):
        tmp += x[i][j]
    prob5 += tmp <= 1

status = prob5.solve()
print(status)
print(prob5.name)
print(prob5.objective)
print(prob5.variables())
```

```
for var in prob5.variables():
    print(var.name, "=", var.varValue)
print("Total Maximization: ", pulp.value(prob5.objective))
```

```
1
assignment_solution
66.8*x_0_0 + 57.2*x_0_1 + 78*x_0_2 + 70*x_0_3 + 67.4*x_0_4 + 75.6*x_1_0 + 66*x_1_1 + 67.8*x_1
_2 + 74.2*x_1_3 + 71*x_1_4 + 87*x_2_0 + 66.4*x_2_1 + 84.6*x_2_2 + 69.6*x_2_3 + 83.8*x_2_4 + 5
8.6*x_3_0 + 53*x_3_1 + 59.4*x_3_2 + 57.2*x_3_3 + 62.4*x_3_4
[x_0_0, x_0_1, x_0_2, x_0_3, x_0_4, x_1_0, x_1_1, x_1_2, x_1_3, x_1_4, x_2_0, x_2_1, x_2_2, x
_2_3, x_2_4, x_3_0, x_3_1, x_3_2, x_3_3, x_3_4]
x_0_0 = 0.0
x_0_1 = 1.0
x_0_2 = 0.0
x_0_3 = 0.0
x_0_4 = 0.0
x_1_0 = 0.0
x_1_1 = 0.0
x_1_2 = 1.0
x_1_3 = 0.0
x_1_4 = 0.0
x_2_0 = 0.0
x_2_1 = 0.0
x_2_2 = 0.0
x_2_3 = 1.0
x_2_4 = 0.0
x_3_0 = 1.0
x_3_1 = 0.0
x_3_2 = 0.0
x_3_3 = 0.0
x_3_4 = 0.0
Total Maximization:  253.20000000000002
```

Exercise 5

enumerate every case and then set them as a variable and convert the question into the discrete linear programming problem.