작업관리 REPORT

-COMSOAL 프로그래밍-

이상주

먼저 저희가 프로그래밍한 코드를 처음부터 조금씩 설명하겠습니다.

```
import pandas as pd
import random as ran
```

저희는 선후관계도를 Excel파일에 저장하여서 불러오기로 하였습니다.

Python이 Excel파일을 불러올 방법을 찾던 중에 pandas라는 Tool이 있다는 것을 찾았고 이를 적용하여 pandas를 pd라는 이름으로 호출할 수 있도록 import를 활용해 Tool을 활성화시켰습니다.

random이라고 하는 Tool은 random이라는 Tool에 choice라고 하는 함수를 활용하여 COMSOAL을 하기 위해 후보작업들 중에서 하나의 값을 랜덤하게 추출하기 위해서 import를 하였습니다.

```
df = pd.read_excel('linebalancing.xlsx')

ct= int(input('Cycle Time :''분'))
iter=int(input('반복횟수:''회'))
```

위에서 import한 pandas Tool을 활용하여 같은 폴더에 저장되어 있는 'linebalancing.xlsx' 라는 이름의 Excel파일을 읽어 들여왔고, Cycle Time과 우리가 COMSOAL을 몇 번 반복할 지에 대한 수치를 입력하도록 프로그래밍하였습니다.

```
for i in range(iter):
```

이부분은 COMSOAL을 원하는 수치만큼 반복할 수 있도록, 반복할 횟수를 저장한 'iter' 라는 변수에 저장된 값만큼 반복하라는 의미를 가진 함수입니다.

```
workEleList = []
for i in range(len(df)):
    workEleList.append(df['요소작업'][i])

preTaskList = []
for i in range(len(df)):
    if type(df['선행작업'][i]) == int:
        preTaskList.append([df['선행작업'][i]])
    else:
        preTaskList += [df['선행작업'][i].split(',')]
preTaskList2 = preTaskList.copy()

timeEleList = []
for i in range(len(df)):
    timeEleList.append(df['작업시간(분)'][i])
```

위 부분은 Excel파일을 통해 읽어들인 선후관계도에 입력된 요소작업들과 그에 대한 작업시간, 선행작업의 정보를 별도의 리스트로 저장하는 부분입니다.

```
area=[[]]
areatime=[[]]
cur=0
sum_areatime=0
end_workEleList = []
for i in range(len(df)):
    end_workEleList+='X'
```

area는 작업장들의 리스트이며, [[A,B,C],[D,E,F]] 와 같은 형식으로 표현이 되고 []속에 있는 []이 하나의 작업장을 의미하며 여기에 적힌 순서대로 작업이 진행됩니다. areatime은 작업장에 투입된 요소작업의 작업시간에 대한 리스트입니다. Area의 리스트와 마찬가지로 []속에 있는 []이 투입된 요소작업의 작업시간을 작업순서대로 나열한 것입니다. cur는 작업장의 번호로서 0부터 시작합니다. sum_areatime은 현재 작업장에 투입된 요소작업들의 작업시간의 합을 의미하며, 아직 투입된 작업이 없기 때문에 0으로 시작합니다. 아래의 end_workEleList는 COMSOAL에 의한 요소작업의 투입이 끝난 후의 요소작업 리스트의 형태입니다. 저희는 투입한 요소작업을 'X'라는 값으로 치환할 것 이기 때문에 투입할 요소작업의 개수만큼의 'X'로 이루어진 리스트를 end_workEleList로 저장합니다.

```
while workEleList != end_workEleList:
```

while이라고 하는 함수는 다음에 나오는 조건을 만족한다면 아래에 이어지는 연산을 계속해서 반복하는 함수입니다. 이를 통해 workEleList에 저장된 리스트의 요소작업들을 모두 투입하여 모든 값들이 'X'가 되어 end_workEleList라는 변수에 저장된 리스트와 같아질 때 까지 작업들을 투입할 것입니다.

```
cE = []
if sum_areatime < ct :
    for j in range(len(df)):
        if preTaskList2[j][0] == 0 and sum(areatime[cur])+timeEleList[j] <= ct and workEleList[j] !='X':
            cE.append(workEleList[j])
```

먼저, 후보작업리스트 cE를 비어있는 리스트로 생성합니다.

if 라는 함수다음에 나오는 sum_areatime < ct 라는 것은 현재의 작업장에 투입된 요소작업들의 작업시간 합이 Cycle Time보다 작을 경우라는 조건을 건 것입니다. 이 조건을 만족한다면 바로 아래의 연산을 실행합니다. 여기서는 for를 활용한 반복문을 통해 각 요소작업들의 선행작업들을 조사합니다.

선행작업의 리스트에 0이라는 값만 들어있다면, 그것은 선행작업이 없다는 것을 의미합니다. 이를 만족하면서 그 작업을 투입했을 때의 작업시간 합이 Cycle Time을 넘지 않고, 이전에 사용된 요소작업이 아니라면 후보작업리스트 cE에 투입합니다.

이 작업을 모든 요소작업들이 거치면 조건에 맞는 후보작업들이 입력된 리스트를 얻을 수 있습니다.

```
if cE != []:
    cho = ran.choice(cE)
    area[cur] += [cho]
    areatime[cur] += [timeEleList[cho-1]]

    for i in range(len(df)):
        if workEleList[i] == cho:
            for n in range(len(df)):
                if workEleList[i] == preTaskList2[n][0]:
                    preTaskList2[n][0] = 0
                if str(workEleList[i]) in preTaskList2[n]:
                    del preTaskList2[n][preTaskList2[n].index(str(workEleList[i]))]
                if preTaskList2[n] == []:
                    preTaskList2[n] =[0]
            preTaskList2[i] = 'X'
    workEleList[cho-1]='X'
    sum_areatime = sum(areatime[cur])
```

이렇게 생성된 cE라는 후보작업 리스트에 하나라도 값이 입력되어 있다면, random Tool의 choice함수를 활용하여 하나의 작업을 랜덤하게 뽑아 cho라는 변수에 저장하고, 작업장에 그 작업과 작업의 작업시간을 추가합니다.

그리고 요소작업리스트(workEleList)에 저장된 작업중 랜덤하게 선정된 작업이 저장된 위치를 알아내어 그 위치에 존재하는 값을 선행작업으로 가진 작업들을 찾아내어 선행작업 리스트에서 선정된 작업을 지우고, 그 리스트가 비게 되었다면 선행작업이 없다는 의미로 [0]으로 만들어줍니다.

투입된 작업은 요소작업리스트와 선행작업리스트에서 'X'라는 형태로 표현되도록 해준 다음, 현재의 작업장에 투입된 작업들의 작업시간합을 sum_areatime에 저장해줍니다.

```
while workEleList != end_workEleList:
```

작업을 투입한 뒤에 다시 while로 돌아가면 이제 하나의 작업을 투입하였기에 모든 작업이 다 투입되기에는 아직 멀었음을 알 수 있습니다. 따라서 계속해서 위의 과정을 Cycle Time을 넘지않으면서 작업을 투입할 수 있는 한 계속해서 진행합니다.

```
else:
    sum_areatime += ct
```

else라고 하는 함수는 if함수 다음에 나오게 되는 함수로 if함수에서 내걸은 조건을 만족하지 않으면 else 다음의 연산을 실행하라는 의미입니다.

여기서 조건을 만족하지 않는다는 것은 cE에 후보작업이 하나도 저장되지 않았다는 것을 의미합니다. 이는 현재 설정된 Cycle Time을 만족하면서 투입할 수 있는 작업이 없다는 뜻이기 때문에 새로운 작업장이 필요합니다.

Sum_areatime에 Cycle Time만큼의 값을 더해줌으로써 아직 Cycle Time을 꽉 채우지는 못했지만 강제적으로 새로운 작업장을 형성하는 조건을 만족시켜줍니다.

이 부분에 의해 Cycle Time을 넘지 않으면서 작업을 투입할 수 없게 되었을 때 제일 처음 나왔던 if문의 조건인 작업시간의 합이 Cycle Tme보다 커졌으므로 이 if문의 else 함수인 아래의 함수로 넘어가게 됩니다.

```
else:
    cur+=1
    sum_areatime=0
    areatime+=[[]]
    area+=[[]]
```

이 부분의 함수는 더 이상 Cycle Time을 넘지 않게 작업을 투입할 수 없기 때문에 새로운 작업장을 형성하는 함수입니다.

cur에 저장된 작업장번호를 하나 더해줌으로써 이제는 작업이 투입될 작업장을 작업장0번에서 작업장1번으로 변경합니다. 그리고, 현재의 작업장의 작업시간합을 0으로 초기화하고 새로운 작업장과 작업시간 리스트를 생성합니다.

```
while workEleList != end_workEleList:
cE = []
if sum_areatime < ct :
    for j in range(len(df)):
        if preTaskList2[j][0] == 0 and sum(areatime[cur])+timeEleList[j] <= ct and workEleList[j] !='X':
            cE.append(workEleList[j])
```

그리고 다시 while로 돌아가 조건이 맞는지를 확인합니다. 아직 모든 작업이 투입되지 않았기에 지금까지의 과정을 다시 진행합니다.

다음이 저희가 프로그래밍한 파이썬 코드입니다.

```
import pandas as pd
import random as ran
import numpy as py

df = pd.read_excel('linebalancing.xlsx')

ct= int(input('Cycle Time :''분'))
iter=int(input('반복횟수:''회'))

for i in range(iter):

    workEleList = []
    for i in range(len(df)):
        workEleList.append(df['요소작업'][i])

    preTaskList = []
    for i in range(len(df)):
        if type(df['선행작업'][i]) == int:
            preTaskList.append([df['선행작업'][i]])
        else:
            preTaskList += [df['선행작업'][i].split(',')]
    preTaskList2 = preTaskList.copy()

    timeEleList = []
    for i in range(len(df)):
        timeEleList.append(df['작업시간(분)'][i])

    area=[[]]
    areatime=[[]]
    cur=0
    sum_areatime=0
    end_workEleList = []
    for i in range(len(df)):
        end_workEleList+='X'


    while workEleList != end_workEleList:
        cE = []
        if sum_areatime < ct :
            for j in range(len(df)):
                if preTaskList2[j][0] == 0 and sum(areatime[cur])+timeEleList[j] <= ct and workEleList[j] !='X':
                    cE.append(workEleList[j])

            if cE != []:
                cho = ran.choice(cE)
                area[cur] += [cho]
                areatime[cur] += [timeEleList[cho-1]]

                for i in range(len(df)):
                    if workEleList[i] == cho:
                        for n in range(len(df)):
                            if workEleList[i] == preTaskList2[n][0]:
                                preTaskList2[n][0] = 0
                            if str(workEleList[i]) in preTaskList2[n]:
                                del preTaskList2[n][preTaskList2[n].index(str(workEleList[i]))]
                            if preTaskList2[n] == []:
                                preTaskList2[n] =[0]
                        preTaskList2[i] = 'X'
                workEleList[cho-1]='X'
                sum_areatime = sum(areatime[cur])
            else:
                sum_areatime += ct
        else:
            cur+=1
            sum_areatime=0
            areatime+=[[]]
            area+=[[]]

    sum_totaltime=0
    sum_areatime=[]
    for i in range(len(areatime)):
        sum_totaltime+=sum(areatime[i])
        sum_areatime += [sum(areatime[i])]
    print('   ')
    print('작업장개수 :', len(area))
    print('작업장 :', area)
    print('작업시간 :', areatime)
    print('작업장별 소요시간 :',sum_areatime)
    print('작업효율 :' ,sum_totaltime/(len(area)*ct) )
```

	A	B	C	D	E	F	G
1	요소작업	작업시간(분)	선행작업				
2	1	7	0				
3	2	5	0				
4	3	8	0				
5	4	10	1				
6	5	7	1				
7	6	4	2				
8	7	3	3				
9	8	6	3				
10	9	10	4				
11	10	7	5,6				
12	11	6	8				
13	12	5	7,10				
14	13	5	12				
15	14	4	13				
16	15	12	9,11,14				
17	16	10	15				
18	17	5	16				
19	18	15	16				
20	19	10	16				
21	20	5	17				
22	21	6	18,19,20				
23							

이전 과제로 나왔던 선후관계도입니다.

이 Excel파일을 이용해 CT=30분, 반복횟수=5회 로 설정하여 도출된 결과는 다음과 같습니다.

```
Cycle Time :분 30
반복횟수:회 5

작업장개수 : 6
작업장 : [[1, 5, 2, 4], [3, 6, 10, 7, 8], [11, 9, 12, 13, 14], [15, 16, 17], [18, 20, 19], [21]]
작업시간 : [[7, 7, 5, 10], [8, 4, 7, 3, 6], [6, 10, 5, 5, 4], [12, 10, 5], [15, 5, 10], [6]]
작업장별 소요시간 : [29, 28, 30, 27, 30, 6]
작업효율 : 0.8333333333333334

작업장개수 : 6
작업장 : [[2, 1, 3, 8, 7], [11, 4, 6, 9], [5, 10, 12, 13, 14], [15, 16, 17], [19, 20, 18], [21]]
작업시간 : [[5, 7, 8, 6, 3], [6, 10, 4, 10], [7, 7, 5, 5, 4], [12, 10, 5], [10, 5, 15], [6]]
작업장별 소요시간 : [29, 30, 28, 27, 30, 6]
작업효율 : 0.8333333333333334

작업장개수 : 6
작업장 : [[3, 1, 7, 4], [8, 11, 9, 2], [5, 6, 10, 12, 13], [14, 15, 16], [17, 19, 18], [20, 21]]
작업시간 : [[8, 7, 3, 10], [6, 6, 10, 5], [7, 4, 7, 5, 5], [4, 12, 10], [5, 10, 15], [5, 6]]
작업장별 소요시간 : [28, 27, 28, 26, 30, 11]
작업효율 : 0.8333333333333334

작업장개수 : 6
작업장 : [[1, 2, 3, 4], [7, 5, 9, 6, 8], [11, 10, 12, 13, 14], [15, 16, 17], [18, 19, 20], [21]]
작업시간 : [[7, 5, 8, 10], [3, 7, 10, 4, 6], [6, 7, 5, 5, 4], [12, 10, 5], [15, 10, 5], [6]]
작업장별 소요시간 : [30, 30, 27, 27, 30, 6]
작업효율 : 0.8333333333333334

작업장개수 : 6
작업장 : [[3, 2, 8, 11, 6], [7, 1, 5, 4], [9, 10, 12, 13], [14, 15, 16], [18, 17, 19], [20, 21]]
작업시간 : [[8, 5, 6, 6, 4], [3, 7, 7, 10], [10, 7, 5, 5], [4, 12, 10], [15, 5, 10], [5, 6]]
작업장별 소요시간 : [29, 27, 27, 26, 30, 11]
작업효율 : 0.8333333333333334
```