Software For Finance Project

- 가계부 프로그램 -

§ 목 차

1. 프로그램 구상 배경

일상생활에서 수입과 지출을 관리하는 것은 매우 중요한 일이다. 기업이든 개인이든 수입과 지출 등 자산의 흐름을 기록하고 분석하는 것은 자산 관리의 중요한 단계로 여겨지고 있다. 그 예로 기업들이 작성하는 재무제표가 이렇게 자산의 흐름을 기록하는 역할을 한다고 할 수 있다. 소비자 금융 측면에서 이러한 역할을 하는 것이 바로 가계부이다. 지금까지 가계부는 개개인의 자금 유통을 관리하는 데 있어서 중요한 수단이 되어왔다.

나 역시 모든 수입과 지출, 저축 등의 현금 흐름 내역을 가계부에 전부 입력하고 있다. 그리고 평소에 가계부를 기록하면서 과소비와 충동구매 등 좋지 않은 소비 습관을 개선할 수 있도록 도와주는 프로그램을 만들 수 있지 않을까? 하는 생각을 해왔다. 또한, 이번 프로젝트를 계획하면서 이것을 단순히 과제를 수행하는 것이 아니라, 나에게 실질적으로 유익하고 필요한 프로그램을 직접 만들어보는 기회로 만들고 싶었다. 그래서 지금까지 배운 Python 언어들을 다양하게 사용해 보고 익히는 기회로 삼고자, 평소에 만들어보고 싶었던 가계부 프로그램을 구현하는 것을 주제로 결정하였다. 이 프로그램은 우선 기본적인 가계부 기능을 수행하는 것과, 만약 지출이 사용 가능 자산(이 프로그램에서는 수입의 총합이라고 표현하였다.)의 50%를 초과할 경우 경고 메시지를 출력하는 기능이 있다.

2. 프로그램 code 설명

1) 알고리즘

코드의 주요 진행 방식은 다음과 같다.

‘ ①필요 모듈 import > ②프로그램 실행 시 사용할 함수들 정의 > ③프로그램 실행메뉴’

2) code 설명

① 필요 module import

이제부터 코드에 대해 구체적으로 설명하도록 하겠다.

```
1  #필요한 모듈들을 import
2  import numpy as np
3  import pandas as pd
4  import os
5  import numpy as np
6  import matplotlib.pyplot as plt
```

먼저 필요한 모듈들을 import 한다.

pandas는 프로그램을 돌리는데 핵심이 되는 데이터 모듈이다. os는 저장될 파일을 불러오기 위해 사용할 것이며, matplotlib.pyplot은 후에 저장된 내역을 분석하여 그래프로 표현할 때 사용할 것이다.

```
8  # 파일을 불러올 때 사용하기 위해서 address에 cwd를 저장한다.
9  address = os.getcwd()
```

후에 저장된 파일을 불러올 때 어느 컴퓨터에서나 주소를 불러올 수 있도록 address에 cwd

를 저장한다.

② 사용할 함수들 정의하기

-1) writeFile()

이 함수는 사용자가 파일을 새로 생성할 때 사용한다. 함수가 시행되면 총 두 개의 파일이 생성된다. 첫 번째는 가계부 내역을 입력하고, 출력하고, 데이터를 기록하는데 주로 사용할 main file이며, 다른 하나는 사용자가 저축 기능을 사용할 경우 실행된다.

```
11 # writeFile() 함수 정의 (파일을 생성할 때 사용) ----------
12 #가계부 항목들을 columns으로 가진 dataframe을 생성하여 csv파일로 저장한다.
13 def writeFile():
14
15     print('₩n----------₩nPlease enter a
16     year=str(input('YEAR(XXXX) : '))
17     month = str(input('MONTH(YY) : '))
18     day = str(input('DAY(ZZ) : '))
19     today = (year+' - '+month+' - '+day)
20     pre_balance=int(input('Enter the balance you currently have.: '))
21     first_row = [today,'','','','','',pre_balance]
22     new_frameFile = pd.DataFrame([first_row],columns = ['date','sort','category','detail','amount','balance
23     new_frameFile.to_csv(user_name+'₩'s_File.csv', index=False)
24
```

처음 11줄의 code는 main file을 만드는 과정이다. 먼저 파일을 만드는 날짜와 현재 가지고 있는 기조 잔액을 입력받는다. 그럼 'date','sort','category','detail','amount','balance' 총 6개의 항목을 column으로 가진 데이터프레임이 생성되고, user name+ '\'s_File.csv'라는 이름으로 저장된다.

```
    # 동시에 사용자가 saving하는 자산들을 기록할 파일을 따로 생성하여 역시 csv파일로 저장한다.
    # saving파일의 기조잔액은 0원이다.
    first_row_saving = [today,'','','','',0]
    new_file_saving = pd.DataFrame([first_row_saving],columns=['date','category','detail','amount','balance'])
    new_file_saving.to_csv(user_name+'₩'s_File_saving.csv', index=False)

    new_frameFile
    print(user_name+"'s File created.")
```

그 후엔 입력된 정보를 바탕으로 동일한 날짜와 기조 잔액 0원을 가진 저축 파일이 생성되어 " user_name+'\'s_File_saving.csv'" 이라는 이름으로 저장된다. 마지막에 print되는 것은 파일을 생성한 후에 생성된 파일을 보여주고 파일 생성이 완료되었다는 메시지이다.

-2) record()

```
40 # record()함수 정의(사용자가 내역 입력을 사용할 경우 실행)----------
41 def record():
42     global data
43     new_frameFile = pd.read_csv(address+'₩₩'+user_name+'₩'s_File.csv', engine = 'python', index_col=False)
44
```

record() 함수는 사용자가 사용 기능 선택 창에서 write기능(내역 입력)을 선택할 경우 실행된다. 먼저 모든 함수들은 저장된 파일을 읽는 변수를 지정하는 것으로 시작한다. 저장된 파일을 불러온 후 pd.read.csv()로 파일을 읽도록 만들어 변수에 저장한다.

```
    #날짜 형식이 통일되도록 > 년, 월, 일 각각 숫자로 입력받아 통일된 형식으로 저장
    print('₩n--------------------------------------------------------------------------------₩nPlease enter a
    year=str(input('YEAR(XXXX) : '))
    month = str(input('MONTH(YY) : '))
    day = str(input('DAY(ZZ) : '))
    today = (year+' - '+month+' - '+day)

    # 데이터에 입력될 것들을 '항목_set'의 변수 이름으로 저장한다.
    # sort의 종류에 따라 내역기입의 실행방식이 달라져야 한다.
    # 따라서 sort_in에 숫자로 선택하도록 한다.
    print('[1. Income  2.Spending  3.Saving ]')
    sort_in = int(input('Please enter types of usage.(choose the number) : '))
    # sort_in이 무엇인지에 따라서 category에 입력할 내용이 달라진다. 각 경우별 category에 입력할 내용의 예시
    if sort_in == 1:
        print('₩nex) Pocket money, Salary, Financial income, Business profits, Rental income, etc₩n')
    if sort_in == 2:
        print('₩nex) Food, Transportation fee, Clothing, Cultural expense, Necessaries, Beauty, Educational
    if sort_in == 3:
        print('Please enter your account to save.₩n( ex)KB Bank )₩n ')
```

그 후엔 날짜를 입력받은 후 해당 내역이 수입, 지출, 저축 중 어디에 해당하는지 선택받아 sort_in이라는 변수에 저장한다. 그리고 경우에 따라 category에 들어갈 항목이 다르기 때문에 if문을 사용하여 각 경우 catefory에 들어갈 내용을 예시로 출력해준다.

```
    # 나머지 항목들을 마저 입력받는다.
    category_set = input('Please enter the category of the input. : ')
    detail_set = input('Enter the details. : ')
    amount_set = int(input('Enter the amount. : '))
    #balance cell 기존 데이터와 합쳐진 후 계산된 값이 저장될 것이므로, 일단 여기서는 blank string 저장
    balance_set = ''

    # 위에서 사용자가 선택한 sort_in에 해당하는 것을 str으로 입력되도록 저장.
    if sort_in == 1:
        sort_set = ('Income')
    elif sort_in == 2:
        amount_set = -(amount_set)
        sort_set = ('Spending')
    elif sort_in == 3:
        amount_set = -(amount_set)
        sort_set = ('Saving')
```

그 후 나머지 항목들에 들어갈 내용을 '항목_set'이라는 변수들에 input으로 저장 받는다. 입력해야 하는 내용에 대한 설명은 input 명령어에 입력을 받을 때 출력하여 알려준다.
그리고 아까 선택한 sort_in에 따라서 내역에 income, spending, saving이라는 string으로 입력되도록 if 문을 사용하였다.

```
    elif sort_in == 3:
        amount_set = -(amount_set)
        sort_set = ('Saving')
        # 사용자가 saving을 한 경우에는 전에 만들어둔 saving 용 파일에도 내역을 따로 저장해야한다.
        # 밑에 내역 기록하는 것과같은 과정을 saving 파일 내에서도 동일하게 실행한다.(과정은 밑과 같으므로 ㅊ
        new_file_saving = pd.read_csv(address+'₩₩'+user_name+'₩'s_File_saving.csv', engine = 'python', inde

        new_row_saving = [today,category_set,detail_set,amount_set,balance_set]
        record_saving = pd.DataFrame([new_row_saving],columns = ['date','category','detail','amount','balan

        combi_saving = new_file_saving.append(record_saving, ignore_index = True)

        combi_saving['amount'][len(combi_saving)-1]= - (combi_saving['amount'][len(combi_saving)-1])
        balance_cal_saving = int(combi_saving['balance'][len(combi_saving)-2])+int(combi_saving['amount'][l
        combi_saving['balance'][len(combi_saving)-1] = balance_cal_saving

        combi_saving.to_csv(user_name+'₩'s_File_saving.csv',index=False)
```

그리고 sort_in == 3, 즉 입력되는 내용의 종류가 saving일 경우 writeFile() 함수에서 생성했던 파일에 동시에 내역이 따로 입력되도록 한다. 즉 만약 항목이 저축이라면 해당 main data와 저축용 data에 동시에 입력되는 것이다.

위 코드는 먼저 저장용 파일을 읽어와서 new_file_saving에 저장하고, 새로 입력받은 내역을 new_row_saving이라는 dataframe으로 형성한다. 그 후 new_file_saving에 new_row_saving을 .append()를 이용하여 새로운 열이 추가되는 방식으로 만들고, combi_saving에 저장한다. 그 후 저축용 파일의 잔액을 계산하고, 그 dataframe을 원래 저축용 파일과 같은 이름으로 저장한다.(작업을 마친 파일을 이전 파일에 덮어쓰는 것이다.)

```
97    #기존 데이터에 새로 기록될 row의 내용들을 새로운 dataframe(record_File이라는 기록용 df)에 저장한다.
98    new_row = [today,sort_set,category_set,detail_set,amount_set,balance_set]
99    record_File = pd.DataFrame([new_row],columns = ['date','sort','category','detail','amount','balance'])
100
101   #기존 data에 새로운 dataframe을 append하여 결과적으로, 새로운 내역을 기록할 때마다 row가 추가되는 형식으
102   combi = new_frameFile.append(record_File, ignore_index = True)
103
104   #내역이 추가된 후 잔액을 계산한다
105   balance_cal = int(combi['balance'][len(combi)-2])+int(combi['amount'][len(combi)-1])
106   combi['balance'][len(combi)-1] = balance_cal
107
108
109   #잔액 계산까지 마친 최종 파일을 기존 파일과 동일한 이름으로 저장한다(덮어쓴다).
110   combi.to_csv(user_name+'₩'s_File.csv', index=False)
111   data = pd.read_csv(user_name+'₩'s_File.csv')
```

다음은 동일한 작업을 main 함수에서 실행한다. 이 작업을 마친 후 이 dataframe 또한 원래 파일과 같은 이름으로 저장하여 덮어쓴다.

```
114   # 경고메시지 띄우기
115   # income,spending,saving 금액을 합할 변수들을 미리 지정한다.
116   # saving도 결국 사용가능자산에서는 빠져나가는 것이므로 withdrawal에 지출과 같이 합한다.
117   sum_usable = float(data['balance'][0])
118   sum_withdrawal = 0
119
120   #for문을 이용하여 데이터의 처음부터 끝까지 각 sort별 합을 구해 위에서 설정한 변수들에 저장한다.
121   for a in range(1,len(data)):
122       if data['sort'][a] == 'Income':
123           #sum_usable 은 사용가능자산들로 시작잔액에 sort가 income인 row들의 합을 더한다.
124           sum_usable = float(sum_usable) + float(data['amount'][a])
125       elif data['sort'][a] == 'Spending' or 'Saving':
126           sum_withdrawal = float(sum_withdrawal) + float(data['amount'][a])
127
128   # 내역의 지출항목의 합이 수입항목의 50%를 초과하였으면 경고메시지를 출력한다.
129   if -(sum_withdrawal) > (sum_usable*(0.5)) :
130       print('₩n₩n**********※ Warning ※**********₩n')
131       print('Your expenses exceeded 50% of available assets.₩nYou should cut back on your expenses.₩n')
132       print('**********************************************₩n')
```

다음은 경고 메시지를 띄우는 코드이다. 먼저 sum_usable[1]에 기초 잔액을 저장하고, sum_withdrawal[2]에 0을 저장한다. 그 후 for 반복문으로 내역의 처음부터 끝까지 프로그램의 ['sort']에 따라 해당 변수에 더해지도록 하였다. 그리고 만약 sum_withdrawal(통장 -의 총합)[3]가 sum_usable의 50%을 초과할 경우 ※Warning※ 메시지를 출력한다.

-3) correcting()

1) 사용 가능 자산(수입의 총합으로 표현하였다).
2) 지출과 저축의 총합을 저장한다. 저축은 지출은 아니지만, 결국 사용 가능한 자산은 아니므로 withdrawal을 계산하는 것에 포함하였다.
3) 여기서 -(sum_withdrawal)인 이유는 내역을 저장할 때 sort가 지출과 저축인 경우 -를 붙여 저장하도록 되어 있기 때문이다. 금액 비교를 위해서는 -를 다시 붙여 비교해야 한다.

```
def correcting():
    global data

    # 수정하기 위해 기존 파일을 읽어온다.
    cort_data = pd.read_csv(user_name+'₩'s_File.csv')
    # 사용자가 무엇을 수정할지 선택해야 하므로, 기존 내역을 print해준다.
    print(cort_data)
```

correcting()은 사용자가 내역 수정 기능(2.correctig)을 선택할 경우 실행된다. 역시 함수 정의 line과 기존 저장된 main File을 불러오는 것에서 시작한다. 내역 수정은 사용자가 현재 저장된 내역을 보고 수정할 내역을 골라야 하므로 먼저 저장된 data를 출력하여 보여준다.

```
    # 수정할 내역의 index와 무슨 항목을 수정할지 선택받는다.
    cort_row = int(input('Choose the index number you want to correct. : '))
    cort_col = input('Choose the content you want to correct. : ₩n( date, sort, category, detail, amount, b
    # 사용자가 date항목을 수정할 경우 통일된 날짜 형식으로 저장되도록 한다.
    if cort_col == 'date':
        print('₩n----------------------------------------------------------------------------------------₩nPlease ente
        year=str(input('YEAR(XXXX) : '))
        month = str(input('MONTH(YY) : '))
        day = str(input('DAY(ZZ) : '))
        cort_data[cort_col][cort_row] = (year+' - '+month+' - '+day)
    else:
        # 나머지 항목들은 어떻게 수정하고 싶은지 물어보고 입력받는다.
        cort_data[cort_col][cort_row]=input( 'How do you want to change? ')
```

다음은 사용자가 저장된 내역을 보고 어느 내역을 수정할지 선택하도록 한다. 이때 index 번호를 입력하도록 한다. 만약 사용자가 date를 수정하도록 할 경우에는 통일된 형식으로 입력되도록 날짜를 입력할 때 사용하던 코드를 다시 사용하였다. 날짜가 아닌 경우 어떻게 바꾸고 싶은지(수정 후 내용) 물어보는 메시지를 출력한다.

```
    # 사용자가 amount나 balance를 수정할 경우 잔액을 재계산한다.
    if cort_col == 'amount' or 'balance':
        if cort_row == 0:
            for i in range(cort_row+1,len(cort_data)):
                cort_data['balance'][i]=int(cort_data['balance'][i-1])+int(cort_data['amount'][i])
        else:
            for i in range(cort_row,len(cort_data)):
                cort_data['balance'][i]=int(cort_data['balance'][i-1])+int(cort_data['amount'][i])

    # 수정을 마친 파일을 기존 파일과 동일한 이름으로 저장한다.(덮어쓴다)
    cort_data.to_csv(user_name+'₩'s_File.csv', index=False)
    data = pd.read_csv(user_name+'₩'s_File.csv')

    # 수정을 완료한 내역을 출력하여 보여준다.
    print('=============================================================================')
    print(data)
    print('=============================================================================₩n₩nComplete.')
```

또한, 만약 사용자가 amount(해당 내역의 입력 금액)이나 balance(잔액)을 수정할 경우 현재 저장되어 있는 잔액 계산이 맞지 않게 되므로 해당 row부터 잔액을 재계산한다.
만약 수정하려는 내역이 처음 잔액이면 [len(cort_data)-1]이 음수가 되어 존재하지 않는 index가 된다. 그럴 경우엔 Error가 발생하므로 if/else문으로 수정하려는 내역의 index가 0일 경우와 0이 아닐 경우로 나눠서 코드를 작성하였다.

모든 작업을 마치면 이 파일 역시 기존 파일과 동일한 이름으로 저장하여 덮어쓴다. 수정이 완료되면 수정 후 내역(가장 last data)을 print하여 보여준 후 작업이 완료되었다는 “complete.” 메시지를 출력하였다.

-4) remove()

```
def remove():
    # 내역 삭제를 위해서 del_data라는 변수로 기존 파일을 불러온다.
    del_data = pd.read_csv(user_name+'₩'s_File.csv')
    print(del_data)

    # 삭제할 내역을 입력받아 del_row에 저장한 후, .drop method를 이용하여 삭제
    del_row = int(input('Choose the index number you want to delete: '))
    del_data = del_data.drop(index = del_row)
```

기존 파일을 불러온 후 삭제할 내역을 고르기 위해 기존 내역을 print하는 것은 correcting() 함수와 동일한 과정이다. 역시 index number로 입력받고 해당 index를 .drop() method를 사용하여 삭제한다.

```
    # 중간 index가 빠져버렸으므로 index를 재정렬한다.
    del_data.index = pd.RangeIndex(len(del_data.index))
    del_data.index = range(len(del_data.index))

    # 중간 내역이 빠져 삭제된 내역부터 잔액이 맞지 않으므로, 잔액을 재계산한다.
    for i in range(1,len(del_data)):
        del_data['balance'][i]=int(del_data['balance'][i-1])+int(del_data['amount'][i])

```

내역을 삭제하면 중간 내역이 삭제되어 해당 index가 비어버리게 된다. 그렇기 때문에 index를 다시 재정렬하는 과정을 거친 후, balance 또한 맞지 않게 되었으므로 잔액을 재계산한다.

```
    # 삭제가 완료된 파일을 data에 저장하고, 삭제 완료된 내역을 print하여 보여준다.
    del_data.to_csv(user_name+'₩'s_File.csv', index=False)
    data = pd.read_csv(user_name+'₩'s_File.csv')

    print('================================================================================')
    print(data)
    print('================================================================================₩n₩nComplete.')
#-------------------------------------------------------------------------------------------
```

그 후 파일을 저장하고 출력하는 과정은 위의 함수와 같은 과정을 반복한다.

-5) analysis()

```
def analysis():
    data = pd.read_csv(user_name+'₩'s_File.csv')

    sum_income = 0
    sum_spending = 0
    sum_saving = 0

    #for문을 이용하여 데이터의 처음부터 끝까지 각 sort별 합을 구해 위에서 설정한 변수들에 저장한다.
    for a in range(1,len(data)):
        if data['sort'][a] == 'Income':
            #sum_usable 은 사용가능자산들로 시작잔액에 sort가 income인 row들의 합을 더한다.
            sum_income = 0 + float(data['amount'][a])
        elif data['sort'][a] == 'Spending':
            sum_spending = float(sum_spending) + float(data['amount'][a])
        elif data['sort'][a] == 'Saving':
            sum_saving = float(sum_saving)+float(data['amount'][a])
```

분석 함수는 현재 내역의 수입, 지출, 저축 항목의 합계와 비율을 알려주며, Pie chart로 출력하여 보여준다. 역시 저장된 데이터를 불러오는 것으로 시작한다. 먼저 sum_income, sum_spending, sum_saving에 0을 저장한다.

그 후 for 반복문을 이용해 main data 전체의 income, spending, saving의 합계를 구하여

sum_income, sum_spending, sum_saving에 저장한다.

```
234     # spending과 saving은 -로 입력되고 있으므로 통계를 위해서 부호를 바꿔서 저장한다.
235     sum_spending = -(sum_spending)
236     sum_saving = -(sum_saving)
237
238     sum_total = (sum_income)+(sum_spending)+(sum_saving)
239
240     # 각 항목의 차지 비율을 구한다.
241     ratio_income = (sum_income)/(sum_total)
242     ratio_spending = (sum_spending)/(sum_total)
243     ratio_saving = (sum_saving)/(sum_total)
244
```

그 후엔 sum_spending, sum_saving의 부호를 바꿔준다. 여기서 부호를 바꾸는 이유는 각주 1)에서의 이유와 같다. 내역 입력 시 -의 형태로 저장했기 때문에 비율을 계산하기 위해서 부호를 바꿔주어야 한다.

다음은 세 변수들의 총합을 이용하여, 각 항목들(income, spending, saving)의 비율을 계산한다.

```
245     # 각 항목의 차지 비율을 print하여 보여준다.
246     print('+======================< Total >======================+₩n')
247     print('                  income : '+str(sum_income)+' won ₩n')
248     print('               spending : '+str(sum_spending)+' won ₩n')
249     print('                 saving : '+str(sum_saving)+' won ₩n')
250     #그래프의 색, lable, 비율을 설정하고 pie차트로 프린트한다.
251     colors = ['lightskyblue','lightpink','gold']
252     labels = ['Income','Spending','Saving']
253     ratio = [ratio_income, ratio_spending, ratio_saving]
254     plt.pie(ratio, labels=labels,colors=colors, shadow=True, startangle=90)
255     plt.show()
```

마지막으로 각각의 비율을 출력하여 보여주고, Pie chart로 출력하여 나타낸다. 각 항목의 label은 income, spending, saving으로 설정하였고, 색상을 지정하였다.

③ 프로그램 실행 창

```
print('************************** Welcome ********************************₩n')

user_name = input('Please let me know your name. : ')
#파일을 생성할 것인가 불러올 것인가 사용자가 정하도록 한다.
```

실행하면 'Welcome' 메시지와 함께 사용자의 이름을 입력하는 창이 뜬다. 이 창은 파일 이름을 저장할 때 사용될 것이다. 후에 같은 사용자가 동일 파일을 이용하기 위해서는 처음 파일 생성 시에 입력한 이름과 같은 이름을 입력하여야 한다.[4)]

```
#사용자가 가계부 생성을 선택하면, writeFile() 함수를 실행
while True:
    mode_set = int(input('Choose the option.₩n₩n1. Create data       2. Continue yours₩n₩n'))
    if mode_set == 1:
        # 만약 사용자가 이미 파일이 있는데 1을 선택하였을 경우, 새로 생성하면 이전 파일이 삭제되므로 경고메시지를
        if os.path.exists(address+'₩₩'+user_name+'₩'s_File.csv')== True:
            print('---------------------------------< ₩arning >-----------------------------------₩n')
            print('Your file already exists. If you continue to this, The existing file will be deleted.₩nDo you
            print('-------------------------------------------------------------------------------')
            # 그래도 실행할 것인지 물어보고 대답이 yes면 새로 생성(이전파일지워짐)
            answer = int(input('1. Yes  2. No (Please choose the number.)'))
            if answer == 1:
                writeFile()
                print('   ** Done **₩n Enter < 2. Continue yours > to continue running this program.')
            # 대답이 No면 option선택창으로 돌아간다.
            if answer == 2:
                print('₩nYou chose ₩'No₩'.>> Return to option selection!')
```

4) 이전 자신의 가계부를 불러오는 ID와 같은 역할을 한다.

이어서 while True:문 속에서 데이터를 새로 생성할 것인지, 이전 것을 불러올 것인지 선택하는 option selection message가 뜬다. 여기서 1을 선택하면 새로운 파일을 생성하는 것이고, 2를 선택하면 이전 파일을 불러온다.

만약 이전에 프로그램을 사용한 적이 있어 기존 파일이 생성되어 있는 사용자가 '1.creat data'를 선택한다면 기존 파일이 지워지게 된다. 그래서 기존 파일이 삭제된다는 경고 메시지를 띄우고 파일 생성을 계속 진행할 것인지 묻는다. 그 대답을 answer에 저장하여 '1.yes'를 선택하면 기존 파일을 지우고 새 데이터를 생성, '2. No'를 선택하면 다시 option을 선택하는 창으로 되돌아간다. 기존 파일이 없다면 정상적으로 파일이 생성된다. 파일 생성 후엔 역시 option selection message로 돌아가는데 여기서 프로그램을 계속 진행하기 위해서는 '2.continue yours'를 선택하여야 한다.

```
    elif mode_set == 2:
        while True:
            #만약 사용자가 이어서쓰기를 선택하였으나, 기존 파일이 없는 경우 writeFile() 함수를 실행하여 파일을 생
            if os.path.exists(address+'₩₩'+user_name+'₩'s_File.csv')== False:
                print('₩n  ** Nope **₩nYour file does not exist .₩n₩n>> Create your File.₩n')
                writeFile()
            else:
                break
        using = 0
        while using==0:
            #기능 선택 메뉴
            print('+------------------------------+')
            print('       (1) Write       ')
            print('       (2) Correcting       ')
            print('       (3) Remove      ')
            print('       (4) Analysis        ')
            print('+------------------------------+')
```

다음은 사용자가 '2. continue yours'를 선택한 경우이다. 이전에 프로그램을 사용한 적이 있는 사용자가 이것을 선택한 경우 정상적으로 프로그램의 기능을 선택하는 메뉴 창이 뜬다. 하지만 만약 이전에 프로그램을 사용한 적이 없어서 기존 파일이 없는 사용자가 이 옵션을 선택한다면 파일이 없다는 메시지를 출력과 writeFile() 함수를 실행하여 파일을 생성한 후, 메뉴 선택 창이 뜬다.

```
        using = 0
        while using==0:
            #기능 선택 메뉴
            print('+------------------------------+')
            print('       (1) Write       ')
            print('       (2) Correcting       ')
            print('       (3) Remove      ')
            print('       (4) Analysis        ')
            print('+------------------------------+')

            #메뉴를 선택할 것을 번호로 입력하도록 input받는다. 그 외의 것을 선택하면 프로그램을 종료하는 것을 선
            print('₩nChoose the Number you want to use. ')
            print('If you want to EXIT, choose the other one. : ')
            choice = int(input())
```

기능 선택은 해당 숫자를 입력하는 방식으로 이루어진다. 만약 아무 기능을 사용하지 않고 프로그램을 종료하고 싶다면 1~4 이외의 다른 것을 입력하면 된다. 이때 string 타입을 입력해서는 안된다.[5)]

5) int()함수로 변환한 후 저장할 것이기 때문이다. int() 함수를 실행하지 못함

```
        #사용자가 작업을 선택하면 각 작업에 해당하는 함수를 실행한다.
        if choice == 1:
            record()
        elif choice == 2:
            correcting()
        elif choice ==3:
            remove()
        elif choice == 4:
            analysis()
        else:
            # 사용자가 종료를 선택하면(1,2,3,4) 외 다른 것 진짜 프로그램을 나갈 것인지 묻는다.
            print('₩nAre you sure you want to exit the program?')

        # 진짜 프로그램을 나갈 것인지 물은 후와
        # 가계부에서 작업을 마친 후엔, 프로그램을 계속할 것인지 물어본다. 0을 입력할 경우 기능 선택 메뉴가 ㅈ
        using=int(input('₩nEnter 0 to continue using the program, or the other to Exit.₩n₩n'))
    # 정말 나가기를 선택하면 프로그램이 종료된다.
    if using != 0:
        break
```

사용자가 기능을 선택하면, 각 기능을 수행하는 함수를 실행한다.[6] 함수 실행 후엔 프로그램을 계속 사용할 것인지 묻고, 계속 실행할 것이라면 0을 입력받도록 한다. 이때 0 이외의 것을 입력하면 프로그램이 종료된다. 0을 입력하면 다시 기능 선택 창으로 되돌아가서 계속해서 프로그램을 이용할 수 있도록 한다.

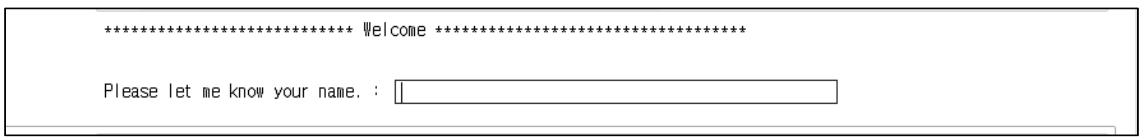
```
    else:
        print('Please choose 1 or 2.')#맨 처음 파일 선택 옵션에서 1이나 2를 선택하지 않았을 경우, 옵션 창으로 되


# 현재까지 한 작업 파일을 data에 읽기로 저장하며 파일 종료시 dataframe형태로 print하여 보여준다.
data = pd.read_csv(user_name+'₩'s_File.csv',engine='python')

print('₩n~~~~~~~~~~~~~~~~~~~~ <'+user_name+'₩'s Household Accounts > ~~~~~~~~~~~~~~~~~~~~')
data
```

맨 마지막의 else:는 처음 프로그램 실행 시의 옵션 선택 창에서 사용자가 1, 2 이외의 것을 입력하였을 경우 다시 옵션을 선택하도록 하는 line이다.

마지막으로 사용자가 프로그램을 종료하면 data로 저장된 파일을 읽을 수 있도록 한 후에 마지막으로 최종 내역을 출력하고 프로그램이 종료된다.

3. 프로그램 사용 설명서(tutorial)

① 프로그램 시작 > 사용자 이름 입력

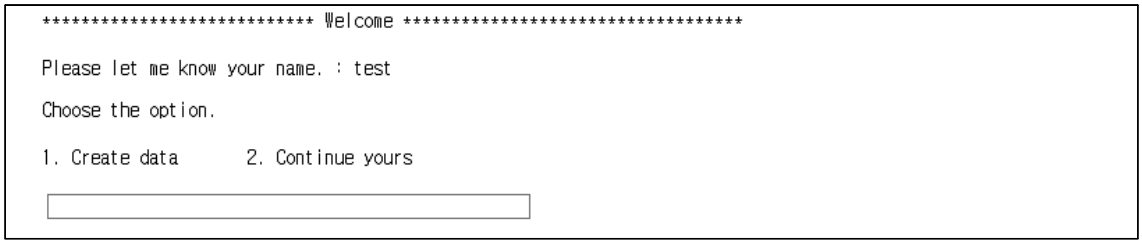


: 프로그램을 실행하면 사용자의 이름을 입력하는 창이 뜬다. 여기서 후에 데이터를 새로 생성하던, 이전 파일을 불러오던 사용자가 입력하는 이름의 파일을 불러오게 된다. 때문에, 파일 생성 후 이어서 쓰는 경우라면 파일을 생성했을 때 입력한 이름과 동일한 이름을 입력해야 한다. 즉, 여기서 입력하는 이름이 위에서 언급했듯이 가계부의 ID와 같은 역할을 하는 것이다. (tutorial에서는 예시로 ‘ test’라는 이름을 사용하겠다.)

---

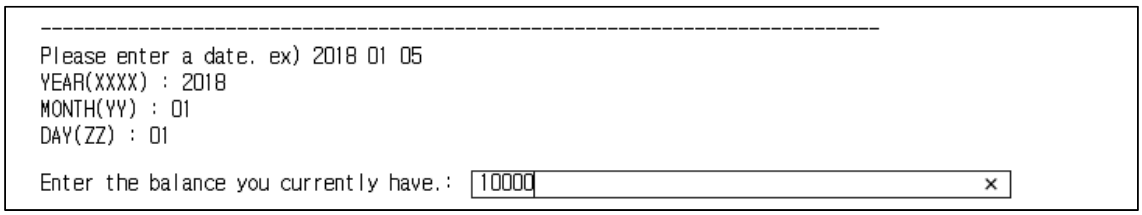
6) 여기서 각 함수들은 위에서 미리 정의해 놓았다.

② Option selection

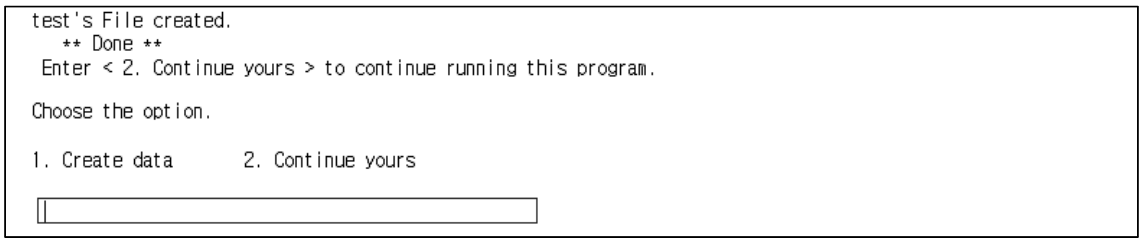


: 데이터를 새로 생성할 것인지, 이전 파일에 계속 입력할 것인지 선택한다.

-1) 1. Create data

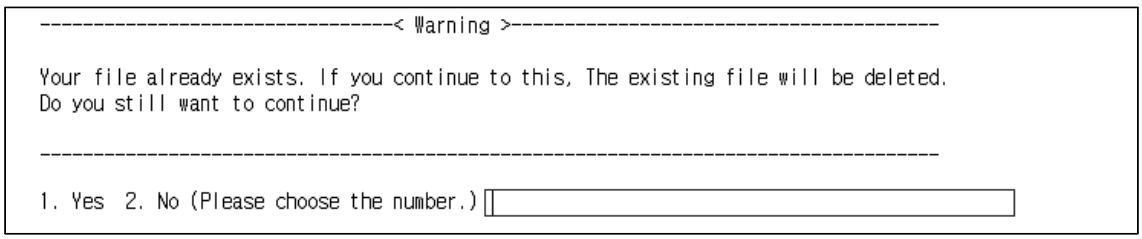


: 데이터 생성을 선택할 경우 날짜와 기조 잔액을 입력한다. 날짜를 입력할 때는 년, 월, 일을 따로 입력한다. 이때 년(YEAR)은 4자리, 월(MONTH)는 2자리, 일(DAY)는 2자리로 입력하는 것이 좋다.[7]



: 잔액 입력 후엔 다시 option 선택 창으로 되돌아간다. 여기서 프로그램을 계속 이용하려면 '2. Continue yours'를 선택해야 한다.

-1.1) 기존 파일이 존재하는데 '1. Create data'를 선택한 경우



: 만약 기존 데이터가 존재하는데 새로운 데이터 생성을 선택하면, 이전 파일이 지워지게 된다. 정말 이전 파일을 지우고 새로 생성할 것이면 1, 아니면 2를 입력한다. 이때 1(Yes)을 입력하면 데이터를 새로 생성하며[8], 2(No)를 입력하면 option 선택으로 되돌아간다.

---

7) 데이터가 통일된 형식으로 출력되기 때문이다.
8) -1) 의 과정이다

```
1. Yes  2. No (Please choose the number.)2

You chose 'No'.>> Return to option selection!

Choose the option.
```

-2) 2. Continue yours

```
2
+-----------------------------+
       (1) Write
       (2) Correcting
       (3) Remove
       (4) Analysis
+-----------------------------+

Choose the Number you want to use.
If you want to EXIT, choose the other one. :
```

: 이어서 쓰기를 선택하면 기능 선택 창이 뜬다. 이때 실행하고 싶은 기능을 번호로 입력하면 된다. 만약 프로그램을 종료하고 싶으면 1~4 이외의 것을 아무거나 입력하면 된다. 하지만 이 이외의 것은 숫자여야 한다.[9)]

-2.1) 이전에 프로그램을 사용한 적 없는 사용자가 2(Continue)를 선택한 경우

```
Please let me know your name. : Who
Choose the option.

1. Create data       2. Continue yours

2

  ** Nope **
Your file does not exist .

>> Create your File.


--------------------------------------------------------------------------------
Please enter a date. ex) 2018 01 05

YEAR(XXXX) :
```

: 파일이 존재하지 않는다는 메시지를 출력한 후 파일 생성으로 넘어간다. 파일 생성 후 기능 선택 창이 뜨면 프로그램을 진행하면 된다.

-3) Else

```
6
Please choose 1 or 2.

Choose the option.

1. Create data       2. Continue yours
```

: 만약 1이나 2를 선택하지 않으면, 1이나 2를 선택해달라는 메시지가 출력된 후 옵션 선택

9) 이것에 대해서는 각주 5에서 설명하였다.

창으로 되돌아간다.

③ 사용할 기능 선택

-1) 1. Write(내역 입력)

```
--------------------------------------------------------------------------------
Please enter a date. ex) 2018 01 05
YEAR(XXXX) : 2018
MONTH(YY) : 01
DAY(ZZ) : 03
[1. Income  2.Spending  3.Saving ]

Please enter types of usage.(choose the number) :
```

: 내역 입력은 파일을 생성할 때와 같이 날짜를 먼저 입력한다. 그 후엔 그 내역이 소비인지 지출인지, 저축인지 그 종류를 입력한다. (숫자로 입력)

```
ex) Pocket money, Salary, Financial income, Business profits, Rental income, etc

Please enter the category of the input. :
```

: 다음은 구체적 카테고리를 입력한다.
이때, [수입 > 용돈, 급여 사업수입, 금융수익, 임대수익 등등/ 지출 > 교통비, 식비, 생필품, 교육비, 의료비, 문화비 등등/ 저축 > 저축 통장] 을 입력하는 것이다.

```
Enter the details. : Mom
Enter the amount. : 5000
================================================================================
            date    sort      category detail amount  balance
0  2018 - 01 - 01       .             .      .      .    10000
1  2018 - 01 - 03  Income  Pocket money    Mom   5000    15000


Enter 0 to continue using the program, or the other to Exit.

```

: detail과 amount를 입력한다. detail은 내역에 대한 구체적 설명, amount는 금액을 입력하면 된다. 입력을 마치면 내역이 출력된다.[10] 내역 입력을 마치면 프로그램을 계속 사용할 것인지, 혹은 종료할 것인지 선택한다. 프로그램을 계속 사용하고 싶으면 0, 종료하고 싶으면 이외의 숫자를 입력하면 된다. 0을 선택하면 기능 선택 창으로 되돌아가며, 종료를 선택하면 프로그램이 종료된다.

-2) correcting (내역 수정)

```
            date      sort       category       detail amount  balance
0  2018 - 01 - 01         .              .            .      .    10000
1  2018 - 01 - 03    Income   Pocket money          Mom   5000    15000
2  2018 - 01 - 04    Income         Salary  PT job_cafe   7600    22600
3  2018 - 01 - 04  Spending           Food       coffee  -1500    21100
4  2018 - 01 - 04  Spending           Food        bread  -2000    19100
5  2018 - 01 - 05  Spending     Transp fee          Bus  -1300    17800
6  2018 - 01 - 06    Income  Financ income     interest   2200    20000
7  2018 - 01 - 07    Saving        KB Bank  installment  -5000    15000

Choose the index number you want to correct. :
```

10) 잔액('balance')은 자동으로 계산됨

: 내역 수정을 입력하면 다음과 같이 현재 저장되어있는 내역이 출력된다. 이 내역을 보고 수정하고 싶은 내역의 index number를 입력한다.

```
Choose the content you want to correct. :
( date, sort, category, detail, amount, balance ) amount
```

: 다음은 수정하고 싶은 내역의 항목을 선택한다. 예시와 같이 괄호 안에 있는 것을 그대로 입력해야 한다.

```
How do you want to change?
```

: 마지막으로 어떻게 수정하고 싶은지 입력하면 된다.

```
            date      sort       category       detail amount  balance
0  2018 - 01 - 01         .              .            .      .    10000
1  2018 - 01 - 03    Income   Pocket money          Mom   5000    15000
2  2018 - 01 - 04    Income         Salary  PT job_cafe   7600    22600
3  2018 - 01 - 04  Spending           Food       coffee  -1500    21100
4  2018 - 01 - 04  Spending           Food        bread  -2000    19100
5  2018 - 01 - 05  Spending     Transp fee          Bus  -1300    17800
6  2018 - 01 - 06    Income  Financ income     interest   2200    20000
7  2018 - 01 - 07    Saving        KB Bank  installment  -5000    15000
Choose the index number you want to correct. : 6
Choose the content you want to correct. :
( date, sort, category, detail, amount, balance )amount
How do you want to change? 1100
================================================================================
            date      sort       category       detail amount  balance
0  2018 - 01 - 01         .              .            .      .    10000
1  2018 - 01 - 03    Income   Pocket money          Mom   5000    15000
2  2018 - 01 - 04    Income         Salary  PT job_cafe   7600    22600
3  2018 - 01 - 04  Spending           Food       coffee  -1500    21100
4  2018 - 01 - 04  Spending           Food        bread  -2000    19100
5  2018 - 01 - 05  Spending     Transp fee          Bus  -1300    17800
6  2018 - 01 - 06    Income  Financ income     interest   1100    18900
7  2018 - 01 - 07    Saving        KB Bank  installment  -5000    13900
================================================================================

Complete.


Enter 0 to continue using the program, or the other to Exit.
```

: 수정을 완료하면 아래와 같이 수정 후 내역이 출력된다. 수정한 것이 바뀌어 입력되며, 잔액은 자동으로 재계산되어 입력된 것을 확인할 수 있다. 이때도 프로그램을 계속 사용할 것이면 0, 아니면 다른 숫자를 입력한다.

-3) Remove(내역 삭제)

```
            date      sort       category       detail amount  balance
0  2018 - 01 - 01         .              .            .      .    10000
1  2018 - 01 - 03    Income   Pocket money          Mom   5000    15000
2  2018 - 01 - 04    Income         Salary  PT job_cafe   7600    22600
3  2018 - 01 - 04  Spending           Food       coffee  -1500    21100
4  2018 - 01 - 04  Spending           Food        bread  -2000    19100
5  2018 - 01 - 05  Spending     Transp fee          Bus  -1300    17800
6  2018 - 01 - 06    Income  Financ income     interest   1100    18900
7  2018 - 01 - 07    Saving        KB Bank  installment  -5000    13900

Choose the index number you want to delete:
```

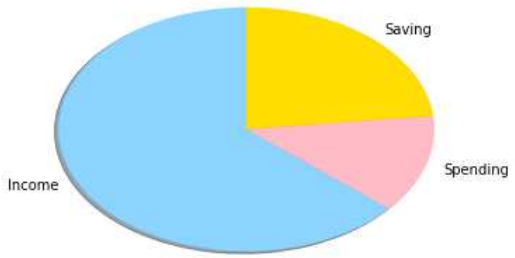
: 내역 이 출력되면 삭제하고 싶은 내역의 index number를 입력한다. (숫자로 입력)

```
================================================================================
          date      sort       category       detail amount  balance
0  2018 - 01 - 01         .              .            .      .    10000
1  2018 - 01 - 03    Income   Pocket money          Mom   5000    15000
2  2018 - 01 - 04    Income         Salary  PT job_cafe   7600    22600
3  2018 - 01 - 04  Spending           Food       coffee  -1500    21100
4  2018 - 01 - 05  Spending     Transp fee          Bus  -1300    19800
5  2018 - 01 - 06    Income  Financ income     interest   1100    20900
6  2018 - 01 - 07    Saving        KB Bank  installment  -5000    15900
================================================================================

Complete.

Enter 0 to continue using the program, or the other to Exit.
```

: 이때도 역시 삭제 후 내역이 출력된다. 이때도 프로그램을 계속 사용할 것이면 0, 아니면 다른 숫자를 입력한다.

-4) Analysis(분석)

```
+======================< Total >=======================+

                  income : 13700.0 won

                spending : 2800.0 won

                  saving : 5000.0 won
```




```
Enter 0 to continue using the program, or the other to Exit.
```

: 4(분석)을 선택하면 총수입, 총지출, 총저축의 합계금액과 그 비율이 Pie 차트로 출력된다.

④ 프로그램 종료

~~~~~~~~~~~~~~~~~~~~ <test's Household Accounts > ~~~~~~~~~~~~~~~~~~~~

	date	sort	category	detail	amount	balance
0	2018 - 01 - 01	.	.	.	.	10000
1	2018 - 01 - 03	Income	Pocket money	Mom	5000	15000
2	2018 - 01 - 04	Income	Salary	PT job_cafe	7600	22600
3	2018 - 01 - 04	Spending	Food	coffee	-1500	21100
4	2018 - 01 - 05	Spending	Transp fee	Bus	-1300	19800
5	2018 - 01 - 06	Income	Financ income	interest	1100	20900
6	2018 - 01 - 07	Saving	KB Bank	installment	-5000	15900

: 작업을 종료하면 지금까지의 데이터가 출력되고 프로그램이 종료된다.

4. 한계점 및 느낀 점

이 프로그램의 한계는 대표적으로 두 가지가 있다. 첫 번째는 입력의 인자의 type 형태가 맞지 않으면 Error가 뜬다는 것이고, 두 번째는 내가 계획했던 경고 메시지를 완벽하게 구현하지 못했다는 점이다.

먼저 입력 인자의 type 형태가 맞으면 프로그램이 Error가 뜨는 것은 예를 들면 내역을 삭제할 때 index number입력 시 숫자가 아닌 알파벳이나 한글, 또는 기호 등 다른 type의 인자를 입력하면 프로그램이 error가 뜨는 것을 의미한다. 이 경우는 while이나 If문을 활용하면 다시 입력받도록 할 수 있겠지만 이것에는 예외적인 경우가 너무 많아 중첩된 loop가 너무 많이 생기게 된다. 이렇게 되면 프로그램 자체가 무거워지고 오류가 생기기 쉬워서, 프로그램 실행 과정에서 입력 예시를 print 하여 보여주는 방법으로 방안을 사용하였다. 하지만 사용자가 이 예시를 무시하고 다른 것을 입력할 경우, 입력 창으로 되돌아가도록 하지 못한 점이 아쉽다.

두 번째로 내가 계획했던 경고 메시지를 완벽하게 구현하지 못했다는 것은 두 가지를 의미한다. 하나는 메시지를 보내는 기능을 구현하지 못한 것이다. 이 기능을 사용하기 위해서는 twillo.com에서 계정을 만들고 휴대폰 번호를 입력하여 인증 절차를 수행한 후 코드에서 구현이 가능했는데, 이 인증이 유료화 되어있었다. 그래서 이 기능을 사용하지 않기로 하였다. 또한, 사용자가 프로그램을 사용할 때 바로 메시지가 출력되기 때문에 불필요하다고 느껴진 것도 기능 삭제의 이유 중 하나이다. 다른 하나는 자산 사용주기를 설정하지 못한 것이다. 자산 사용주기는 예를 들면, 매달 5일에 월급이 들어온다면 이번 달 5일부터 다음 달 5일까지의 주기를 의미한다. 이것을 구현하기 위해서는 월과 일을 입력하는 것에서 그것이 실제 날짜 체계로 연관되도록 설정해야 했다. 또, 사용주기가 얼마인지, 그 사용주기의 시작일은 얼마인지 등을 모두 입력받도록 해야 했다. 이것을 구현하는 방법을 여러 가지로 생각해 보았지만 결국 구현하지 못해서 아쉽다. 더 많은 연구 통해서 이 점을 개선할 방안을 마련해보고 싶다. 또 분석 기능을 사용했을 때 더 자세한 분석을 하도록 개선할 것이다.
~~~~~~~~~~~~~~~~~~~~