Python programming and practice

Expenditure Management Program Production Project

Final Report

2023. 12. 24
Kim Liha
204201

1. Introduction

1) Background

People are having a hard time saving due to increased consumption that they don't need due to incorrect consumption habits. You need a program that can simply check your consumption and help you develop the right consumption habits.

2) Project goal

Store the user's expenditure details by content and allow you to check the average consumption per month.

3) Differences from existing programs

If you look at the amount of consumption only in units of numbers, you may not feel it, so you can feel more intuitively how much you saved through the price of the product (e.g., coffee, chicken).

2. Functional Requirement

1) Expenditure storage function

- Expenditures are entered, stored, and converted to date data.

2) Spending history confirmation function

- Prints the sum of past expenditure details and expenditure amount.

3. Implementation

(1) Save Expenditure History

- Enter: Expenditure (spending, amount, date, category)

- Output: Saved

- Explanation: When the user enters the expenditure details, it is saved with the date in the csv file.

- learned applied:

  Module (modularize functions)

  Functions (use functions to store expenses, print details, etc.)

  Pandas, csv (save the details as csv data)

- Code screenshot

```python
# csv 저장 함수
def save_to_csv(expense_name, amount, date, category):
    # CSV 파일에 저장할 데이터
    data = [[expense_name, str(amount), date, category]]

    #CSV 파일에 데이터 추가
    with open("가계부.csv", "a", newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerows(data)
```

```python
# 지출 입력, 저장 함수
def enter_expense():
    # 사용자로부터 지출 정보 입력 받기
    expense_name = input("지출 항목을 입력하세요: ")
    print("\n")
    amount = float(input("금액을 입력하세요: "))
    print("\n")
    date = input("날짜를 입력하세요 (YYYY-MM-DD 형식): ")
    print("\n")
    category = input("카테고리를 입력하세요: ")
    print("\n")

    # 입력된 정보 파일에 저장
    save_to_csv(expense_name, amount, date, category)
    print("지출 정보가 저장되었습니다.\n")

    # 데이터 정리를 위해 csv 카피 (가계부 정리 파일로 카피 후 데이터를 변환합니다.)
    shutil.copy('가계부.csv', '가계부정리.csv')
```

(2) Spending history view function

- Output: Expense details

- Explanation: Print out the spending details for a specific month or the total spending for each month.

- learned applied:

    Module (modularize functions)

    Functions (Expenditure details output function)

    Pandas, csv (Import expenditure data, output data in table)

- Code screenshot

```python
# 특정 달의 지출 내역
# 연도는 지정 불가합니다.(현재 csv 파일에 2023 데이터만 있습니다.)
def show_expenses_month():
    df = pd.read_csv('가계부정리.csv')
    month = int(input("지출 내역을 조회할 달을 입력해주세요. \n예시) 8월 --> 8: "))

    # 입력받은 month만 추출
    filtered_month = df.loc[df['월'] == month]

    # 카테고리까지만 범위로 지정하여 중복되는 정보 출력 X
    filtered_index = filtered_month.loc[:, :'카테고리']
    print(tabulate(filtered_index, headers='keys', tablefmt='grid', showindex=False, stralign='center'))



# pandas를 이용해서 데이터 날짜 변환 (사용자가 입력한 (YYYYMMDD) 데이터를 날짜 데이터로 변환시킵니다.)
def date_conversion():
    df = pd.read_csv('가계부정리.csv')
    df['날짜(2023MMDD)'] = pd.to_datetime(df['날짜(2023MMDD)'], format='%Y%m%d')
    df['날짜_datetime'] = pd.to_datetime(df['날짜(2023MMDD)'])
    df['연도'] = df['날짜_datetime'].dt.year
    df['월'] = df['날짜_datetime'].dt.month
    df['일'] = df['날짜_datetime'].dt.day
    df['요일'] = df['날짜_datetime'].dt.day_name()
    df.to_csv('가계부정리.csv', index=False)


# 각 달의 지출 합계 출력 함수
def sum_expense():
    df = pd.read_csv('가계부정리.csv')

    # 년,월을 추출하여 각 기간의 합을 출력
    newtable = pd.pivot_table(data=df,index= ['연도', '월'], values='지출액', aggfunc='sum')
    print(tabulate(newtable, headers='keys', tablefmt='grid', stralign='center'))


    # 12월의 소비액을 치킨값으로 표현

    dec_expense = newtable.loc[(2023, 12), '지출액']
    price_chiken = float(dec_expense / 20000)

    print(f'12월 지출로 치킨을 {round(price_chiken, 1)}마리 살 수 있어요!')
```

4. Test Result

(1) Save Expenditure History

- 1. Enter the spending amount, item, date, and category from the user and save it in a csv file.

  2. Copy the saved csv file to another csv file for data conversion when outputting.

- Result screenshot

(console)

```
::::::::::::::::::
지출 관리 프로그램
::::::::::::::::::
::::::::메뉴::::::::
1. 지출 내용 입력
2. 지출 상세
3. 나의 한달 지출
4. 종료
번호를 입력해주세요: 1


지출 항목을 입력하세요: 맥도날드


금액을 입력하세요: 8000


날짜를 입력하세요 (YYYY-MM-DD 형식): 20231221


카테고리를 입력하세요: 식비


지출 정보가 저장되었습니다.
```

(csv)

```
음악강습,55000,20230710,취미/교육
의류구매,80000,20230715,쇼핑
전화요금,35000,20230720,통신비
자동차 주유,90000,20230725,교통비
도서구입,38000,20230730,문화
의료비,50000,20230805,의료/건강
식당외식,42000,20230810,식비
점심,8000.0,20231208,식비
저녁밥,9500.0,20231208,식비
저녁밥,8500.0,20231208,식비
맥도날드,8000.0,20231221,식비
```

(csv file used for output)

```
도서구입,38000.0,2023-07-30,문화,2023-07-30,2023,7,30,Sunday
의료비,50000.0,2023-08-05,의료/건강,2023-08-05,2023,8,5,Saturday
식당외식,42000.0,2023-08-10,식비,2023-08-10,2023,8,10,Thursday
점심,8000.0,2023-12-08,식비,2023-12-08,2023,12,8,Friday
저녁밥,9500.0,2023-12-08,식비,2023-12-08,2023,12,8,Friday
저녁밥,8500.0,2023-12-08,식비,2023-12-08,2023,12,8,Friday
맥도날드,8000.0,2023-12-21,식비,2023-12-21,2023,12,21,Thursday
```

(2) Spending history view

- 1. You can check your spending details for a specific month.

  2. You can check the sum of your past expenses.

- Result screenshot

(menu 2)

```
::::::::::::::::::
지출 관리 프로그램
::::::::::::::::::
:::::::메뉴:::::::
1. 지출 내용 입력
2. 지출 상세
3. 나의 한달 지출
4. 종료
번호를 입력해주세요: 2


지출 내역을 조회할 달을 입력해주세요.
예시) 8월 --> 8: 4
+--------------+----------+-----------------+------------+
|   지출내용   |  지출액  | 날짜(2023MMDD)  |  카테고리  |
+==============+==========+=================+============+
|   여행경비   |   180000 |    2023-04-05   |    여행    |
+--------------+----------+-----------------+------------+
|   음악강습   |    60000 |    2023-04-10   | 취미/교육  |
+--------------+----------+-----------------+------------+
|   의류구매   |    85000 |    2023-04-15   |    쇼핑    |
+--------------+----------+-----------------+------------+
|   전화요금   |    32000 |    2023-04-20   |  통신비    |
+--------------+----------+-----------------+------------+
| 자동차 수리비 |   100000 |    2023-04-25   |  교통비    |
+--------------+----------+-----------------+------------+
|   도서구입   |    40000 |    2023-04-30   |    문화    |
+--------------+----------+-----------------+------------+
```

(menu 3)

```
::::::::::::::::::
지출 관리 프로그램
::::::::::::::::::
:::::::메뉴:::::::
1. 지출 내용 입력
2. 지출 상세
3. 나의 한달 지출
4. 종료
번호를 입력해주세요: 3


+------------+----------+
|            |  지출액  |
+============+==========+
| (2023, 1)  |   325000 |
+------------+----------+
| (2023, 2)  |   467000 |
+------------+----------+
| (2023, 3)  |   331000 |
+------------+----------+
| (2023, 4)  |   497000 |
+------------+----------+
| (2023, 5)  |   525000 |
+------------+----------+
| (2023, 6)  |   314000 |
+------------+----------+
| (2023, 7)  |   488000 |
+------------+----------+
| (2023, 8)  |    92000 |
+------------+----------+
| (2023, 12) |    34000 |
+------------+----------+
12월 지출로 치킨을 1.7마리 살 수 있어요!
```

5. Changes in Comparison to the Plan

1) Output expenditure history

- Before: Enter year and month to output history

- After: Limit duration to months

- Reason: I tried to get the year and month intersection but failed because the datetime error continued.

6. Lessons Learned & Feedback

- Even though I took a basic programming class, I experienced a lot of trial and error and felt that I still had a lot to learn. This was my first-class using GitHub, and even though it was a personal project, I was able to feel the need for GitHub. Also, various new contents used in the field that I had not heard about in other classes were very helpful.