

파이썬 파일처리

가계부 프로그램

김이삭 isaac7263@naver.com

<https://i-sak.github.io/profile>

목차

1. 프로그램 소개

2. 프로그램 구조

1) 클래스 다이어그램

2) 순차 다이어그램

3. 프로그램 소스코드 설명

4. 구현 화면

5. 소감 및 향후 계획

1. 프로그램 소개

[가계부 프로그램] 만들기 과제를 받아 만들게 된 가계부 프로그램입니다. 이 가계부 프로그램은 파일을 생성하고, 파일 내부의 내용을 불러들여와 GUI 화면으로 출력하는 프로그램입니다.

1 차 초기 버전으로 콘솔 가계부 프로그램을 만들었고, 2 차 버전으로 GUI 를 포함한 프로그램을 만들었습니다. GUI 버전 기준으로 프로그램의 기능을 설명하자면 엑셀 파일을 생성하고, 파일 탐색기를 통해 엑셀파일을 불러들이고, 파일 내부의 전체 내용 출력, 파일에 내용(하나의 행)을 추가하거나, 내용(하나의 행) 삭제, 사용내역 검색하여, 합계 출력 등의 기능이 있습니다.

일반적으로 가계부를 작성할 때 표의 형태로 작성을 하는 경우가 많습니다. 따라서 일반 텍스트 파일이 아닌, 엑셀 파일을 생성하고 다룰 수 있는 openpyxl 라이브러리를 사용하여 파일 입출력을 하게 되었습니다.

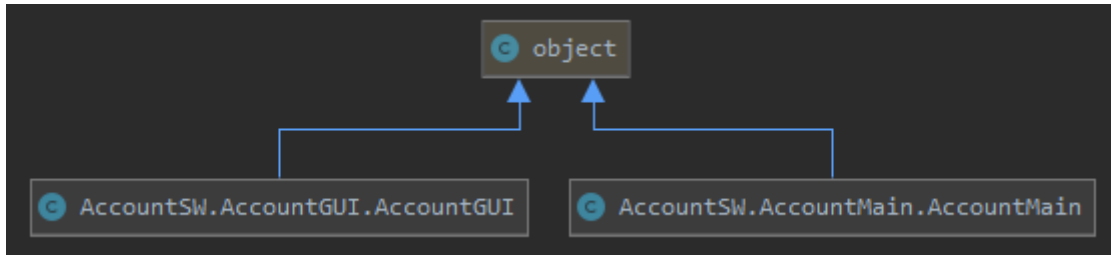
Gui 는 tkinter 라이브러리를 사용하였습니다.



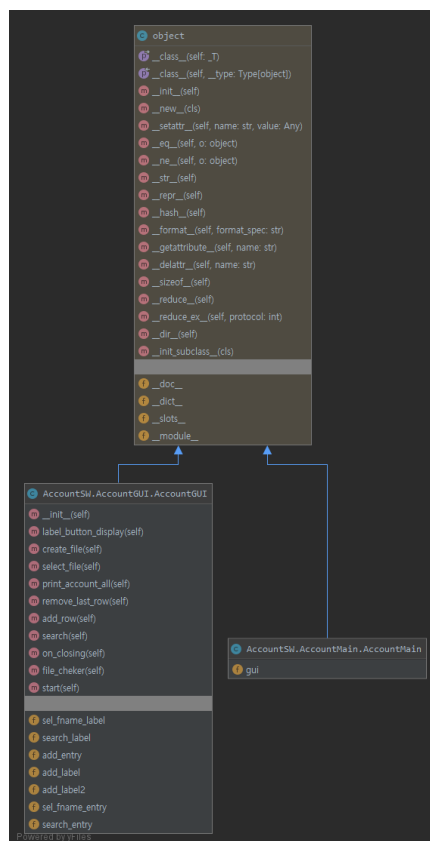
2. 프로그램 구조

1) 클래스 다이어그램

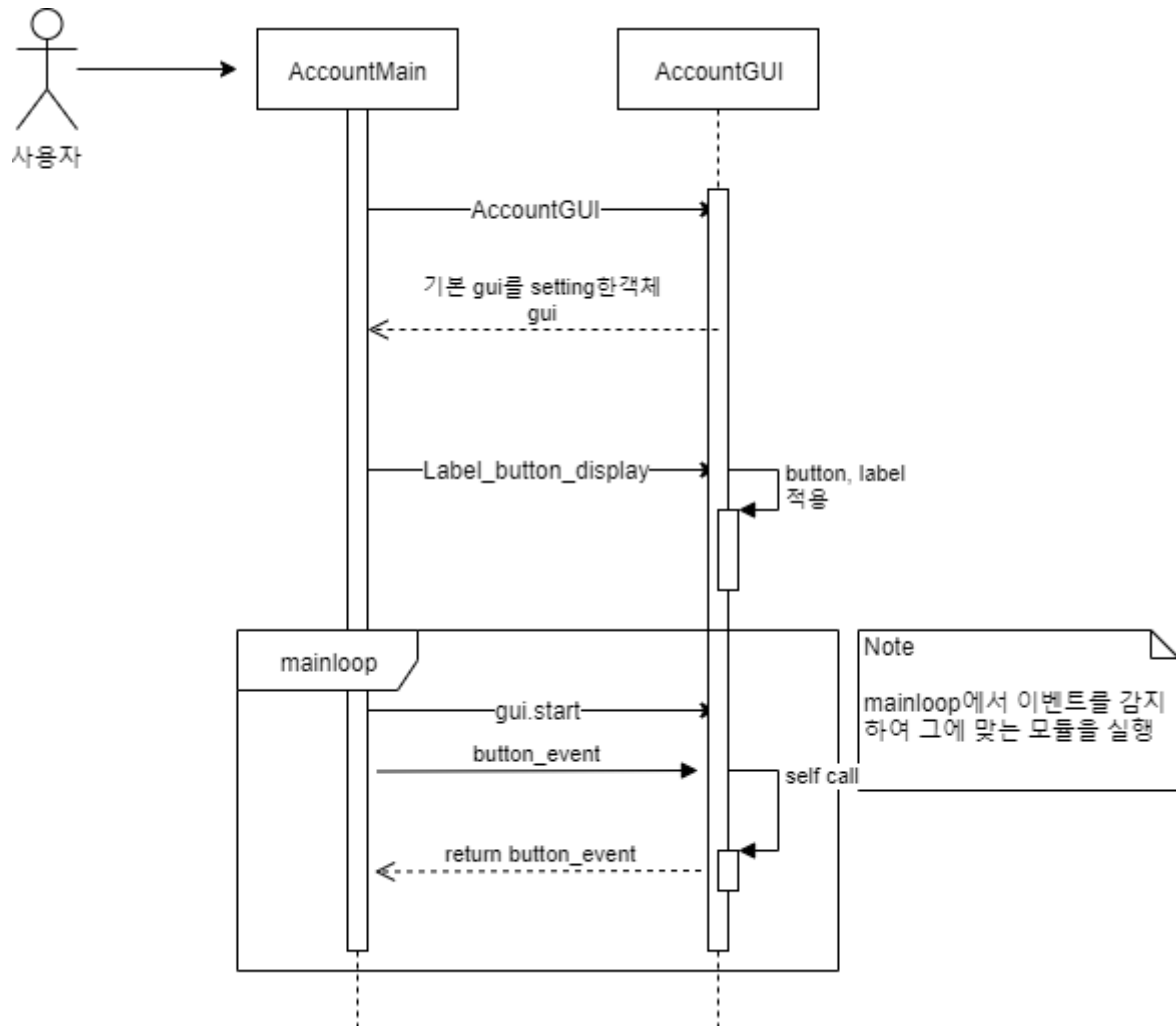
프로그램은 파이썬 언어로 구현이 되었습니다. 클래스는 다음과 같이 구현이 되어 있습니다.



다음 이미지는 Methods 와 Fields 를 포함한 클래스 다이어그램 입니다.



2) 순차 다이어그램

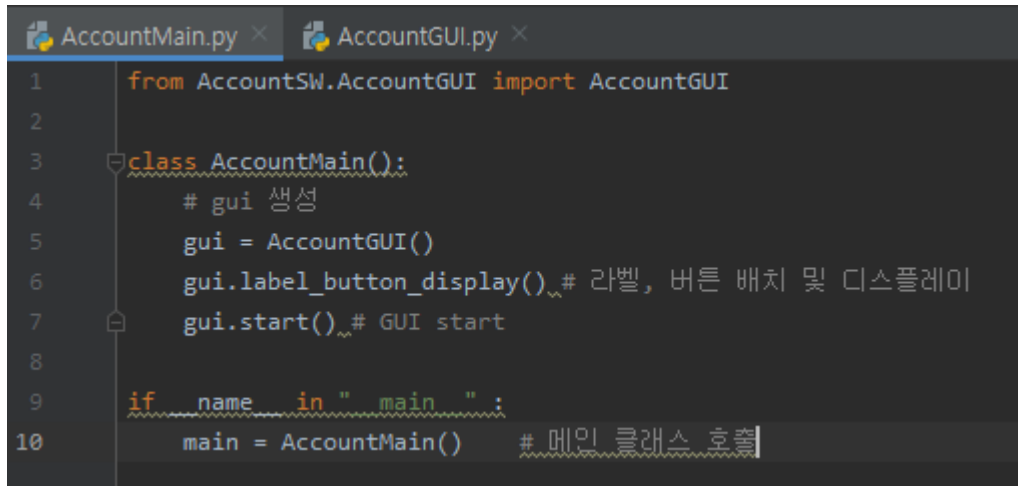


사용자는 메인 클래스로부터 GUI 클래스를 생성, 호출하고, 라벨, 버튼의 적용 기능을 호출하고, 게임 스타트를 호출합니다. Mainloop에서는 이벤트를 감지하는 루프가 실행됩니다. 이벤트를 감지하는 AccountGUI는 클래스에 정의되어 있는 각 모듈들을 self call 하여 이벤트를 발생시킵니다.

3. 프로그램 소스코드 설명

파일명 : AccountMain.py

[class] AccountMain



```
1 from AccountSW.AccountGUI import AccountGUI
2
3 class AccountMain():
4     # gui 생성
5     gui = AccountGUI()
6     gui.label_button_display() # 라벨, 버튼 배치 및 디스플레이
7     gui.start() # GUI start
8
9 if name in " main " :
10     main = AccountMain() # 메인 클래스 호출
```

AccountMain.py 을 프로그램 시작으로 정하였습니다. 파이썬 문서의 main 을 통해 AccountMain 클래스를 생성하게 됩니다. AccountMain 클래스는 gui 를 구현하기 위한 AccountGUI 클래스를 생성합니다.

AccountGUI 클래스에 구현된 모듈

모듈 1. __init__(self) : gui 를 구현하기 위한 초기화 함수

모듈 2. Label_button_display(self) : GUI 라벨과 버튼의 생성 및 배치

모듈 3. Create_file(self) : 엑셀 파일 생성하기

모듈 4. Select_file(self) : 파일 브라우징으로 파일 선택하기

모듈 5. Print_account_all(self) : 불러들인 파일 내용을 전체 출력

모듈 6. Remove_last_row(self) : 가계부 파일 마지막 행 삭제하기

모듈 7. Add_row(self) : 가계부 파일 마지막 행에 사용(지출)내역 추가하기

모듈 8. Search(self) : 가계부 파일 내에 사용내역을 특정 검색어로 검색하여 출력하기

모듈 9. On_closing(self) : [X] 버튼에 대한 종료 확인 메시지

모듈 10. File_checker(self) : 파일 체커 - 파일의 존재 유무를 확인하는 기능

모듈 11. Start(self) : GUI 시작

파일명 : AccountGUI.py

[class] A

```
AccountMain.py x AccountGUI.py x
1 import tkinter as tk # tk 이름으로 사용
2 import tkinter.font # 폰트
3 from tkinter import ttk # Label 추가
4 from tkinter import scrolledtext # 스크롤 박스
5 from tkinter import messagebox # 종료시 메세지 박스
6 from tkinter import filedialog # 파일 선택창
7 from openpyxl import load_workbook # 파일 로드
8 from openpyxl import Workbook # 파일 생성
9 import getpass # os 로그인 정보 얻기
10
11 # GUI 관리 클래스
12 class AccountGUI():
13     def __init__(self): # 초기화
14         global win, file, scr, not_selected_msg, cre_num
```

AccountGUI .py 파일에서 AccountGUI 클래스에 사용될 라이브러리 및 모듈의 목록입니다.

◆ 모듈 1. __init__(self) : gui 를 구현하기 위한 초기화 함수

```
# GUI 관리 클래스
class AccountGUI():
    def __init__(self): # 초기화
        global win, file, scr, not_selected_msg, cre_num
        # Create tk instance
        win = tk.Tk()
        win.title("가계부 by. isaac") # 제목
        win.geometry("1100x630") # window width x window Height + position right + position down / 가로 x 세로
        win.resizable(False, False) # x, y 사이즈 변경 불가
        font=tk.font.Font(family="맑은 고딕", size=20) # 폰트

        ttk.Label(win, text="가계부 by. isaac", font=font).grid(column=0, row=0, columnspan=10) # 프로그램 제목 라벨

        label_text=["날짜", "사용내역", "지출구분", "기존잔액", "사용금액", "사용 후 잔액"] # column label
        col_label = [] # Label 리스트 : 출력부
        for i in range(len(label_text)): # label_text 배치
            col_label.append(ttk.Label(win, text=label_text[i], width=15)) # label 추가
            col_label[i].grid(column=i, row=1) # 첫 행

        # 스크롤 텍스트 창
        scr = scrolledtext.ScrolledText(win, width=130, height=30, wrap=tk.WORD)
        scr.config(state='disabled') # readonly 속성
        scr.grid(column=0, columnspan=6)

        # 파일
        file = ""
        cre_num = 0 # 추후 파일 생성 번호 컨트롤

        # 자주 쓰이는 메시지
        not_selected_msg = "파일이 선택되지 않았습니다.\n\n먼저 파일을 선택해주세요."

        # 종료 이벤트
        win.protocol("WM_DELETE_WINDOW", self.on_closing)
```

Gui 틀, 제목, 기본적인 라벨, 스크롤 텍스트창, 메시지, 종료 이벤트에 대한 세팅을 합니다.

◆ 모듈 2. Label_button_display(self) : GUI 라벨과 버튼의 생성 및 배치

```
# 라벨, 버튼 생성 및 배치 함수
def label_button_display(self):
    # ----- 생성 -----
    # 선택된 파일 라벨
    self.sel_fname_label = tk.Label(win, text="선택된 파일") # 파일 선택 라벨
    self.sel_fname_entry = tk.Entry(win) # 파일 선택 엔트리
    self.sel_fname_entry.insert(0, "파일을 선택해주세요.") # 파일 선택 엔트리 문구
    self.sel_fname_entry.config(state='readonly') # readonly 속성
    self.add_label = tk.Label(win, text="[*가계부 지출내역 추가*] 슬래쉬(/)를 구분문자로 추가할 가계부 내역을 작성해주세요."
                                "\nex) 2019-11-29/간식/현금/50000(마지막 사용잔액)/5000(사용금액)") # 파일 추가 라벨
    self.add_entry = tk.Entry(win, width=120) # 엔트리 사이즈
    self.add_label2 = tk.Label(win, text="*입금시 사용금액에 음수로 작성!! ex) -5000 [5000원 입금]")
    self.search_entry = tk.Entry(win, width=20) # 사용내역 검색 엔트리 사이즈
    self.search_label = tk.Label(win, text="*사용내역 기준으로 검색") # 사용내역 라벨
    # 버튼
    b0 = tk.Button(win, text="x1 파일 생성하기", width=20, command=self.create_file)
    b1 = tk.Button(win, text="파일 선택하기", width=20, command=self.select_file)
    b2 = tk.Button(win, text="가계부 출력하기", width=20, command=self.print_account_all)
    b3 = tk.Button(win, text="마지막 행 삭제", width=20, command=self.remove_last_row)
    b4 = tk.Button(win, text="사용내역 검색하기", width=20, command=self.search)
    b5 = tk.Button(win, text="가계부 지출내역 추가", width=20, command=self.add_row)
    btq = tk.Button(win, text="종료", width=20, command=self.on_closing)
```

```
# -----배치-----
# 버튼
bx = 937
b0.place(x=bx, y=120) # x1 파일 생성
b1.place(x=bx, y=150) # 파일 선택
b2.place(x=bx, y=180) # 가계부 출력
b3.place(x=bx, y=210) # 마지막 행 삭제
b4.place(x=bx, y=330) # 사용내역 검색하기
b5.place(x=bx, y=508) # 가계부 지출내역 추가
btq.place(x=bx, y=570) # 종료

# 파일선택 라벨, 엔트리
self.sel_fname_label.place(x=bx, y=50) # 파일 이름 라벨
self.sel_fname_entry.place(x=bx, y=80) # 파일 엔트리
self.add_label.place(x=30, y=470) # 행 추가하기 라벨1
self.add_label2.place(x=30, y=530) # 행 추가하기 라벨2
self.add_entry.place(x=30, y=510) # 행 추가하기 엔트리
self.search_label.place(x=bx, y=270) # 사용내역 검색에 대한 라벨
self.search_entry.place(x=bx, y=300) # 사용내역 검색에 대한 엔트리
```

기능에 사용될 엔트리, 라벨, 버튼을 생성하고 위치를 배치합니다. 버튼은 클릭 이벤트 발생시에 설정된 함수를 호출합니다.

◆ 모듈 3. Create_file(self) : 엑셀 파일 생성하기


```

# xl 파일 생성하기
def create_file(self):
    global cre_num
    cre_num += 1
    tmp_name = "가계부" + str(cre_num) + ".xlsx"
    Workbook().save("C://Users/" + getpass.getuser() + "/Desktop/" + tmp_name) # 파일 생성
    save_file = "C://Users/" + getpass.getuser() + "/Desktop/" + tmp_name
    load_wb = load_workbook(save_file, data_only=True) # 파일 로드
    sheet = load_wb["Sheet"]
    sheet.append(["안녕하세요. 반갑습니다. by isaac"]) # list로 넣기
    load_wb.save(save_file) # 저장
    load_wb.close() # 닫기

```

엑셀 파일을 생성하는 함수입니다. 파일을 생성한 뒤, 문구 하나를 작성하고, 저장하고 파일을 닫아줍니다.

◆ 모듈 4. Select_file(self) : 파일 브라우징으로 파일 선택하기

```

# 파일 선택하기
def select_file(self):
    global file
    file = filedialog.askopenfilename(initialdir="C://Users/" + getpass.getuser() + "/Desktop/") # 파일 선택창
    sel_value = self.sel_fname_entry.get() # 선택된 파일 값
    self.sel_fname_entry.config(state='normal') # entry 상태
    self.sel_fname_entry.delete(0, len(sel_value)) # 값 지우기
    self.sel_fname_entry.insert(0, file) # 값 넣기
    self.sel_fname_entry.config(state='readonly') # readonly

    scr.config(state='normal') # 작성할 수 있도록 함
    scr.delete('1.0', tk.END) # 내용 지우기
    scr.config(state='disabled') # 읽기 전용

```

파일 선택하기 버튼 클릭 시 파일 브라우징을 통해 파일을 선택할 수 있습니다.

◆ 모듈 5. Print_account_all(self) : 불러들인 파일 내용을 전체 출력

```
# 가계부 출력하기
def print_account_all(self):
    if self.file_checker() == False: # 파일체커 파일이 있는지 없는지 검사
        return

    load_wb = load_workbook(file, data_only=True) # 파일 로드
    sheet = load_wb["Sheet"] # Sheet1 로드

    # 출력하기
    scr.config(state='normal') # 작성할 수 있도록 함
    scr.delete('1.0', tk.END)
    # =====
    sheet_rows = []
    str_rows = ""
    for row in sheet.rows:
        # sheet_row = [] # 한 줄에 대해서 초기화
        for cell in row:
            # sheet_row.append(cell.value) # 한 줄의 셀 모두 더하여, 행 만들기
            str_rows += str(cell.value).replace(" 00:00:00", "") # 모든 셀을 하나의 row로 만들어주기
            str_rows += "\t\t\t\t"
        str_rows += '\n'
    scr.insert(tk.INSERT, str_rows) # 스크롤 박스에 추가하기
    # =====
    scr.config(state='disabled') # 읽기 전용
    load_wb.close()
```

엑셀 파일의 첫번째 시트를 불러오고 전체 행을 출력합니다.

◆ 모듈 6. Remove_last_row(self) : 가계부 파일 마지막 행 삭제하기

```
# 마지막 행 삭제하기
def remove_last_row(self):
    if self.file_checker() == False: # 파일체커 파일이 있는지 없는지 검사
        return
    else:
        confirm = tk.messagebox.askquestion("경고 메시지", "정말로 삭제하시겠습니까?") # yes, no
        if confirm == "no":
            return

    global sh_rows_len, sh_cols_len # 행의 수와 열의 수
    load_wb = load_workbook(file, data_only=True) # 파일 로드
    sheet = load_wb["Sheet"] # Sheet1 로드
    sh_rows_len = sheet.max_row
    sh_cols_len = sheet.max_column
    sheet.delete_rows(sh_rows_len) # 마지막 행 삭제하기
    load_wb.save(file) # 파일 저장하기
    load_wb.close() # 파일 닫기
```

◆ 모듈 7. Add_row(self) : 가계부 파일 마지막 행에 사용(지출)내역 추가하기

```
# 가계부 지출내역 추가하기
def add_row(self):
    if self.file_checker() == False: # 파일체커 파일이 있는지 없는지 검사
        return

    text = self.add_entry.get() # 입력된 텍스트를 가져온다.
    if text.count('/') != 4:
        tk.messagebox.showinfo("알림 메시지", "ex)\n 2019-11-29/간식/현금/50000(마지막 사용잔액)/5000(사용금액)")
        return

    load_wb = load_workbook(file, data_only=True) # 파일 로드
    sheet = load_wb["Sheet"] # sheet 로드
    sh_rows_len = sheet.max_row # sheet 행의 개수
    row_data = text.split("/")

    # 행 단위로 추가
    sheet.append([
        row_data[0], row_data[1], row_data[2], row_data[3], row_data[4], int(row_data[3]) - int(row_data[4])
    ])
    load_wb.save(file) # 파일 저장하기
    load_wb.close() # 파일 닫기

    self.add_entry.delete(0, len(text)) # 입력창 지우기
    tk.messagebox.showinfo("알림 메시지", "데이터가 추가되었습니다.")
```

가계부 지출내역 추가하기 입니다. 파일이 선택되었는지 확인하고, 내용을 추가하기 위해 엔트리를 확인합니다. 예시를 나타내고 있으며 작성 예시와 같이 작성하였다면 정상적으로 저장됩니다.

◆ 모듈 8. Search(self) : 가계부 파일 내에 사용내역을 특정 검색어로 검색하여 출력하기

사용 내역에서 엔트리에 입력된 검색어를 확인하고, 검색어와 일치하는 내용이 사용내역에 있다면 해당하는 행만 출력하고, 그 행에서 지출된 비용의 합을 마지막에 출력하도록 하였습니다.

모듈 9. On_closing(self) : [X] 버튼에 대한 종료 확인 메시지

모듈 10. File_checker(self) : 파일 체커 - 파일의 존재 유무를 확인하는 기능

모듈 11. Start(self) : GUI 시작

```
# 버튼 기능 : 종료 메시지
def on_closing(self):
    if messagebox.askokcancel("Quit", "Do you want to quit?"):
        win.destroy()

# 파일 체커 - 파일이 있는지 없는지를 체크한다.
def file_checker(self):
    if file == "":
        tk.messagebox.showwarning("경고 메시지", not_selected_msg) # 파일이 선택되지 않은 경우 메시지 출력
        return False

# Start GUI
def start(self):
    win.mainloop() # 이벤트 순환문을 시작할 때까지 gui는 표시되지 않는다.
```

모듈 9, 10, 11 은 간단한 내용으로 모듈에 소개된 내용으로 간단히 이해할 수 있습니다.

4. 구현 화면

가계부 by. isaac

가계부 by. isaac

날짜	사용내역	지출구분	기존잔액	사용금액	사용 후 잔액
----	------	------	------	------	---------

[*가계부 지출내역 추가*] 슬래쉬(/)를 구분문자로 추가할 가계부 내역을 작성해주세요.
ex) 2019-11-29/간식/현금/50000(마지막 사용잔액)/5000(사용금액)

*입금시 사용금액에 음수로 작성!! ex) -5000 [5000원 입금]

선택된 파일

파일을 선택해주세요.

xl 파일 생성하기

파일 선택하기

가계부 출력하기

마지막 행 삭제

*사용내역 기준으로 검색

사용내역 검색하기

가계부 지출내역 추가

종료

[구현-그림 1 메인 화면]

가계부 by. isaac

가계부 by. isaac

날짜	사용내역	지출구분	기존잔액	사용금액	사용 후 잔액
----	------	------	------	------	---------

[*가계부 지출내역 추가*] 슬래쉬(/)를 구분문자로 추가할 가계부 내역을 작성해주세요.
ex) 2019-11-29/간식/현금/50000(마지막 사용잔액)/5000(사용금액)

*입금시 사용금액에 음수로 작성!! ex) -5000 [5000원 입금]

선택된 파일

파일을 선택해주세요.

xl 파일 생성하기

파일 선택하기

가계부 출력하기

마지막 행 삭제

*사용내역 기준으로 검색

사용내역 검색하기

가계부 지출내역 추가

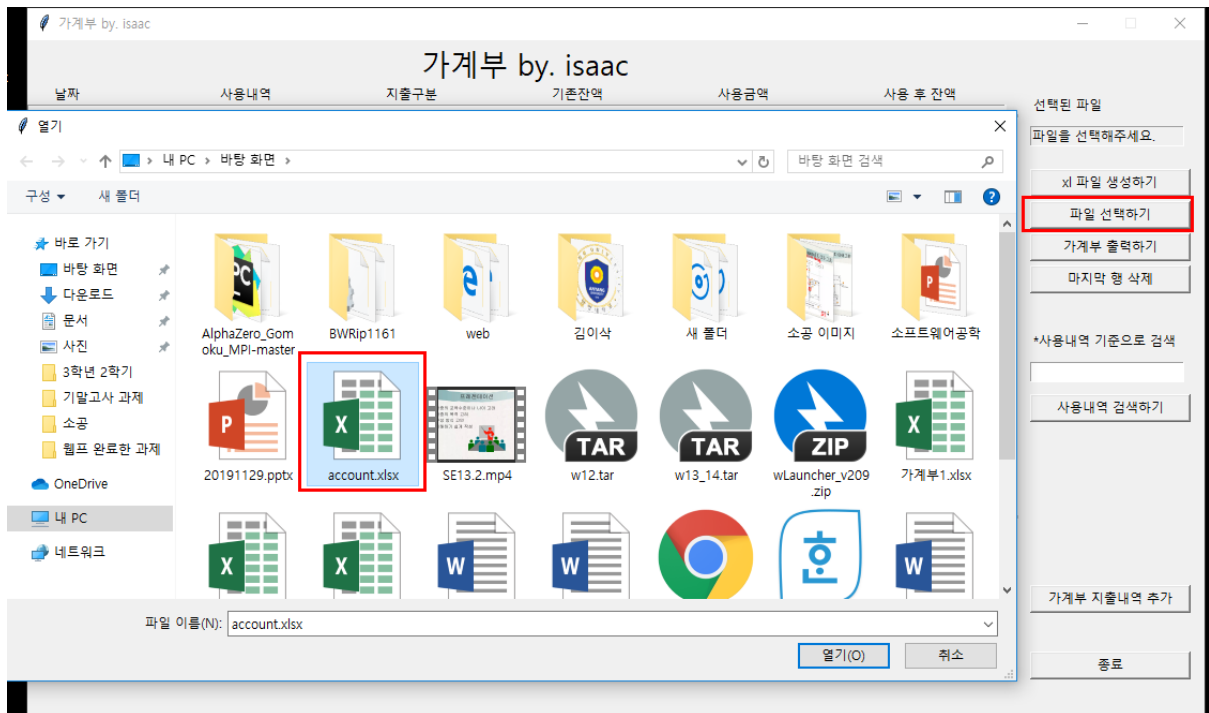
종료

가계부1.xlsx

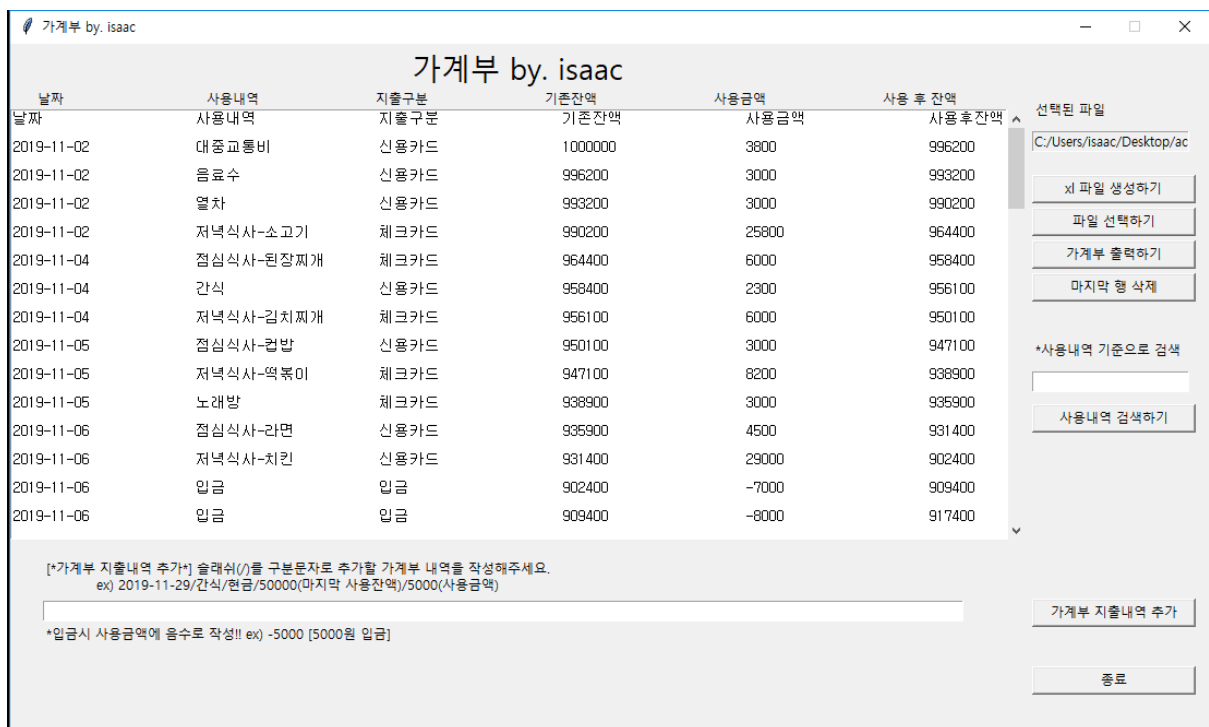
가계부2.xlsx

가계부3.xlsx

[그림 2 엑셀 파일 생성하기 화면]



[그림 3 파일 선택하기]



[그림 4 가계부 출력하기 화면]

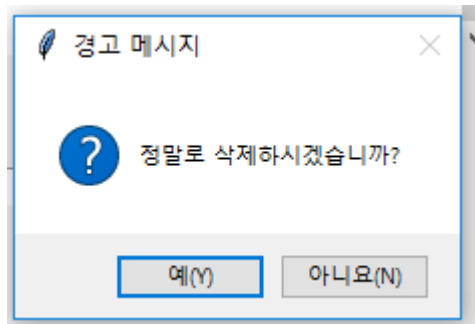
2019-12-03	점심식사(김밥)	신용카드	441500	2000	439500
2019-12-03	저녁식사(짜장면)	신용카드	439500	5000	434500
2019-12-04	점심식사(김치찌개)	신용카드	434500	6000	428500
2019-12-04	저녁식사(편의점)	신용카드	428500	4000	424500
2019-12-05	점심식사(김치찌개)	신용카드	424500	6000	418500
2019-12-05	저녁식사(짜장면)	신용카드	418500	5000	413500
2019-12-06	점심식사(편의점)	신용카드	413500	4000	409500
2019-12-09	저녁식사(김치찌개)	신용카드	409500	6000	403500
2019-12-10	야식(족발)	신용카드	403500	7500	396000

가계부 불러하기

마지막 행 삭제

*사용내역 기준으로 검색

사용내역 검색하기



2019-12-02	점심식사(김치찌개)	신용카드	447500	6000	441500
2019-12-03	점심식사(김밥)	신용카드	441500	2000	439500
2019-12-03	저녁식사(짜장면)	신용카드	439500	5000	434500
2019-12-04	점심식사(김치찌개)	신용카드	434500	6000	428500
2019-12-04	저녁식사(편의점)	신용카드	428500	4000	424500
2019-12-05	점심식사(김치찌개)	신용카드	424500	6000	418500
2019-12-05	저녁식사(짜장면)	신용카드	418500	5000	413500
2019-12-06	점심식사(편의점)	신용카드	413500	4000	409500
2019-12-09	저녁식사(김치찌개)	신용카드	409500	6000	403500

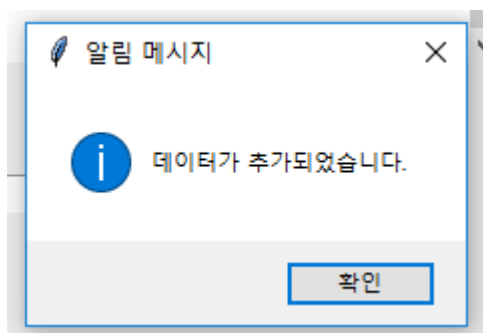
마지막 행 삭제

*사용내역 기준으로 검색

사용내역 검색하기

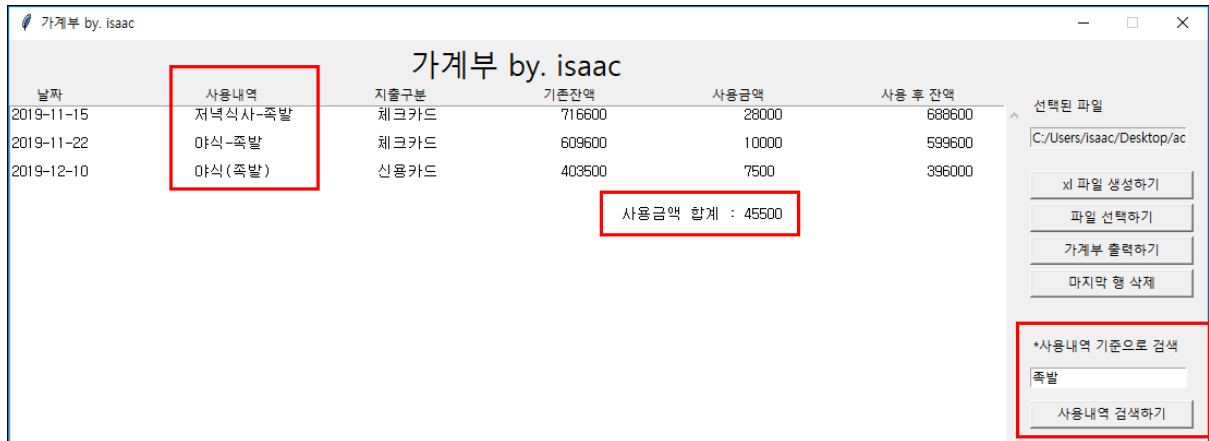
[*가계부 지출내역 추가시 슬래쉬(/)를 구분문자로 추가할 가계부 내역을 작성해주세요.
ex) 2019-11-29/간식/현금/50000(마지막 사용잔액)/5000(사용금액)]

[그림 5, 6, 7 – 마지막 행 삭제하기]



2019-12-05	저녁식사(짜장면)	신용카드	418500	5000	413500
2019-12-06	점심식사(편의점)	신용카드	413500	4000	409500
2019-12-09	저녁식사(김치찌개)	신용카드	409500	6000	403500
2019-12-10	야식(족발)	신용카드	403500	7500	396000

[그림 8, 9 – 가계부에 데이터 추가하기]



[그림 10 – 사용내역에서 죽발 검색]

5. 소감 및 향후 계획

소프트웨어를 개발하면서 실제로 사용자가 편안하게 느낄 수 있을지 고민하였고, 실제로 편하게 사용될 수 있도록 UI/UX 를 구현해보았다. 콘솔버전으로 시작하여 GUI 버전까지 구현을 하면서 다양한 지식을 새로 습득해야 하는 어려움이 있었지만, 얻은 것도 많았던 개발 경험이었다. 실제로 이 프로그램을 배포해보고 싶고, 사용이 잘 될 수 있다면 버전을 업그레이드 해보고 싶은 생각이 있다. 또한 구조적으로 잘 구현될 수 있도록 다양한 기법과 구문에 대해 학습을 이어 나갈 계획이다.