Python programming and practice

Gift Recommending system on MBTI Types

Final Report

Date : 2023. 12. 24

Name : Minchae Choi

ID : 223479

1. Introduction

1) Background

Recently, the MBTI personality type test has gained popularity and is being utilized in various fields. Additionally, there is a growing trend in the use of online gift-giving features.

2) Project goal

By recommending gifts based on MBTI personality types, the program offers convenience to its users, fulfills practicality for gift recipients, and additionally brings advertising benefits to the suppliers.

3) Differences from existing programs

The current gift recommendation system relies solely on preferences based on age and gender, which may not always align with the actual preferences of individuals. To address this limitation, a system has been developed to recommend gifts based on MBTI personality types. Furthermore, users can input their MBTI and preferred gifts to gather data, making the system more tailored to individual needs and preferences.

2. Functional Requirement

1) Gift recommendations based on MBTI types & category

- At first, the user inputs the gift category. Depending on the entered category, extract the most frequent item from the preferred gifts data files (4 files of each MBTI position E/I, S/N, F/T, P/J). This item becomes the recommended gift.

2) Enter favorite gift and MBTI type

- When a user inputs their preferred gift and their own MBTI into the program, it is saved in the preferred gift data list. This helps increase the amount of data, thus improving the program's performance.

3) Providing a gift card message

- Based on the category of the recommended gift, the program generates a gift card message. This function provides convenience to the user.

3. Implementation

1) Functional Requirements

(0) Backbone of the program & enter_mbti function

- For Backbone of the program, I applied class, module and utilized a constructor for automatic program execution upon class creation. In the constructor, I wrote calls to other methods within the class. Addly, I used a while loop and if-elif-else with a break for program termination.

- I created the **enter_mbti** function to consolidate repetitive sections. I applied While Loop, if-elif-else condition with logic operators, Function, and String(index and lower function).

```
import operator
import random

class MbtiGift:
    def __init__(self):
        print("******** This is a gift recommending program depending on mbti ********")

        while True:
            print("\n*************************************************************************")
            print("*     category: 도서/패션/상품권/코스메틱/전자기기/문구/리빙/건강     *")
            print("* ---------------------- << choose option >> ---------------------- *")
            print("*                                                                   *")
            print("* (1) Get gift recommendation | (2) Enter the desired gift | (3) Quit *")
            print("*                                                                   *")
            print("*************************************************************************")
            option = int(input("enter the number of your option --> "))

            if option == 1:
                self.output_gift()
            elif option == 2:
                self.input_gift()
            elif option == 3:
                print("프로그램을 종료합니다.")
                break
            else:
                print("Wrong input. Please input 1~3.")
```

```
def enter_mbti(self):
    while True:
        mbti = input("Enter the mbti --> ").lower()
        if len(mbti) != 4:
            print("Incorrect input !! MBTI consists of four letters.")
        elif mbti[0] != 'i' and mbti[0] != 'e':
            print("Incorrect input !! MBTI starts with 'i' or 'e'.")
        elif mbti[1] != 's' and mbti[1] != 'n':
            print("Incorrect input !! MBTI's second letter is 's' or 'n'.")
        elif mbti[2] != 'f' and mbti[2] != 't':
            print("Incorrect input !! MBTI's third letter is 'f' or 't'.")
        elif mbti[3] != 'p' and mbti[3] != 'j':
            print("Incorrect input !! MBTI's last letter is 'p' or 'j'.")
        else:
            return mbti
```

(1) Gift recommendations based on MBTI types & category

- Input the MBTI and the category.

- Output the recommending gift which is the most frequently preferred. Furthermore, it prints a gift card message corresponding to the category of the recommended gift through the invocation of the **output_msg** function.

- I applied File IO, Function, String(strip and split function), Condition, Loop(while & for), List, Set, Dictionary, Class and Module based on search-engine project.

```
def  output_gift(self):
    fp_e = open("present/present_e.txt", "r+", encoding="utf8")
    fp_i = open("present/present_i.txt", "r+", encoding="utf8")
    fp_n = open("present/present_n.txt", "r+", encoding="utf8")
    fp_s = open("present/present_s.txt", "r+", encoding="utf8")
    fp_f = open("present/present_f.txt", "r+", encoding="utf8")
    fp_t = open("present/present_t.txt", "r+", encoding="utf8")
    fp_p = open("present/present_p.txt", "r+", encoding="utf8")
    fp_j = open("present/present_j.txt", "r+", encoding="utf8")

    mbti_ = self.enter_mbti()
    gift_category = input("Enter the category --> ")
    gifts = []

    if mbti_[0] == 'e':
        gifts += fp_e.readlines()
    else:
        gifts += fp_i.readlines()
    if mbti_[1] == 'n':
        gifts += fp_n.readlines()
    else:
        gifts += fp_s.readlines()
    if mbti_[2] == 'f':
        gifts += fp_f.readlines()
    else:
        gifts += fp_t.readlines()
    if mbti_[3] == 'p':
        gifts += fp_p.readlines()
    else:
        gifts += fp_j.readlines()
```

```
category_selected_gifts = []
for gift in gifts:
    tokens = gift.strip().split(",")
    if tokens[0] == gift_category:
        category_selected_gifts.append(tokens[1])

freq = dict()
category_selected_gifts_set = set(category_selected_gifts)
for selected_gift in category_selected_gifts_set:
    freq[selected_gift] = category_selected_gifts.count(selected_gift)
sorted_freq = sorted(freq.items(), key = operator.itemgetter(1))
# printing for debugging
# print(sorted_freq)
# print(sorted_freq[-3:])
recommended_gift = random.choice(sorted_freq[-3:])[0]
print("--------------------------------------------------------------------------")
print("recommended gift: ", recommended_gift)
self.output_msg(gift_category)
```

(2) Enter favorite gift and MBTI type

- Input the user's MBTI and a favorite gift with its category

- Add pairs of category and gift data within the dataset that match the entered MBTI.

- I applied File Out system, Function, Condition and Class.

```
def input_gift(self):
    fp_e_w = open("present/present_e.txt", "a", encoding="utf8")
    fp_i_w = open("present/present_i.txt", "a", encoding="utf8")
    fp_n_w = open("present/present_n.txt", "a", encoding="utf8")
    fp_s_w = open("present/present_s.txt", "a", encoding="utf8")
    fp_f_w = open("present/present_f.txt", "a", encoding="utf8")
    fp_t_w = open("present/present_t.txt", "a", encoding="utf8")
    fp_p_w = open("present/present_p.txt", "a", encoding="utf8")
    fp_j_w = open("present/present_j.txt", "a", encoding="utf8")

    mbti_ = self.enter_mbti()
    preferred_gift = input("Enter your favorite gift with the category (category,gift) --> ")
    line = "\n" + preferred_gift
```

```
    if mbti_[0] == 'e':
        fp_e_w.write(line)
    else:
        fp_i_w.write(line)
    if mbti_[1] == 'n':
        fp_n_w.write(line)
    else:
        fp_s_w.write(line)
    if mbti_[2] == 'f':
        fp_f_w.write(line)
    else:
        fp_t_w.write(line)
    if mbti_[3] == 'p':
        fp_p_w.write(line)
    else:
        fp_j_w.write(line)

    fp_e_w.close()
    fp_i_w.close()
    fp_n_w.close()
    fp_s_w.close()
    fp_f_w.close()
    fp_t_w.close()
    fp_p_w.close()
    fp_j_w.close()
```

(3) Providing a gift card message

- Input the category of a gift, previously inputted from the user in input_gift function.

- Print a gift card message corresponding to the category of a gift.

- I applied Dictionary, If-else condition, In Operator, Function, Class.

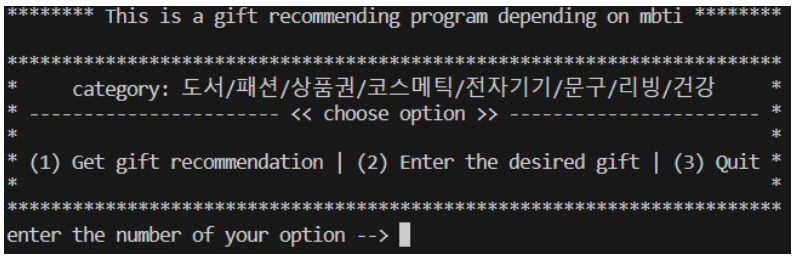
```
def output_msg(self, gift_category):
    category_messages = {
        "도서": "기억에 오래 남는 책이 되길..",
        "패션": "잘 어울릴 것 같아 선물합니다!",
        "상품권": "필요한 곳에 유용히 잘 쓰세요.",
        "코스메틱": "향기로운 일상을 위한 선물!",
        "전자기기": "바쁜 일상에 함께 하세요.",
        "문구": "생각과 기억을 기록하세요.",
        "리빙": "보금자리를 위한 선물을 준비했어요.",
        "건강": "건강이 최고!"
    }

    if gift_category in category_messages:
        print("gift card message: ", category_messages[gift_category])
    else:
        print("카테고리를 찾을 수 없습니다. 올바른 카테고리를 입력하세요.")
```

4) Test Results

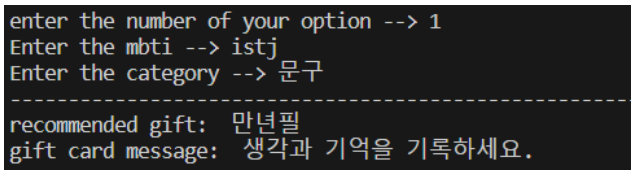
(0) Backbone of the program

- At the beginning of the program execution, it displays an explanation of the program. Throughout the program execution (until option 3 for termination is selected), it repetitively shows the list of categories and options, prompting the user to input their choice of option.

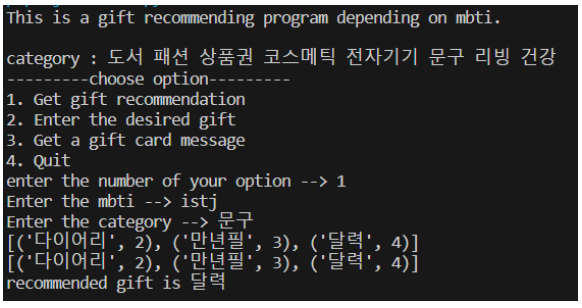


(1) Gift recommendations based on MBTI types & category

- When the user inputs their MBTI and gift category, the program outputs a recommended gift. The recommended gift is randomly selected from the top 3 gifts with the highest frequency in the dataset that matches the input criteria.

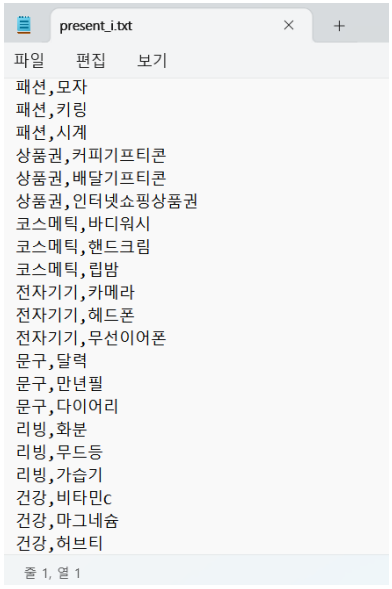


code for debugging >>

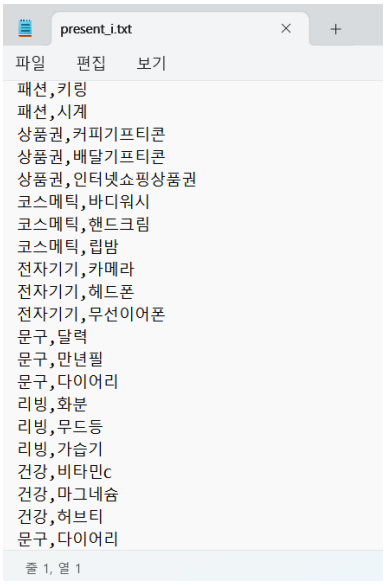


(2) Enter favorite gift and MBTI type

- By using out file pointer, add the entered category and gift data to the data files corresponding to the inputted MBTI. Utilize the write() function for this task, and include "₩n" for line breaks.



--------->




```
This is a gift recommending program depending on mbti.

category : 도서 패션 상품권 코스메틱 전자기기 문구 리빙 건강
---------choose option---------
1. Get gift recommendation
2. Enter the desired gift
3. Get a gift card message
4. Quit
enter the number of your option --> 2
Enter the mbti --> infp
Enter your favorite gift with the category (category,gift) --> 문구,다이어리
```

5. Changes in Comparison to the Plan

1) simultaneous gift recommendation and gift card message display

- Originally, I had planned to implement the gift recommendation and gift card message display as separate features.

- In the final implementation, I have integrated the functionality to simultaneously recommend a gift and display the corresponding gift card message for the selected category.

- Reason : To eliminate the inconvenience of redundant user input and for a more cohesive program flow.

6. Lessons Learned & Feedback

It was a valuable opportunity to apply the overall concepts learned in the 'Python programming and practical' class in designing and implementing a program. One regrettable aspect is that the project planning occurred before the final exam period, so I couldn't incorporate the content covered in the later part of the lecture.