

알게콘 부자되고싶조 프로그램 알고리즘 정리

1. 파이썬 모듈이란? tkinter 모듈

우리가 복잡한 프로그램을 작성하기 위해서 필요한 모든 과정을 직접 만들어야 한다면 어떤 모습이 될까요? 전체적인 모습에서부터 작은 기능 하나하나까지 모두 구상해서, 만들고, 오류를 수정해서 한 곳에 모아두면 또 오류가 생기고... 더구나, 또 다른 프로그래머는 나와 비슷한 기능을 하는 프로그램을 만들면서 똑같은 시행착오를 답습할테구요.

그래서, 이런 문제를 해결하기 위해 대부분의 프로그래밍 언어에서는 모듈이라는 개념을 사용합니다. 모듈은 프로그램의 꾸러미라고 생각하시면 되지요.

이렇게 파이썬에서는 좋은 기능들을 모듈로 묶어서 자체적으로 제공해 준답니다. 파이썬 뿐만이 아니라 대부분의 언어에서 이런 식으로 프로그래밍을 편리하게 할 수 있도록 지원해주지요.

파이썬에서 모듈(module)이란 **하나의 파이썬 파일(.py 파일)**을 의미합니다.

또한 패키지(package)란 이러한 **모듈들이 여러 개 모여있는 것**을 가리킵니다.

여러 개의 파이썬 파일이 폴더에 따라 나뉘어 있는 하나의 큰 폴더를 생각하면 이해하기가 쉬울 것입니다.

파이썬이 지금처럼 많은 사용자들을 보유할 수 있었던 이유가 바로 모듈과 패키지 때문입니다.

다른 사람들이 만든 코드를 간단하게 사용할 수 있으며, 배포 또한 손쉽기 때문에 단시간에 크게 성장할 수 있었습니다.

모듈 사용법 1

파이썬에서 모듈을 사용하기 위해서는 우선 import 문을 사용하여 해당 모듈을 아래와 같이 import 해야만 합니다.

```
import tkinter
import tkinter.ttk
from dataclasses import dataclass
```

이때 import 하고자 하는 모듈의 확장자는 제외하고 파일의 이름만을 import 키워드 뒤에 나열합니다.

이처럼 모듈을 import 문을 사용하여 가져오게 되면 모듈의 모든 내용이 작성 중인 코드상에 그대로 옮겨진다고 생각하면 됩니다.

코드에서 모듈에 저장된 변수나 함수를 사용하려면 import 한 모듈의 이름을 적은 후 온점(.)을 찍고 나서 사용하고자 하는 변수나 함수의 이름을 적으면 됩니다.

2. 윈도우

윈도우 창을 띄우기 위한 주 2개 명령어가 존재

1. 생성 명령어

window=tkinter.Tk()

윈도우명이 'window'인 가장 상위 레벨의 윈도우 창 생성시키는 명령어

```
28 window=tkinter.Tk() |
```

2. 반복 명령어

window.mainloop()

‘window’라는 윈도우 창을 해당 파이썬 실행이 종료될 때까지 띄워놓는, 즉 반복해서 해당 윈도우창을 실행시키는 명령어

위 두 명령어, 생성~반복 명령어 사이에 위젯을 생성하고 적용시키는 명령어를 적는다.

```
323 window.mainloop() # 반복
324 # ...
```

window.title("가계부")

윈도우 창 “window”의 제목을 문자열 “가계부”로 설정하는 명령어

window.geometry("700x700+600+100")

윈도우 창 “window”을 너비 700, 높이 700으로 설정하고

컴퓨터 화면 내의 x 좌표 600, y 좌표 100에 위치시키겠다.

window.resizable(True, True)

첫 번째 bool 형 인자값은

윈도우창 “window”의 너비를 사용자가 조정 가능하게 할 것인지.

가능하게 할 것이라면 True,

고정시킬 것이라면 False로 설정.

두 번째 bool 형 인자값은 높이에 대해 위와 동일.

(True=1, False=0을 의미하여 상수를 입력해도 적용이 가능합니다.)

window.option_add("*Font", "돋움 15")

윈도우 창 “window”에 옵션을 추가.

해당 윈도우 창 내의 글의 글꼴을 돋움으로, 글씨 크기를 15로 설정.

```

83         window2=tkinter.Tk() # 생성
84         window2.title(text) # 윈도우 창의 제목
85         temp T date V = mv date V[text+1

93     window2.geometry("700x700+900+100") # 너비*높이 + x좌표 + y좌표
94     window2.resizable(True, True)
95     window2.option_add("*Font", "돋움 15")
96
220
227         window2.mainloop()
228
204     def fx_show():
205         window3=tkinter.Tk() # 생성
206         window3.title("가계부 기록 열람")
207
208         window3.geometry("700x1000+900+100") # 너비*높이 + x좌표 + y좌표
209         window3.resizable(True, True)
210         window3.option_add("*Font", "돋움 15")
211

```

3. 위젯이란?

entry: 입력창

button

.pack() 화면에 나타내기

Label

```

39     label1=tkinter.Label(window, text="<< 가계부 프로그램 >>\n", width=30, height=5, bd='0', fg="blue", relief="solid")
40     label2=tkinter.Label(window, text="본 프로그램은 지출만을 기록합니다. \n수요자가 많을 시 소득 기록 기능이 추가될 수 있습니다.\n")
41     label3=tkinter.Label(window, text="각 입력창에 연도, 월, 일을 입력한 뒤 [일자 추가] 버튼을 눌러주세요.")
42
43
44     label1.pack() # 위젯을 배치할 수 있습니다.
45     label2.pack()
46     label3.pack()
47

```

‘window’라는 이름의 윈도우 창에 문자열 text를 출력한다. 이 때 해당 문자열의 너비는 width, 높이는 height, 배경색인가 뭐지이거 bd, 글자색 fg, relief는 어찌고

```

97         label4=tkinter.Label(window2, text="지출 항목을 선택하세요.")
98         label4.pack() # 위젯을 배치할 수 있습니다.
99

```

```

lab = tkinter.Label(window2)
lab.config(text = temp.I_type)
lab.pack()

cl_label=tkinter.Label(window2, text="지출 내용을 입력해주세요.\n", width=30, height=5, bd='0', fg="blue", relief="solid")
cl_label.pack()

```

'window2'라는 이름의 윈도우 창에 문자열 text를 출력한다. 나머지는 동일.

```

---
212 |         label4=tkinter.Label(window3, text="날짜 | 항목 | 금액 | 내용")
213 |         label4.pack() # 위젯을 배치할 수 있습니다
214 |

```

```

215 |
216 |         label_list = []
217 |
218 |         for i in range (len(info_date_Y)):

```

```

label_list = []

```

```

    for i in range (len(info_date_Y)):
        print("\n"+str(i)+"번째 for 문")
        print("len(my_date_Y)=", len(my_date_Y))
        print("info_date_Y["+str(i)+"] = "+str(info_date_Y[i]))
        print("info_date_M["+str(i)+"] = "+str(info_date_M[i]))
        print("info_date_D["+str(i)+"] = "+str(info_date_D[i]))
        print("my_type["+str(i)+"] = "+ my_type[i])
        print("my_item["+str(i)+"] = "+ my_item[i])
        print("my_money["+str(i)+"] = "+ str(my_money[i]))
        print("\n")
        if(int(my_money[i]) >= 10000 and int(my_money[i]) < 30000):
            my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "orange")
            elif(int(my_money[i]) >= 30000 and int(my_money[i]) < 100000):
                my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "tomato")
            elif(int(my_money[i]) >= 100000):
                my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "red")
            else :
                my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "blue")

            label_list.append(my_label)

    for i in range(0, len(info_date_Y), 1):
        label_list[i].pack()
        print("label_list[i].pack()", label_list[i])

```

Listbox

```
63 | # my_date 리스트박스
64 | my_date = tkinter.Listbox(window, selectmode='extended', height=0)
65 | my_date Y = 1
```

'my_date'라는 리스트박스 선언. 'window'라는 이름의 윈도우창에 출력.

기본 높이는 크기 0. 즉, 보이지 않는, 화면에 나타나지 않는 상태.

selectmode

'extended'는 메뉴가 추가되었을 때 해당 창에 자동으로 확장되는 모드로 설정했다는 뜻.

```
my_date.insert('end', Y_entry.get()+"/"+M_entry.get()+"/"+D_entry.get()) # 추가
my_date Y.append(Y_entry.get())
```

```
59 | my_date.pack()
60 |
```

리스트박스.curselection()[0]

```
80 | text = my_date.curselection()[0]
```

Entry

입력창.

```
70 | # 입력창 entry
71 | Y_entry=tkinter.Entry(window) #속성
72 | #Y_entry.bind("<Return>", fx_date) #입력하고 enter 누르면 fx_date 함수 실행
73 | M_entry=tkinter.Entry(window) #속성
74 | D_entry=tkinter.Entry(window) #속성
75 | Y_entry.pack()
76 | M_entry.pack()
77 | D_entry.pack()
```

'window'라는 이름의 윈도우 창에 띄운다. 나타낸다. 출력한다.

```
1(Y_entry.get())
1(M_entry.get())
1(D_entry.get())
```

```

130
131 # 지출 내용 입력받는 entry 생성 (1)
132 cl_entry=tkinter.Entry(window2)
133 cl_entry.pack()
134

```

combobox

내기

```

101 # Combobox 생성 및 출력
102 type_list = ["식품비", "저축", "통신비", "교통비", "문화생활비", "의류미용비", "교육비", "의료건강비", "기타"]
103 cb = tkinter.ttk.Combobox(window2)
104 cb.config(values = type_list)
105 cb.pack()
106
107

```

button

1. 항목 선택 btn

내

```

108 # [항목 선택] 버튼 정의
109 btn = tkinter.Button(window2)
110 btn.config(text = "항목 선택")
111
222
223 # [항목 선택] 버튼 활성화. 버튼 클릭 시 click 함수 실행
224 btn.config(command = click)
225 btn.pack()
226
227

```

2. 지출 내용 입력 버튼 btn2

내

```

126 # [지출 내용 입력] 버튼 정의
127 btn2 = tkinter.Button(window2)
128 btn2.config(text = "지출 내용 입력")
129
215
216 # [지출 내용 입력] 버튼 출력, 버튼 클릭 시 fx_what 함수 실행
217 btn2.config(command = fx_what)
218 btn2.pack()
219
220

```

3. 지출 금액 입력 버튼 btn3

내

```
145
146         # [지출 금액 입력] 버튼 정의
147         btn3 = tkinter.Button(window2)
148         btn3.config(text = "지출 금액 입력")
149
150
209
210         # [지출 금액 입력] 버튼 출력, 버튼 클릭 시 fx_money 함수 실행
211         btn3.config(command = fx_money)
212         btn3.pack()
213
214
```

4. 일자 추가 버튼 lb_button

내

```
185         # 버튼 누르면 listbox 추가
186         lb_button = tkinter.Button(window)
187         lb_button.config(overrelief="solid", command=fx_date)
188         lb_button.config(repeatdelay=1000, repeatinterval=100)
189         lb_button.config(width=20, height=1, bd='0.5', bg='OliveDrab1')
190         lb_button.config(text = "일자 추가")
191         lb_button.pack() # 위젯 배치
192
```

5. 해당 일자 기록하기 버튼 button

내

```
194         # 버튼 누르면 fx_contents 실행
195         button = tkinter.Button(window)
196         button.config(overrelief="solid", command=fx_contents)
197         button.config(repeatdelay=1000, repeatinterval=100)
198         button.config(width=20, height=1, bd='0.5', bg='OliveDrab1')
199         button.config(text = "해당 일자 기록하기")
200         button.pack() # 위젯 배치
201
```

6. 최종 입력 확인 버튼 Y_btn

내

```

154
155 # [확인] 버튼 정의
156 Y_btn = tkinter.Button(window2)
157 Y_btn.config(text = "확인")
158
159
204
205 # [확인] 버튼 출력 , 버튼 클릭 시 fx_Y 함수 실행
206 Y_btn.config(command = fx_Y)
207 Y_btn.pack()
208
209

```

7. 기록 열람하기 버튼 show_button

내

```

240
247 # 버튼 누르면 fx_show 실행
248 show_button = tkinter.Button(window)
249 show_button.config(overrelief="solid", command=fx_show)
250 show_button.config(repeatdelay=1000, repeatinterval=100)
251 show_button.config(text = "기록 열람하기")
252 show_button.pack() # 위젯 배치
253

```

8. 해당일자 삭제하기 버튼 del_button

내

```

269 del_button = tkinter.Button(window)
270 del_button.config(overrelief="solid", command=fx_delete)
271 del_button.config(repeatdelay=1000, repeatinterval=100)
272 del_button.config(width=20, height=1, bd='0.5', bg = 'tomato2')
273 del_button.config(text = "해당 일자 삭제하기")
274 del_button.pack() # 위젯 배치
275

```


Named colour chart

snow	deep sky blue	gold	seashell3	SlateBlue2	LightBlue3	SpringGreen2	DarkGoldenrod1	brown4	pink3	purple1	gray26	gray84
ghost white	sky blue	light goldenrod	seashell4	SlateBlue3	LightBlue4	SpringGreen3	DarkGoldenrod2	salmon1	pink4	purple2	gray27	gray85
white smoke	light sky blue	goldenrod	AntiqueWhite1	SlateBlue4	LightCyan2	SpringGreen4	DarkGoldenrod3	salmon2	LightPink1	purple3	gray28	gray86
gainsboro	steel blue	dark goldenrod	AntiqueWhite2	RoyalBlue1	LightCyan3	green2	DarkGoldenrod4	salmon3	LightPink2	purple4	gray29	gray87
floral white	light steel blue	rosy brown	AntiqueWhite3	RoyalBlue2	LightCyan4	green3	RosyBrown1	salmon4	LightPink3	MediumPurple1	gray30	gray88
old lace	light blue	indian red	AntiqueWhite4	RoyalBlue3	PaleTurquoise1	green4	RosyBrown2	LightSalmon2	LightPink4	MediumPurple2	gray31	gray89
linen	powder blue	saddle brown	bisque2	RoyalBlue4	PaleTurquoise2	chartreuse2	RosyBrown3	LightSalmon3	PaleVioletRed1	MediumPurple3	gray32	gray90
antique white	pale turquoise	sandy brown	bisque3	blue3	PaleTurquoise3	chartreuse3	RosyBrown4	LightSalmon4	PaleVioletRed2	MediumPurple4	gray33	gray91
papaya whip	dark turquoise	dark salmon	bisque4	blue4	PaleTurquoise4	chartreuse4	IndianRed1	orange2	PaleVioletRed3	thistle1	gray34	gray92
blanched almond	medium turquoise	salmon	PeachPuff2	DodgerBlue2	CadetBlue1	OliveDrab1	IndianRed2	orange3	PaleVioletRed4	thistle2	gray35	gray93
bisque	turquoise	light salmon	PeachPuff3	DodgerBlue3	CadetBlue2	OliveDrab2	IndianRed3	orange4	maroon1	thistle3	gray36	gray94
peach puff	cyan	orange	PeachPuff4	DodgerBlue4	CadetBlue3	OliveDrab4	IndianRed4	DarkOrange1	maroon2	thistle4	gray37	gray95
navajo white	light cyan	dark orange	NavajoWhite2	SteelBlue1	CadetBlue4	DarkOliveGreen1	sienna1	DarkOrange2	maroon3		gray38	gray96
lemon chiffon	cadet blue	coral	NavajoWhite3	SteelBlue2	turquoise1	DarkOliveGreen2	sienna2	DarkOrange3	maroon4		gray39	gray97
mint cream	medium aquamarine	light coral	NavajoWhite4	SteelBlue3	turquoise2	DarkOliveGreen3	sienna3	DarkOrange4	VioletRed1		gray40	gray98
azure	aquamarine	tomato	LemonChiffon2	SteelBlue4	turquoise3	DarkOliveGreen4	sienna4	coral1	VioletRed2		gray41	gray99
alice blue	dark green	orange red	LemonChiffon3	DeepSkyBlue2	turquoise4	khaki1	burlywood1	coral2	VioletRed3		gray42	gray100
lavender	dark olive green	red	LemonChiffon4	DeepSkyBlue3	cyan2	khaki2	burlywood2	coral3	VioletRed4		gray43	gray101
lavender blush	dark sea green	hot pink	cornsilk2	DeepSkyBlue4	cyan3	khaki3	burlywood3	coral4	magenta2		gray44	gray102
misty rose	sea green	deep pink	cornsilk3	SkyBlue1	cyan4	khaki4	burlywood4	tomato2	magenta3		gray45	gray103
dark slate gray	medium sea green	pink	cornsilk4	SkyBlue2	DarkSlateGray1	LightGoldenrod1	wheat1	tomato3	magenta4		gray46	gray104
dim gray	light sea green	light pink	ivory2	SkyBlue3	DarkSlateGray2	LightGoldenrod2	wheat2	tomato4	orchid1		gray47	gray105
slate gray	pale green	pale violet red	ivory3	SkyBlue4	DarkSlateGray3	LightGoldenrod3	wheat3	OrangeRed2	orchid2		gray48	gray106
light slate gray	spring green	maroon	ivory4	LightSkyBlue1	DarkSlateGray4	LightGoldenrod4	wheat4	OrangeRed3	orchid3		gray49	gray107
gray	lawn green	medium violet red	honeydew2	LightSkyBlue2	aquamarine2	LightYellow2	tan1	OrangeRed4	orchid4		gray50	gray108
light gray	medium spring green	violet red	honeydew3	LightSkyBlue3	aquamarine4	LightYellow3	tan2	red2	plum1		gray51	gray109
midnight blue	green yellow	medium orchid	honeydew4	LightSkyBlue4	DarkSeaGreen1	LightYellow4	tan4	red3	plum2		gray52	gray110
navy	lime green	dark orchid	LavenderBlush2	SlateGray1	DarkSeaGreen2	yellow2	chocolate1	red4	plum3		gray53	gray111
cornflower blue	yellow green	dark violet	LavenderBlush3	SlateGray2	DarkSeaGreen3	yellow3	chocolate2	DeepPink2	plum4		gray54	gray112
dark slate blue	forest green	blue violet	LavenderBlush4	SlateGray3	DarkSeaGreen4	yellow4	chocolate3	DeepPink3	MediumOrchid1		gray55	gray113
slate blue	olive drab	purple	MistyRose2	SlateGray4	SeaGreen1	gold2	firebrick1	DeepPink4	MediumOrchid2		gray56	gray114
medium slate blue	dark khaki	medium purple	MistyRose3	LightSteelBlue1	SeaGreen2	gold3	firebrick2	HotPink1	MediumOrchid3		gray57	gray115
light slate blue	khaki	thistle	MistyRose4	LightSteelBlue2	SeaGreen3	gold4	firebrick3	HotPink2	MediumOrchid4		gray58	gray116
medium blue	pale goldenrod	snow2	azure2	LightSteelBlue3	PaleGreen1	goldenrod1	firebrick4	HotPink3	DarkOrchid1		gray59	gray117
royal blue	light goldenrod yellow	snow3	azure3	LightSteelBlue4	PaleGreen2	goldenrod2	brown1	HotPink4	DarkOrchid2		gray60	gray118
blue	light yellow	snow4	azure4	LightBlue1	PaleGreen3	goldenrod3	brown2	pink1	DarkOrchid3		gray61	gray119
dodger blue	yellow	seashell2	SlateBlue1	LightBlue2	PaleGreen4	goldenrod4	brown3	pink2	DarkOrchid4		gray62	gray120

4. class 란?

구조체란?

파이썬에는 구조체가 없을 뿐더러 클래스 또한 데이터 타입을 지정할 수 없었음.

파이썬 3.7부터 dataclass를 지원하여

다음과 같이 class를 이용해 구조체 형태로 정의할 수 있다.

```
--
14 class MYINFO:
15     I_date_Y:int = 22
16     I_date_M:int = 11
17     I_date_D:int = 20
18     I_moeny:int = 0
19     I_item:str = ''
20     I_type:str = ''
21
```

5. 알고리즘

5-1. 변수

전역변수 list

전역변수 list 6개 선언

```
6   # Data
7   info_date_Y = []
8   info_date_M = []
9   info_date_D = []
10  my_money = []
11  my_item = []
12  my_type = []
```

class 안의 int형 4개, string형 2개

temp

temp라는 MYINFO

```
23
24   temp = MYINFO()
25
26
27
28
29
30
31
32
33
34   my_date = {}
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53   my_date_Y.append(Y_entry.get())
54   my_date_M.append(M_entry.get())
55   my_date_D.append(D_entry.get())
56
57
58
59
60
61
62
63
64
65   my_date_Y = []
66   my_date_M = []
67   my_date_D = []
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85   temp.I_date_Y = my_date_Y[text]
86   temp.I_date_M = my_date_M[text]
87   temp.I_date_D = my_date_D[text]
```

click() 함수에서 콤보박스 'cb'에 입력된 문자열을 항목명에 넣음.

```
115   temp.I_type = cb.get() # 선택한 콤보박스 항목값 temp.I_type에 넣기
```

fx_money() 함수에서 입력창 'cl2_entry'에 입력된 수를 금액 값에 넣음.

```
def fx_money():
    temp.I_money = c12_entry.get() # 입력한 지출 금액 temp.I_money에 넣기
```

temp에 담긴 모든 값을 전역변수 list에 모두 삽입.

지금까지 바로바로 전역변수 list에 넣었어도 됐지만

혹시 모를 오류를 줄이기 위해서(즉, 안정성을 위해서)

중간에 temp라는 변수에 임시로 저장해둔 뒤

모든 입력이 완료되었을 때 한꺼번에 불러와서 넣어 저장해준 것임.

```
def fx_Y():
    info_date_Y.append(temp.I_date_Y) # 맨 앞에서 선언한 리스트에 temp.I_date_Y의 값 추가
    info_date_M.append(temp.I_date_M)
    info_date_D.append(temp.I_date_D)
    my_type.append(temp.I_type)
    my_item.append(temp.I_item)
    my_money.append(temp.I_money)
```

5-2. 함수

1. fx_contents()

역변수

2. click()

역

```
temp.I_type = cb.get() # 선택한 콤보박스 항목값 temp.I_type에 넣기
```

2-1. fx_what()

역

2-2. fx_money()

역

2-3. fx_Y()

역

3. fx_show()

역

4. fx_delete()

역

```
257 def fx_delete():
258     text = my_date.curselection()[0]
259     my_date.delete(text)
260     # 0, 1, 2 ...
261     del info_date_Y[text]
262     del info_date_M[text]
263     del info_date_D[text]
264     del my_type[text]
265     del my_item[text]
266     del my_money[text]
```

```
import tkinter
import tkinter.ttk
from dataclasses import dataclass
# Data
info_date_Y = []
info_date_M = []
info_date_D = []
my_money = []
my_item = []
my_type = []
class MYINFO:
    I_date_Y:int = 22
    I_date_M:int = 11
    I_date_D:int = 20
    I_moeny:int = 0
    I_item:str = ''
    I_type:str = ''

temp = MYINFO()
print("temp 세팅")
window=tkinter.Tk()
window.title("가계부") # 윈도우 창의 제목
window.geometry("700x700+600+100") # 너비*높이 + x 좌표 + y 좌표
window.resizable(True, True)
window.option_add("*Font", "돋움 15")
```

```

label1=tkinter.Label(window, text="<< 가계부 프로그램 >>\n", width=30, height=5, bd='0',
fg="blue", relief="solid")
label2=tkinter.Label(window, text="본 프로그램은 지출만을 기록합니다. \n 수요자가 많을 시 소득
기록 기능이 추가될 수 있습니다.\n")
label3=tkinter.Label(window, text="각 입력창에 연도, 월, 일을 입력한 뒤 [일자 추가] 버튼을
눌러주세요.")
label1.pack()
label2.pack()
label3.pack()
# 리스트박스[date_count] 에 date 추가 함수
def fx_date():
    my_date.insert('end', Y_entry.get()+"/"+M_entry.get()+"/"+D_entry.get()) # 추가
    my_date_Y.append(Y_entry.get())
    my_date_M.append(M_entry.get())
    my_date_D.append(D_entry.get())
    print("my_date [index] = ", Y_entry.get()+"/"+M_entry.get()+"/"+D_entry.get())
    my_date.pack()

# my_date 리스트박스
my_date = tkinter.Listbox(window, selectmode='extended', height=0)
my_date_Y = []
my_date_M = []
my_date_D = []
# 입력창 entry
Y_entry=tkinter.Entry(window) #속성
#Y_entry.bind("<Return>", fx_date) #입력하고 enter 누르면 fx_date 함수 실행
M_entry=tkinter.Entry(window) #속성
D_entry=tkinter.Entry(window) #속성
Y_entry.pack()
M_entry.pack()
D_entry.pack()
def fx_contents():
    text = my_date.curselection()[0]
    # 0, 1, 2 ...

    window2=tkinter.Tk() # 생성
    window2.title(text) # 윈도우 창의 제목
    temp.I_date_Y = my_date_Y[text]
    temp.I_date_M = my_date_M[text]
    temp.I_date_D = my_date_D[text]
    print("text =", text)
    print("temp.I_date_Y =", temp.I_date_Y)
    print("temp.I_date_M =", temp.I_date_M)
    print("temp.I_date_D =", temp.I_date_D)

    window2.geometry("700x700+900+100") # 너비*높이 + x 좌표 + y 좌표
    window2.resizable(True, True)
    window2.option_add("*Font", "돋움 15")
    label14=tkinter.Label(window2, text="지출 항목을 선택하세요.")
    label14.pack() # 위젯을 배치할 수 있습니다.
    # Combobox 생성 및 출력
    type_list = ["식품비", "저축", "통신비", "교통비", "문화생활비", "의류미용비", "교육비",
"의료건강비", "기타"]
    cb = tkinter.ttk.Combobox(window2)
    cb.config(values = type_list)
    cb.pack()
    # [항목 선택] 버튼 정의
    btn = tkinter.Button(window2)
    btn.config(text = "항목 선택")

    # [항목]값 불러와서 출력
    def click():

```

```

temp.I_type = cb.get() # 선택한 콤보박스 항목값 temp.I_type 에 넣기
print("temp.I_type =", temp.I_type)
lab = tkinter.Label(window2)
lab.config(text = temp.I_type)
lab.pack()
cl_label=tkinter.Label(window2, text="지출 내용을 입력해주세요.\n", width=30,
height=5, bd='0', fg="blue", relief="solid")
cl_label.pack()
# [지출 내용 입력] 버튼 정의
btn2 = tkinter.Button(window2)
btn2.config(text = "지출 내용 입력")

# 지출 내용 입력받는 entry 생성 (1)
cl_entry=tkinter.Entry(window2)
cl_entry.pack()
def fx_what():
    temp.I_item = cl_entry.get() # 입력한 지출 내용 temp.I_item 에 넣기
    print("temp.I_item =", temp.I_item)

    la = tkinter.Label(window2, text=temp.I_item) # 지출 내용 출력
    la.pack()
    # 지출 금액 입력받기
    cl2_entry=tkinter.Entry(window2)
    cl2_entry.pack()

    # [지출 금액 입력] 버튼 정의
    btn3 = tkinter.Button(window2)
    btn3.config(text = "지출 금액 입력")

    def fx_money():
        temp.I_money = cl2_entry.get() # 입력한 지출 금액 temp.I_money 에 넣기
        print("temp.I_money =", temp.I_money)
        # [확인] 버튼 정의
        Y_btn = tkinter.Button(window2)
        Y_btn.config(text = "확인")
        def fx_Y():
            info_date_Y.append(temp.I_date_Y) # 맨 앞에서 선언한 리스트에
temp.I_date_Y의 값 추가
            info_date_M.append(temp.I_date_M)
            info_date_D.append(temp.I_date_D)
            my_type.append(temp.I_type)
            my_item.append(temp.I_item)
            my_money.append(temp.I_money)

            # [확인] 버튼 출력 , 버튼 클릭 시 fx_Y 함수 실행
            Y_btn.config(command = fx_Y)
            Y_btn.pack()
        # [지출 금액 입력] 버튼 출력, 버튼 클릭 시 fx_money 함수 실행
        btn3.config(command = fx_money)
        btn3.pack()
        # [지출 내용 입력] 버튼 출력, 버튼 클릭 시 fx_what 함수 실행
        btn2.config(command = fx_what)
        btn2.pack()
    # [항목 선택] 버튼 활성화. 버튼 클릭 시 click 함수 실행
    btn.config(command = click)
    btn.pack()
    window2.mainloop()
# 버튼 누르면 listbox 추가
lb_button = tkinter.Button(window)
lb_button.config(overrelief="solid", command=fx_date)
lb_button.config(repeatdelay=1000, repeatinterval=100)

```

```

lb_button.config(width=20, height=1, bd='0.5', bg='OliveDrab1')
lb_button.config(text = "일자 추가")
lb_button.pack() # 위젯 배치
# 버튼 누르면 fx_contents 실행
button = tkinter.Button(window)
button.config(overrelief="solid", command=fx_contents)
button.config(repeatdelay=1000, repeatinterval=100)
button.config(width=20, height=1, bd='0.5', bg='OliveDrab1')
button.config(text = "해당 일자 기록하기")
button.pack() # 위젯 배치
def fx_show():
    window3=tkinter.Tk() # 생성
    window3.title("가계부 기록 열람")

    window3.geometry("700x1000+900+100") # 너비*높이 + x 좌표 + y 좌표
    window3.resizable(True, True)
    window3.option_add("*Font", "돋움 15")
    label4=tkinter.Label(window3, text="날짜 | 항목 | 금액 | 내용")
    label4.pack() # 위젯을 배치할 수 있습니다

    label_list = []

    for i in range (len(info_date_Y)):
        print("\n"+str(i)+"번째 for 문")
        print("len(my_date_Y)=", len(my_date_Y))
        print("info_date_Y["+str(i)+"] = "+str(info_date_Y[i]))
        print("info_date_M["+str(i)+"] = "+str(info_date_M[i]))
        print("info_date_D["+str(i)+"] = "+str(info_date_D[i]))
        print("my_type["+str(i)+"] = "+ my_type[i])
        print("my_item["+str(i)+"] = "+ my_item[i])
        print("my_money["+str(i)+"] = "+ str(my_money[i]))
        print("\n")
        if(int(my_money[i]) >= 10000 and int(my_money[i]) < 30000):
            my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "orange")
            elif(int(my_money[i]) >= 30000 and int(my_money[i]) < 1000000):
            my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "tomato")
            elif(int(my_money[i]) >= 100000):
            my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "red")
            else :
            my_label = tkinter.Label(window3, text = str(info_date_Y[i]) + "/" +
str(info_date_M[i]) + "/" + str(info_date_D[i]) + " " + my_type[i] + " " +
str(my_money[i]) + " " + my_item[i], fg = "blue")

        label_list.append(my_label)

    for i in range(0, len(info_date_Y), 1):
        label_list[i].pack()
        print("label_list[i].pack()", label_list[i])
# 버튼 누르면 fx_show 실행
show_button = tkinter.Button(window)
show_button.config(overrelief="solid", command=fx_show)
show_button.config(repeatdelay=1000, repeatinterval=100)
show_button.config(text = "기록 열람하기")
show_button.pack() # 위젯 배치
def fx_delete():
    text = my_date.curselection()[0]
    my_date.delete(text)

```

```
# 0, 1, 2 ...
del info_date_Y[text]
del info_date_M[text]
del info_date_D[text]
del my_type[text]
del my_item[text]
del my_money[text]

del_button = tkinter.Button(window)
del_button.config(overrelief="solid", command=fx_delete)
del_button.config(repeatdelay=1000, repeatinterval=100)
del_button.config(width=20, height=1, bd='0.5', bg = 'tomato2')
del_button.config(text = "해당 일자 삭제하기")
del_button.pack() # 위젯 배치
window.mainloop() # 반복
# window 의 윈도우 창을 윈도우가 종료될 때까지 실행
```