Anaconda3-2020.11-Windows-x86\_64 다운로드

https://www.anaconda.com/products/individual/download-success

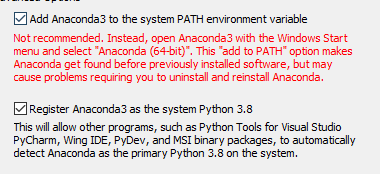
vmware workstation pro

https://www.vmware.com/kr/products/workstation-pro.html

안 다운 받는데?

숨김파일 표시, 아나콘다 인스톨러 관리자 권한으로 실행



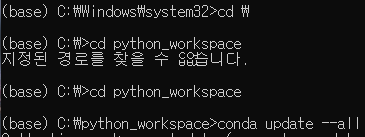


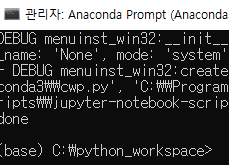
Anaconda Navigator (Anaconda3) 🡪 프로그램들 설치 지원 하는거

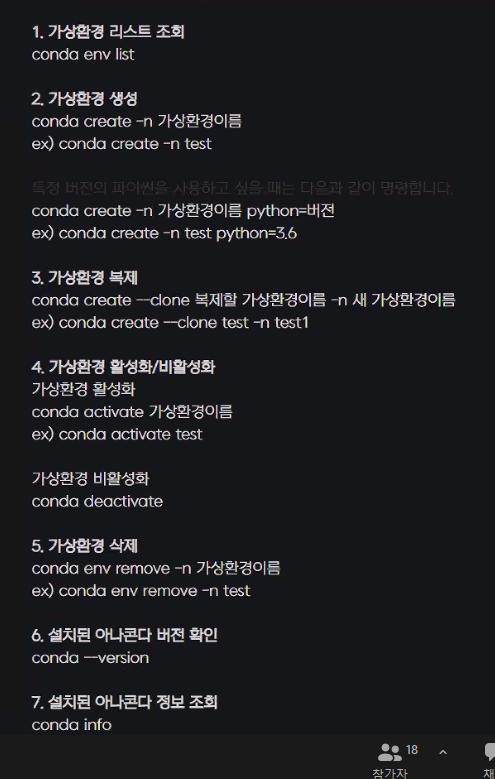
Anaconda Prompt (Anaconda3)

conda list -> 라이브러리 리스트

C 밑에 python\_workspace 생성



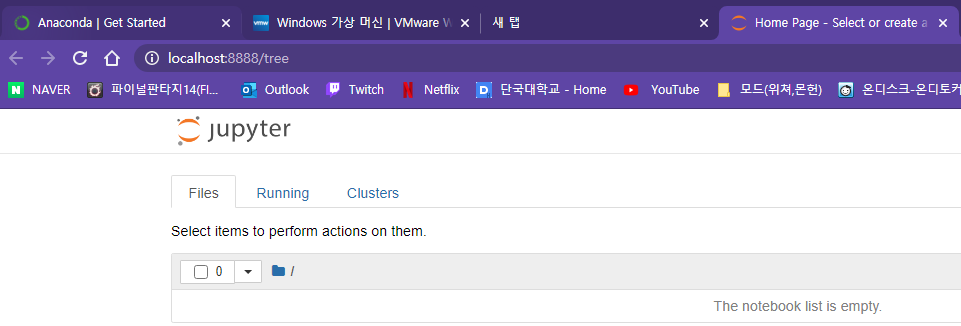
 업데이트 끝남



(base) C:\python\_workspace>conda create -n goott7

(base) C:\python\_workspace>conda activate goott7

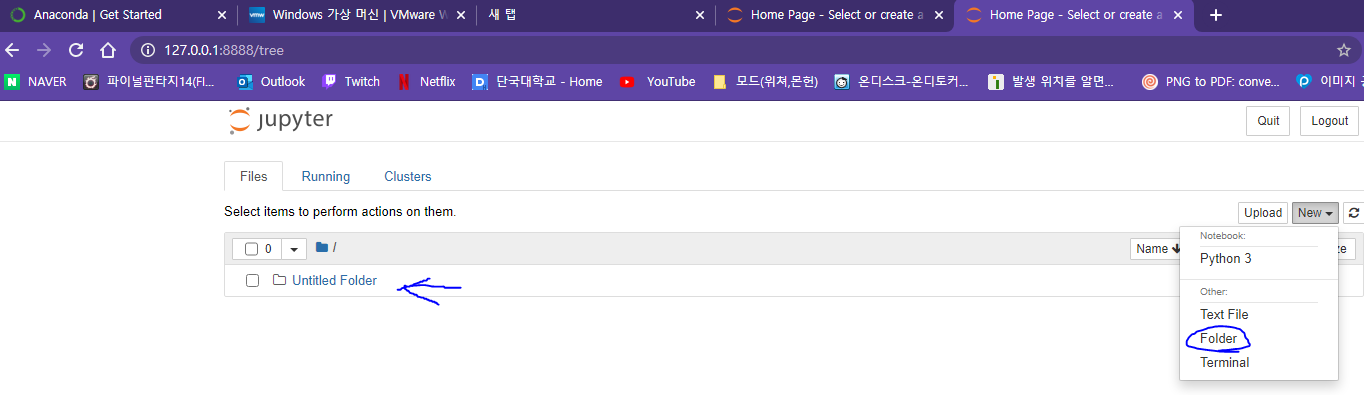
(goott7) C:\python\_workspace>jupyter notebook

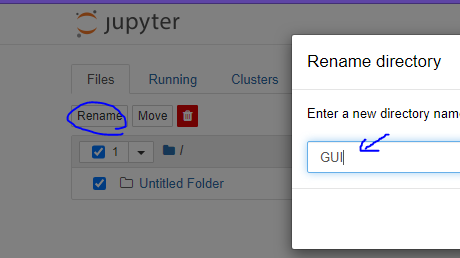


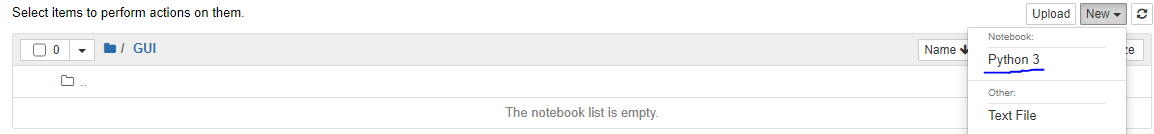
이런 창이 뜸



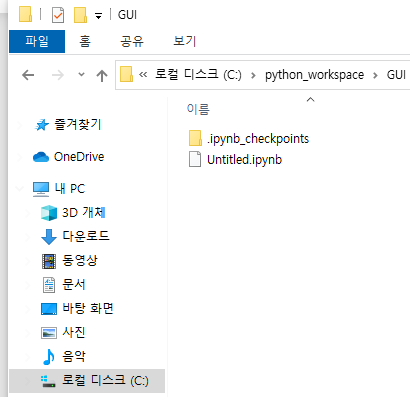
주피터



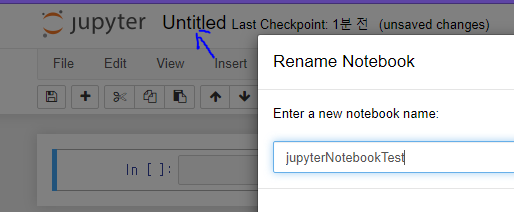




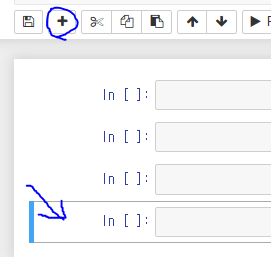
실제 경로에 파일이 생성됨



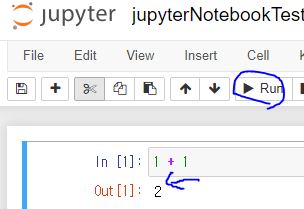
제목 부분을 누르면 변경 가능



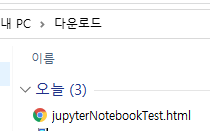
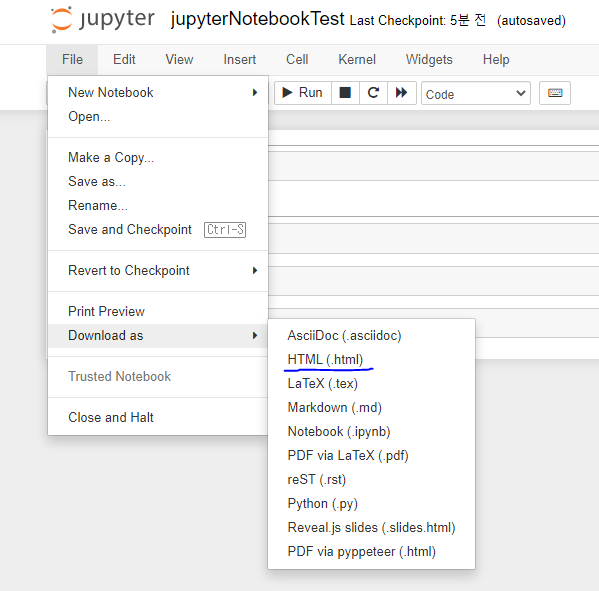
줄 추가 기능

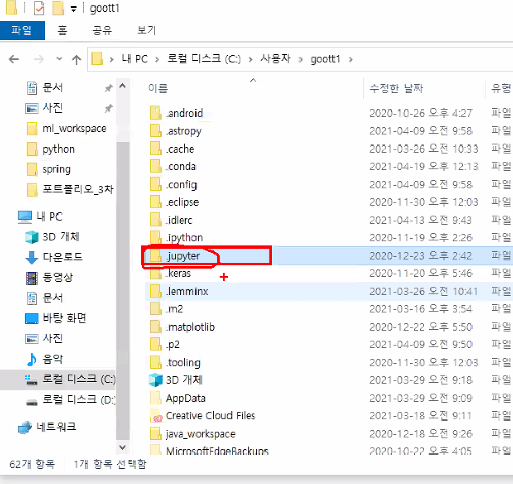


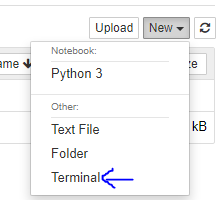
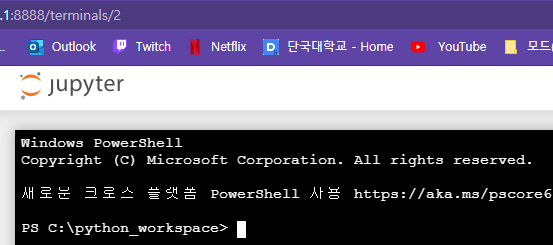
실행, 한 줄씩



HTML로 다운로드





(base) C:\Users\yuiop>cd \

(base) C:\>cd python\_workspace

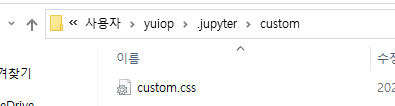
(base) C:\python\_workspace>jupyter notebook --generate-config

Writing default config to: C:\Users\yuiop\.jupyter\jupyter\_notebook\_config.py

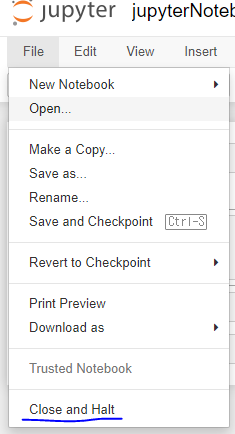
(base) C:\Users\yuiop>jupyter notebook --generate-config

Writing default config to: C:\Users\yuiop\.jupyter\jupyter\_notebook\_config.py

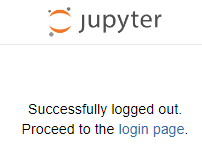
이 경로에 받은 custom.css 를 넣어두기

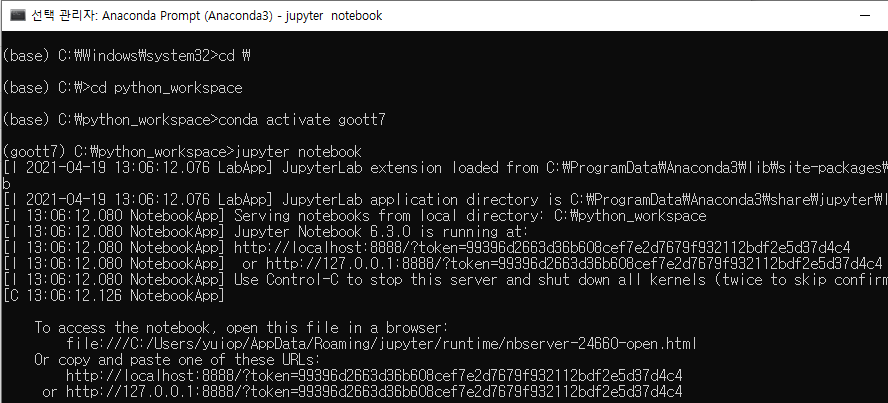


exit

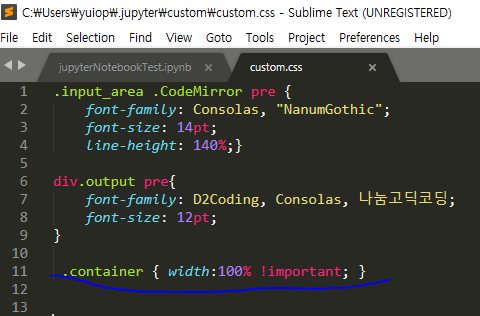


로그아웃

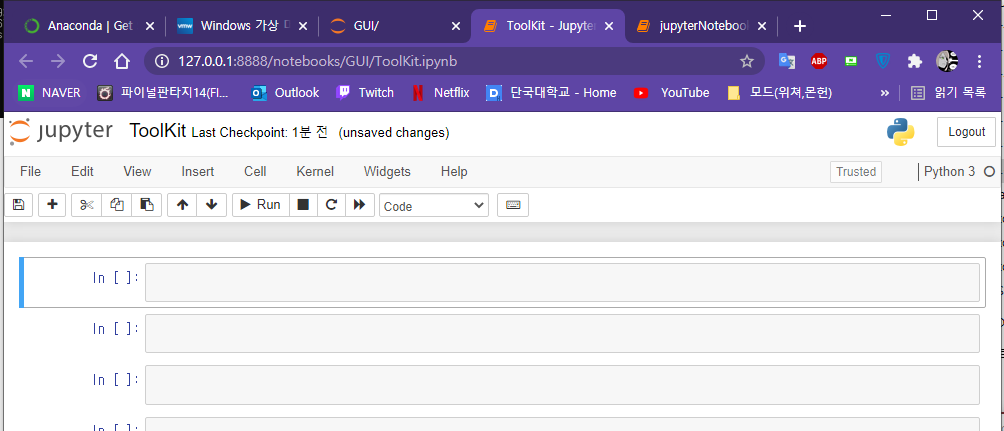




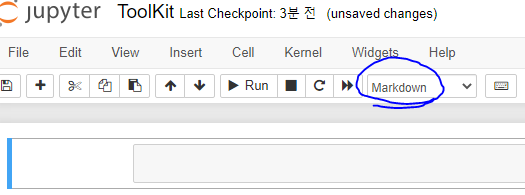
custom.css 파일 수정



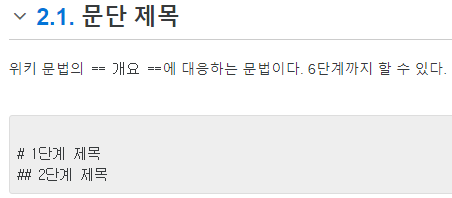
새 파일 ToolKit 생성, css 변경 확인 가능

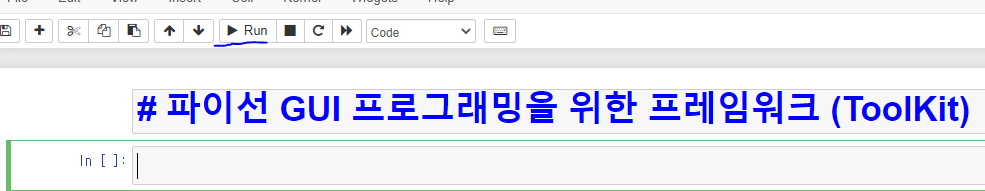


마크다운

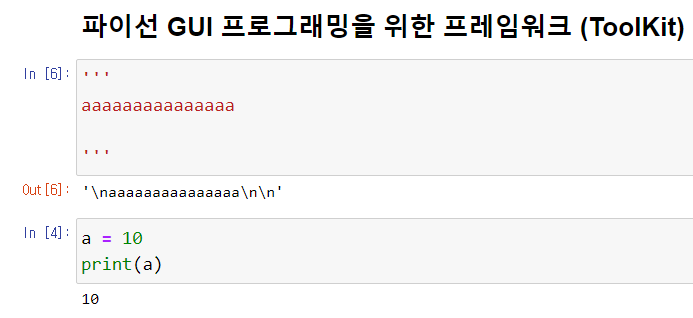


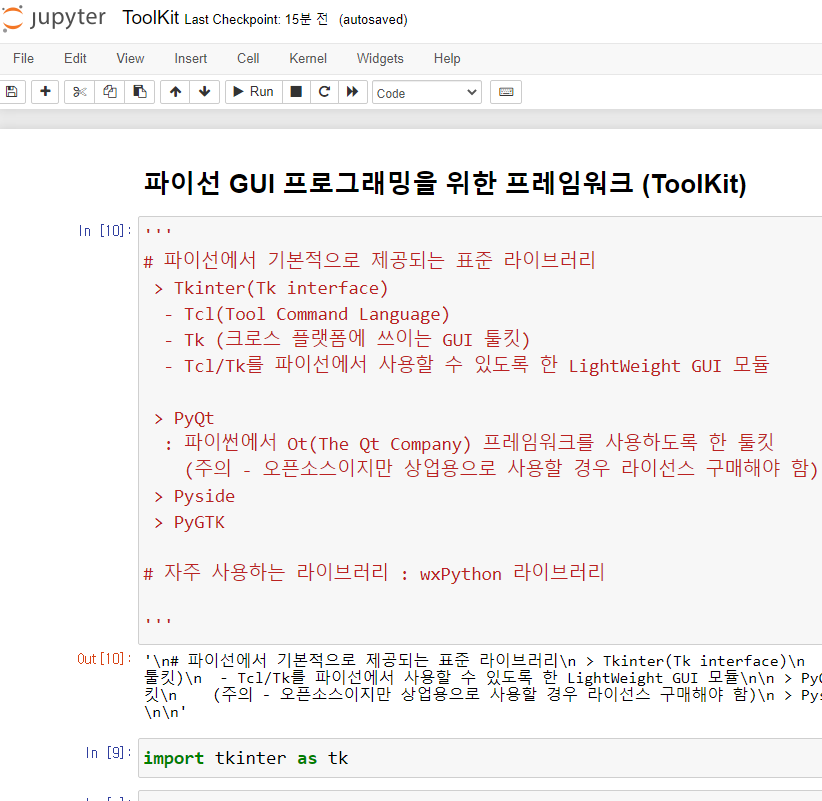
HTML 태그처럼 # 사용





이렇게 사용;;





# 파이선 GUI 프로그래밍을 위한 프레임워크

'''

# 파이선에서 기본적으로 제공되는 표준 라이브러리

> Tkinter(Tk interface)

- Tcl(Tool Command Language)

- Tk (크로스 플랫폼에 쓰이는 GUI 툴킷)

- Tcl/Tk를 파이선에서 사용할 수 있도록 한 LightWeight GUI 모듈

> PyQt

: 파이썬에서 Ot(The Qt Company) 프레임워크를 사용하도록 한 툴킷

(주의 - 오픈소스이지만 상업용으로 사용할 경우 라이선스 구매해야 함)

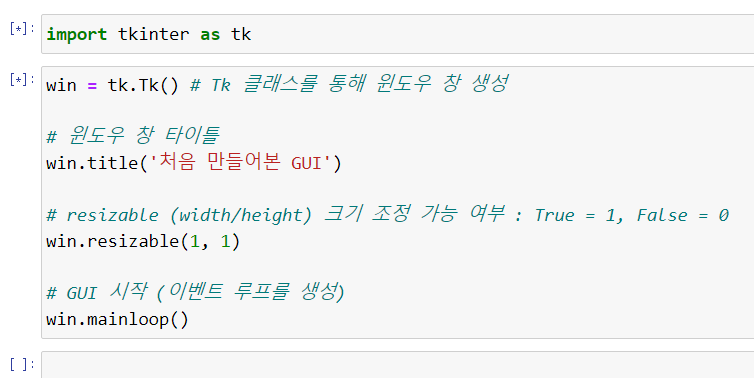
> Pyside

> PyGTK

# 자주 사용하는 라이브러리 : wxPython 라이브러리

'''

import tkinter as tk



import tkinter as tk

win = tk.Tk() # Tk 클래스를 통해 윈도우 창 생성

# 윈도우 창 타이틀

win.title('처음 만들어본 GUI')

# resizable (width/height) 크기 조정 가능 여부 : True = 1, False = 0

win.resizable(1, 1)

# GUI 시작 (이벤트 루프를 생성)

win.mainloop()



## 주요 위젯

### 라벨, 버튼 위젯

import tkinter as tk

# tkinter 서브 모듈 : ttk(themed tk) - 개선된 GUI를 제공하는 모듈

from tkinter import ttk

win = tk.Tk()

win.title('라벨ex')

# 라벨 위젯 : Label(윈도우명, text='내용')

label = ttk.Label(win, text='처음 만들어본 라벨')

# 라벨 부착

label.pack()

win.mainloop()

'''

# tkinter 패키지는 위젯 없을 때는 기본 크기를 사용

# 위젯을 추가할 경우 일반적으로 위젯을 표시하는데 필요한 만큼의 공간을 사용하도록 최적화

'''



win = tk.Tk()

win.title('window size setting')

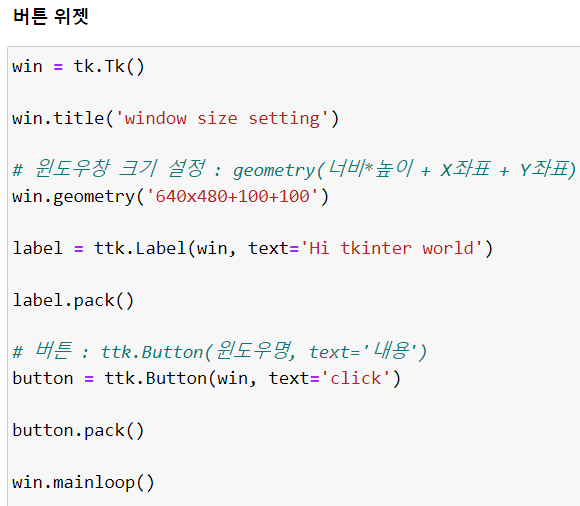
# 윈도우창 크기 설정 : geometry(너비\*높이 + X좌표 + Y좌표)

win.geometry('640x480+100+100')

label = ttk.Label(win, text='Hi tkinter world - adasdasdawfafsdawdawdasfawdsadadw')

label.pack()

win.mainloop()



win = tk.Tk()

win.title('window size setting')

# 윈도우창 크기 설정 : geometry(너비\*높이 + X좌표 + Y좌표)

win.geometry('640x480+100+100')

label = ttk.Label(win, text='Hi tkinter world')

label.pack()

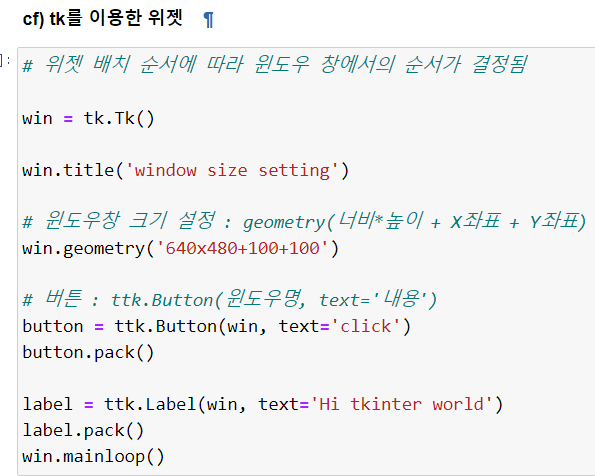
# 버튼 : ttk.Button(윈도우명, text='내용')

button = ttk.Button(win, text='click')

button.pack()

# pack이 놓는 거라서 순서 중요, 위젯 배치 순서에 따라 윈도우 창에서의 순서가 결정됨

win.mainloop()



# 위젯 배치 순서에 따라 윈도우 창에서의 순서가 결정됨

win = tk.Tk()

win.title('window size setting')

# 윈도우창 크기 설정 : geometry(너비\*높이 + X좌표 + Y좌표)

win.geometry('640x480+100+100')

# 버튼 : ttk.Button(윈도우명, text='내용')

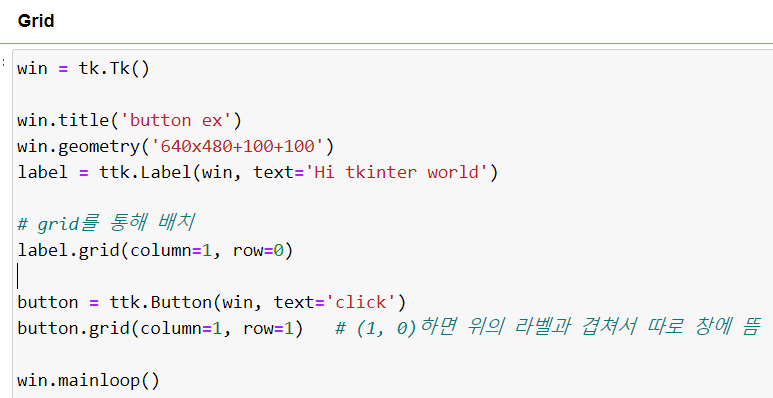
button = ttk.Button(win, text='click')

button.pack()

label = ttk.Label(win, text='Hi tkinter world')

label.pack()

win.mainloop()



win = tk.Tk()

win.title('button ex')

win.geometry('640x480+100+100')

label = ttk.Label(win, text='Hi tkinter world')

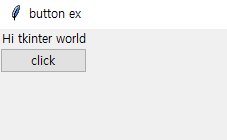
# grid를 통해 배치

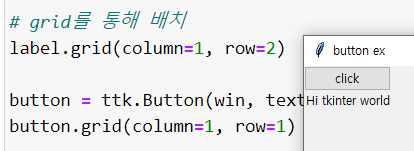
label.grid(column=0, row=0)

button = ttk.Button(win, text='click')

button.grid(column=0, row=1) # (1, 0)하면 위의 라벨과 겹쳐서 따로 창에 뜸

win.mainloop()





잘 조절하면 될 듯?

tkinter.ttk 위젯 https://docs.python.org/ko/3/library/tkinter.ttk.html

command: 콜백

제목 레이블을 누를 때 호출되는 콜백.



win = tk.Tk()

win.title('button ex')

win.geometry('640x480+100+100')

#clickEvent

def clickMe():

label.configure(text='button click') # 속성 변경:configure

button.configure(text='wow')

label.configure(background='yellow')

label = ttk.Label(win, text='Hi tkinter world')

label.grid(column=0, row=0)

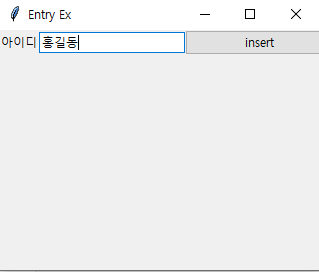
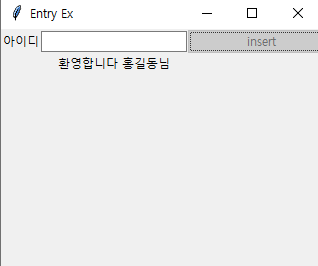
button = ttk.Button(win, text='click', command=clickMe)

button.grid(column=0, row=1)

win.mainloop()

* 버튼 누르면 라벨, 버튼 내용 변경됨 확인



win = tk.Tk()

win.title('Entry Ex')

win.geometry('320x240+200+200')

def inputId():

label2.config(text='환영합니다 '+id.get()+'님')

button1.config(state='disabled') #버튼 비활성화

id.set('') #id 인식시키고 null로

label1 = ttk.Label(win, text='아이디')

label1.grid(column=0, row=0)

'''

# entry 위젯 parameter

- show ex) (None | '\*')

- textvariable : 입력박스(텍스트 박스)에 작성한 문자열을 가져올 변수

- justify : 입력박스의 문자열이 여러 줄일 경우 정렬 방법 (left, right, center - default는 center)

# 파이선은 동적 타입 언어 (할당된 데이터를 바탕으로 타입을 유추)

예를 들어 name 변수에 문자열을 할당하면 string 타입이 됨 (name='hong')

age 변수에 정수를 할당하면 integer 타입이 됨 (age=20)

# ttk은 파이선이 아닌 다른언어로 되어있는 모듈임. 단지 파이선에서 사용할 수 있을 뿐.

그러므로 파이선처럼 동적으로 타입을 유추하는 모듈이 아님.

타입 지정이 필요한 언어임에 유의

'''

id = tk.StringVar() # 타입 지정

entry1 = ttk.Entry(win, textvariable=id)

entry1.grid(column=1, row=0)

button1 = ttk.Button(win, text='insert', command=inputId, width=20)

button1.grid(column=2, row=0)

label2 = ttk.Label(win, text='')

label2.grid(column=1, row=1)

win.mainloop()

4) CheckButton 위젯 : 다중 선택 가능

win = tk.Tk()

win.title('check button ex')

win.geometry('640x480+200+200')

def click():

chkBtn2.flash() # flash (normal 상태 배경 색상과 active 상태 배경 색상 사이에서 깜빡)

chkBtn1 = tk.Checkbutton(win, text='독서', state='disabled')

chkBtn1.grid(column=0, row=0, sticky=tk.W) # stricky : center가 없는 옵션

chkBtn2 = tk.Checkbutton(win, text='영화감상', activebackground='yellow')

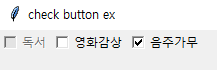
chkBtn2.grid(column=1, row=0, sticky='W')

chkBtn3 = tk.Checkbutton(win, text='음주가무', command=click)

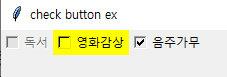
chkBtn3.select()

chkBtn3.grid(column=2, row=0)

win.mainloop()



3번 클릭하면 2번 플래시



5) ComboBox 위젯 : 선택지

win = tk.Tk()

win.title('combobox ex')

win.geometry('640x480+200+200')

def select():

lb.config(text=coffeeName.get()+'를/을 '+amount.get()+'잔 주문하셨습니다.')

ttk.Label(win, text='커피 종류 입력').grid(column=0, row=0)

coffeeName = tk.StringVar()

coffeeEntry = ttk.Entry(win, textvariable=coffeeName, width=10)

coffeeEntry.grid(column=1, row=0)

ttk.Label(win, text='잔 수를 입력하세요').grid(column=0, row=1)

amount = tk.StringVar() #연산할거 아니니까 일단 스트링

comboBox = ttk.Combobox(win, textvariable=amount, width=5)

comboBox.grid(column=1, row=1)

comboBox['values'] = (1, 2, 5, 10, 20) #튜플로

btn = ttk.Button(win, text='주문', command=select) #이벤트 처리 함수는 위에 둬야 함

btn.grid(column=0, row=2)

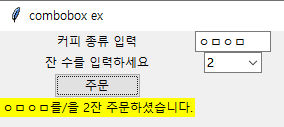
lb = ttk.Label(win, text='', background='yellow')

lb.grid(column=0, row=3)

# focus

coffeeEntry.focus()

win.mainloop()



6) RadioButton 위젯 : 변수 하나로 묶인 애들 중 택일

win = tk.Tk()

win.title('combobox ex')

win.geometry('640x480+200+200')

rBtn1 = tk.IntVar() # 라디오버튼 값 받을 변수라디오버튼 값 받을 변수

rBtn2 = tk.IntVar()

def check():

lb2.config(text='rBtn1 value : '+str(rBtn1.get())+'rBtn2 value : '+str(rBtn2.get()))

# value : 이 버튼을 누르면 값 넣게

# variable : 어디에 value를 넣을지

radioBtn1 = tk.Radiobutton(win, text='첫번째', value=1, variable=rBtn1, command=check)

radioBtn1.pack()

radioBtn2 = tk.Radiobutton(win, text='두번째', value=2, variable=rBtn1, command=check)

radioBtn2.pack()

radioBtn3 = tk.Radiobutton(win, text='세번째', value=3, variable=rBtn1, command=check)

radioBtn3.pack()

lb1 = tk.Label(win, text='---------------------')

lb1.pack()

lb2 = tk.Label(win, text='result')

lb2.pack()

lb3 = tk.Label(win, text='---------------------')

lb3.pack()

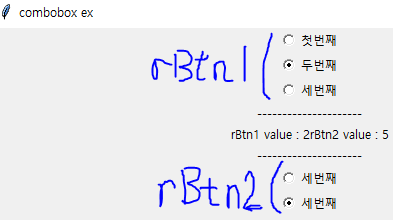
radioBtn4 = tk.Radiobutton(win, text='세번째', value=4, variable=rBtn2, command=check)

radioBtn4.pack()

radioBtn5 = tk.Radiobutton(win, text='세번째', value=5, variable=rBtn2, command=check)

radioBtn5.pack()

win.mainloop()



7) Text 위젯 : 여러 줄의 문자열을 출력하기 위한 위젯, 텍스트 에어리어 같음

win = tk.Tk()

win.title('Text ex')

win.geometry('640x480+200+200')

text = tk.Text(win)

text.insert(tk.CURRENT, 'Hello Text Widget - tkinter \n') #current는 커서 깜빡이는 곳 기준

text.insert('current', 'Hello Python World with Jupyter notebook \n')

text.insert(2.5, "!!!!") #2번째줄 5번째에 삽입

text.insert(tk.END, 'See you later~!') #마지막

text.pack()

text.tag\_add('emphasis', '1.6', '1.17') #태그 생성 및 범위 지정

text.tag\_config('emphasis', foreground='red')

text.tag\_add('emphasis2', '2.28', '2.44')

text.tag\_config('emphasis2', foreground='white', background='black')

win.mainloop()



8) ScrolledText 위젯 : Text 위젯을 구현하면서 스크롤 기능을 추가한 위젯

from tkinter import scrolledtext

win = tk.Tk()

win.title('Text ex')

win.geometry('640x480+200+200')

scrollWidth = 30

scrollHeight = 10

#텍스트 박스 넓이 높이 지정

text = scrolledtext.ScrolledText(win, width=scrollWidth, height=scrollHeight)

text.insert(tk.CURRENT, 'Hello Text Widget - tkinter \n') #current는 커서 깜빡이는 곳 기준

text.insert('current', 'Hello Python World with Jupyter notebook \n')

text.insert(2.5, "!!!!") #2번째줄 5번째에 삽입

text.insert(tk.END, 'See you later~! \n') #마지막

text.pack()

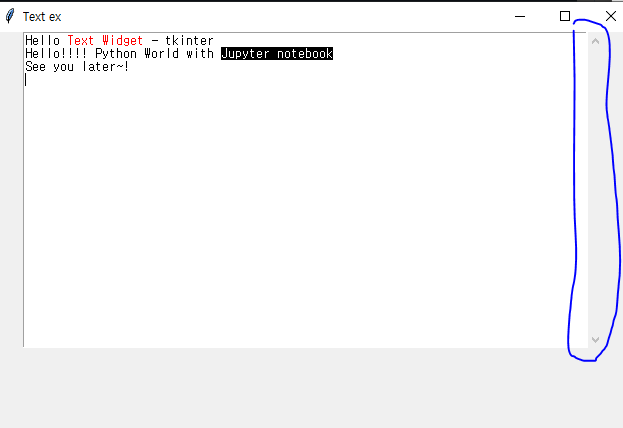
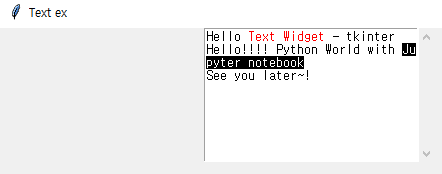
text.tag\_add('emphasis', '1.6', '1.17') #태그 생성 및 범위 지정

text.tag\_config('emphasis', foreground='red')

text.tag\_add('emphasis2', '2.28', '2.44')

text.tag\_config('emphasis2', foreground='white', background='black')

win.mainloop()

9) Frame 위젯 : 공간 분리할 때 사용

win = tk.Tk()

win.title('frame ex')

win.geometry('640x480+200+200')

frame1 = tk.Frame(win, bd=3, relief='solid', background='yellow') #bd:border

#어떻게 붙일지 설정 가능

frame1.pack(side='left', fill='both', expand=True) #fill을 none하면 frame을 안쓰겠다는 뜻

# x -> x축

# side 써서 오른쪽 왼쪽으로 할 수도

frame2 = tk.Frame(win, background='black')

frame2.pack(side='right', fill='both', expand=True)

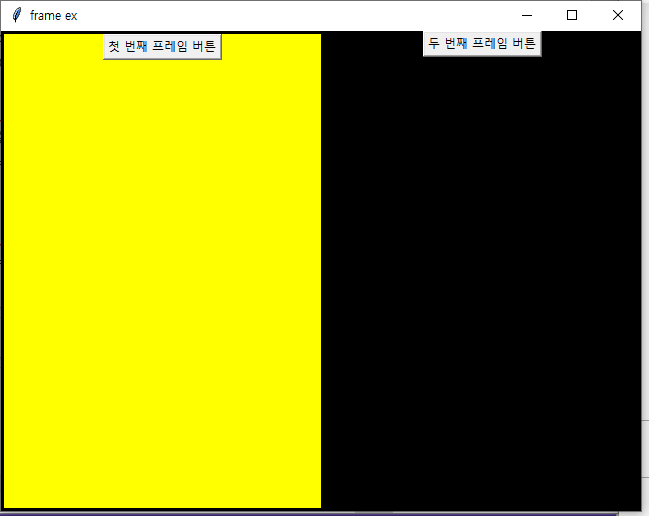
btn1 = tk.Button(frame1, text='첫 번째 프레임 버튼')

btn1.pack()

btn2 = tk.Button(frame2, text='두 번째 프레임 버튼')

btn2.pack()

win.mainloop()



10) ScrollBar 위젯 : 스크롤바와 연결된 위젯과는 서로 다른 객체이므로 프레임을 활용하여 사용

별도로 tk.Listbox 사용함

win = tk.Tk()

win.title('scrollbar ex')

win.geometry('640x480+200+200')

# 하나만 고를 수 있게, 클릭할때 밑줄 생기는거 안생기게

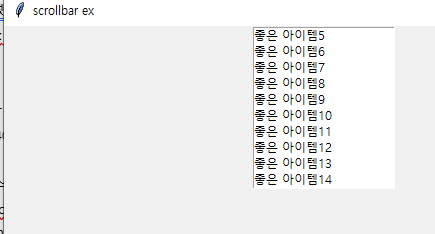
listBox = tk.Listbox(win, selectmode='single', activestyle='none')

for i in range(1, 20):

listBox.insert(i-1, '좋은 아이템{}'.format(i))

listBox.pack()

win.mainloop()



# ListBox에 ScrollBar를 부착, 스크롤 동작까지 하도록

win = tk.Tk()

win.title('scrollbar ex')

win.geometry('640x480+200+200')

frame = tk.Frame()

scrollBar = tk.Scrollbar(frame)

scrollBar.pack(fill='y', side='right')

listBox = tk.Listbox(frame, selectmode='single', activestyle='none'

, yscrollcommand=scrollBar.set #스크롤바 상호작용 하도록

, highlightcolor='yellow') #선택한 리스트박스 테두리색 변경

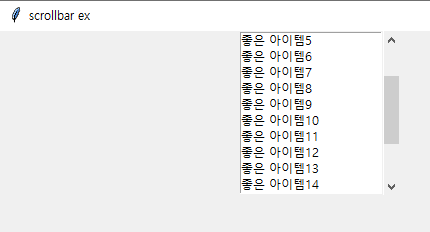
for i in range(1, 20):

listBox.insert(i-1, '좋은 아이템{}'.format(i))

listBox.pack(side='left')

frame.pack()

win.mainloop()



> thread (비동기 프로그래밍)

🡪 \_thread module

import \_thread

import time

import random as rd

def threadFunction(str): # thread로 동작하는 함수

cnt = 0

while(cnt < 10):

time.sleep(rd.randint(0, 100)/100.0)

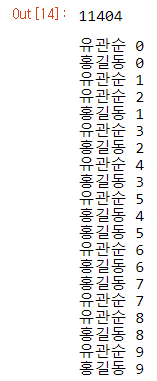
print(str, cnt)

cnt += 1

# \_thread.start\_new\_thread(쓰레드 진입점, (쓰레드 진입점에 전달할 인자)) #튜플로

\_thread.start\_new\_thread(threadFunction, ('홍길동',))

\_thread.start\_new\_thread(threadFunction, ('유관순',))



🡪 threading module

import threading

import time

import random

def threadFunction(str):

cnt = 0

while(cnt<10):

time.sleep(random.randint(0, 100)/100.0)

print(str, cnt)

cnt += 1

thread1 = threading.Thread(target=threadFunction, args=('신사임당',))

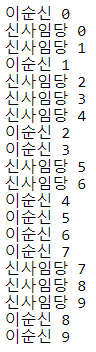
thread2 = threading.Thread(target=threadFunction, args=('이순신',))

thread1.start()

thread2.start()

thread1.join()

thread2.join()



스크래핑 작업???

import threading

import requests

import time

def getUrlText(url):

res = requests.get(url)

time.sleep(1)

print(url, len(res.text), ' 문자 정보')

td = threading.Thread(target=getUrlText, args=('http://example.com/',))

td.start()

for i in range(1, 11): #이 반복문 이 밑에 있음에도 위의 td가 반복문 뒤에 실행됨

print(i)

