Anaconda3-2020.11-Windows-x86\_64 다운로드

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

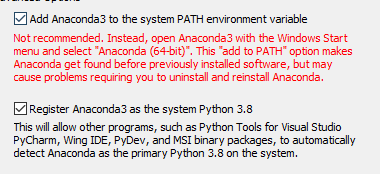
vmware workstation pro

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

안 다운 받는데?

숨김파일 표시, 아나콘다 인스톨러 관리자 권한으로 실행



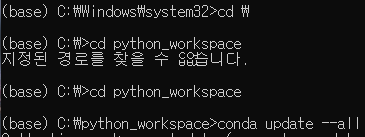


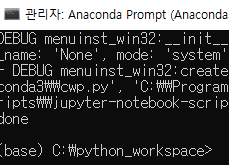
Anaconda Navigator (Anaconda3) 🡪 프로그램들 설치 지원 하는거

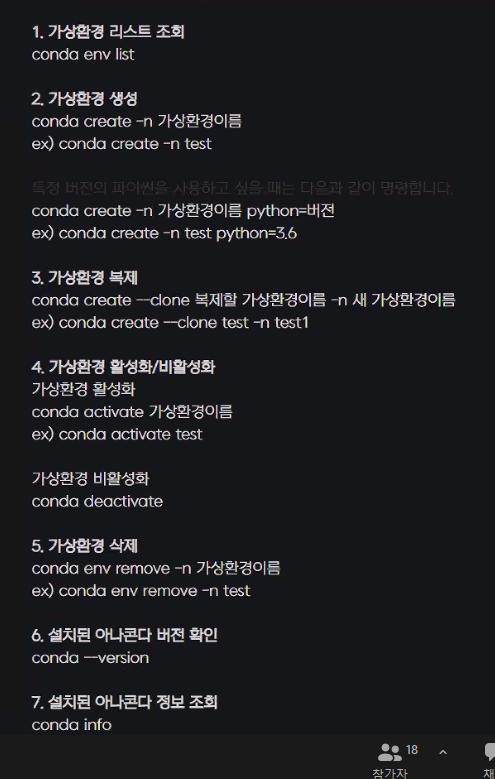
Anaconda Prompt (Anaconda3)

conda list -> 라이브러리 리스트

C 밑에 python\_workspace 생성



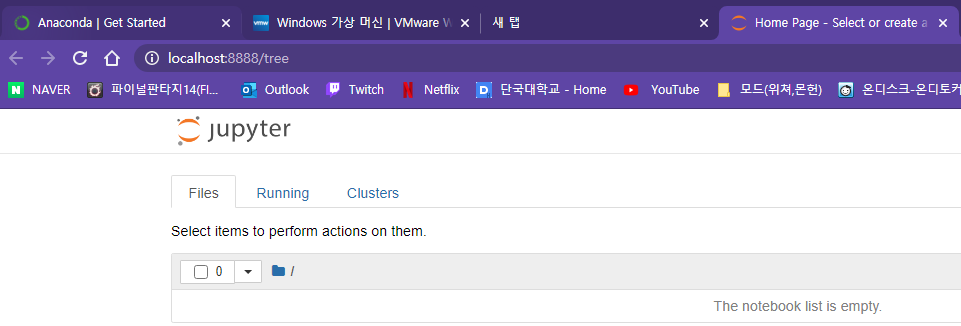
 업데이트 끝남



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

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

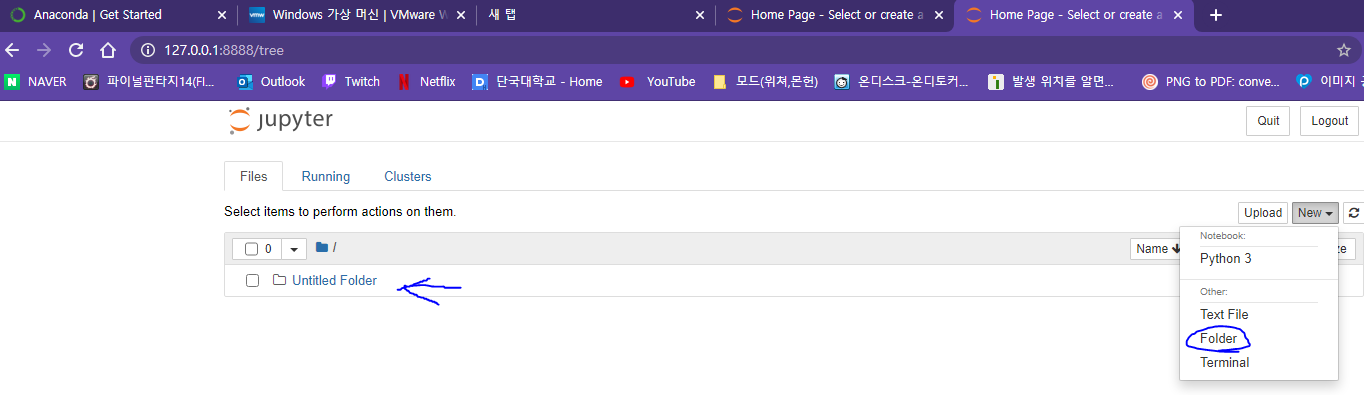
(goott7) C:\python\_workspace>jupyter notebook

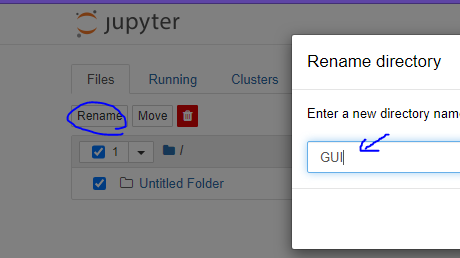


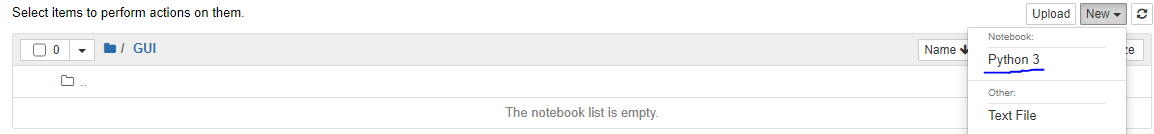
이런 창이 뜸



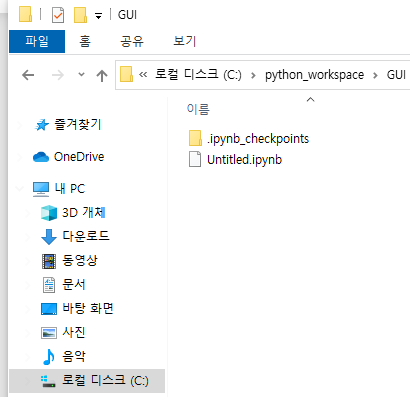
주피터



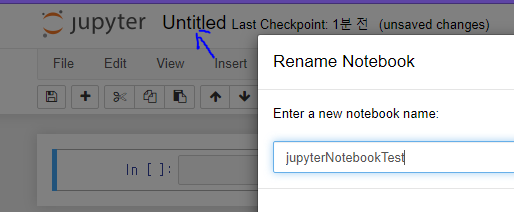




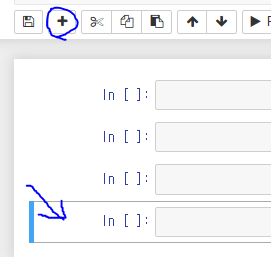
실제 경로에 파일이 생성됨



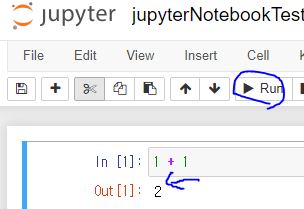
제목 부분을 누르면 변경 가능



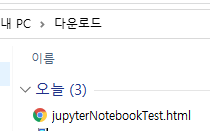
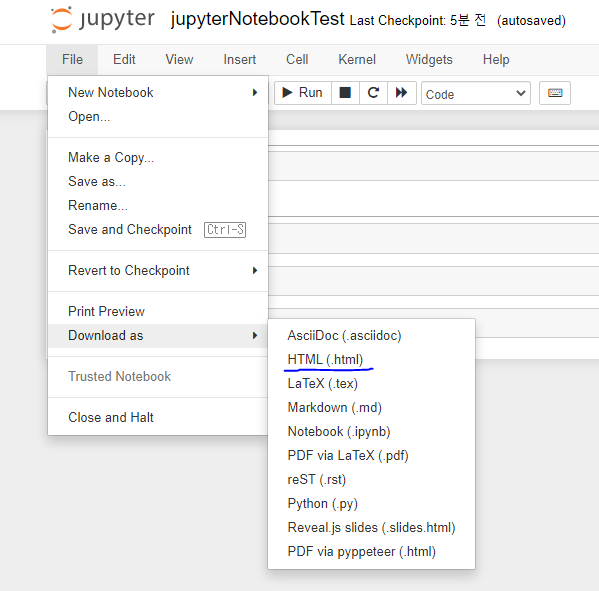
줄 추가 기능

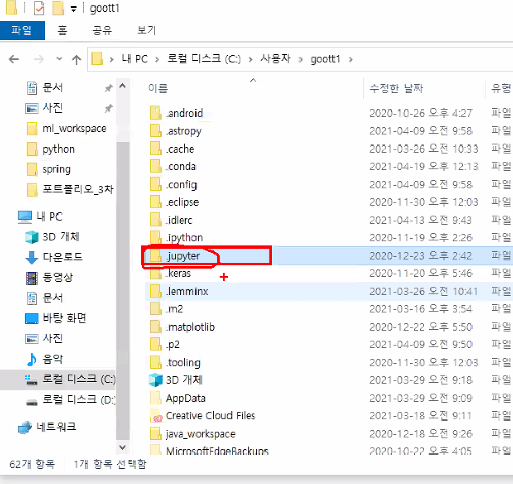


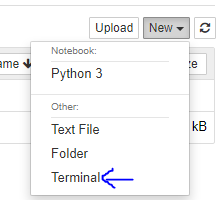
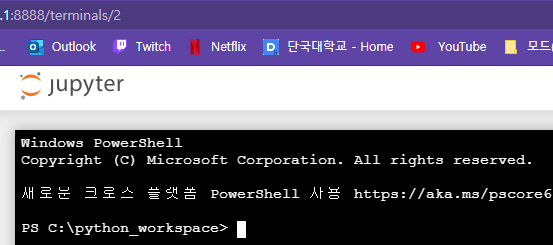
실행, 한 줄씩



HTML로 다운로드





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

(base) C:\>cd python\_workspace

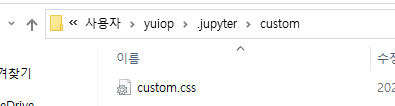
(base) C:\python\_workspace>jupyter notebook --generate-config

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

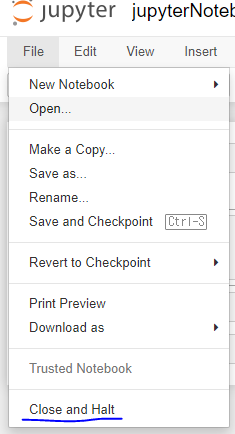
(base) C:\Users\yuiop>jupyter notebook --generate-config

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

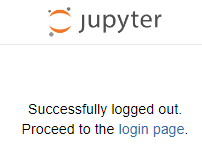
이 경로에 받은 custom.css 를 넣어두기

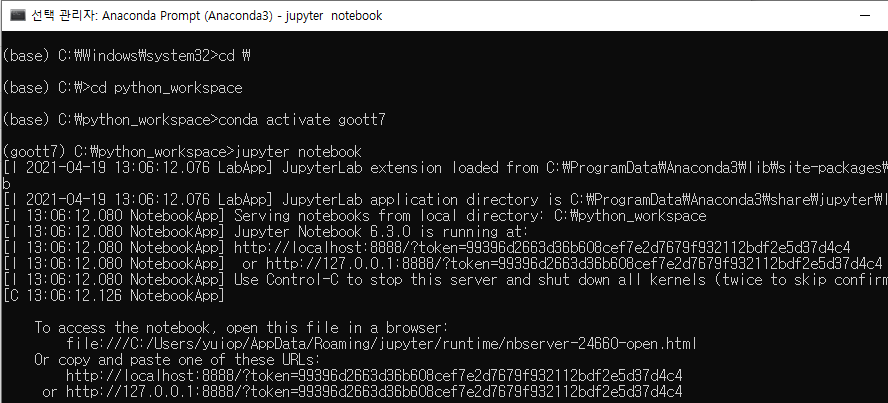


exit

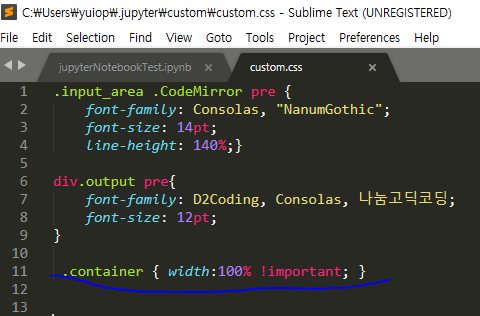


로그아웃

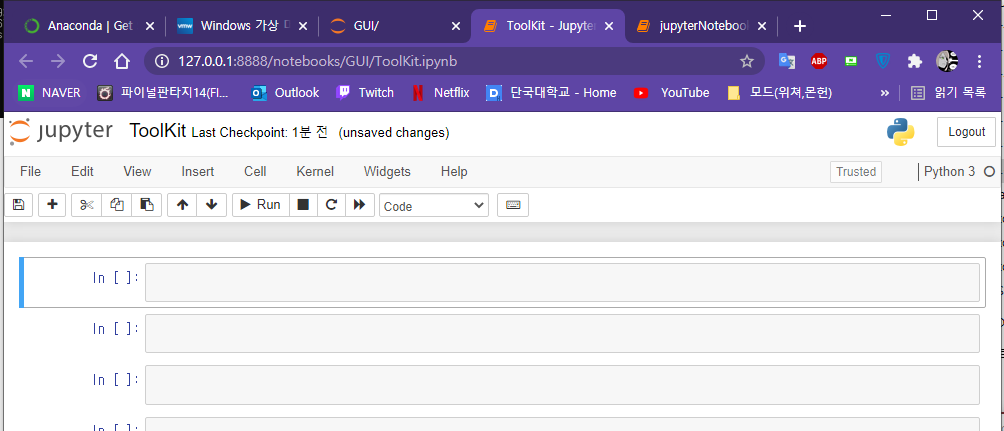




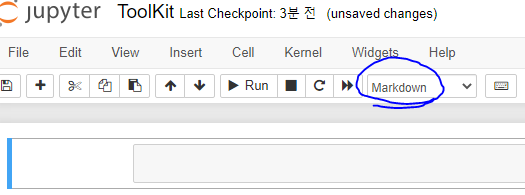
custom.css 파일 수정



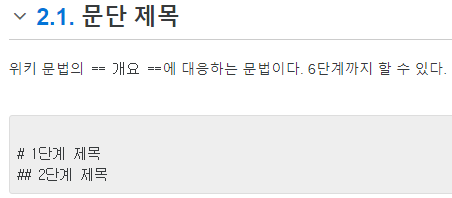
새 파일 ToolKit 생성, css 변경 확인 가능

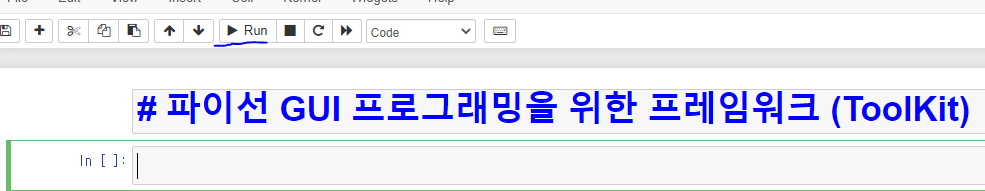


마크다운

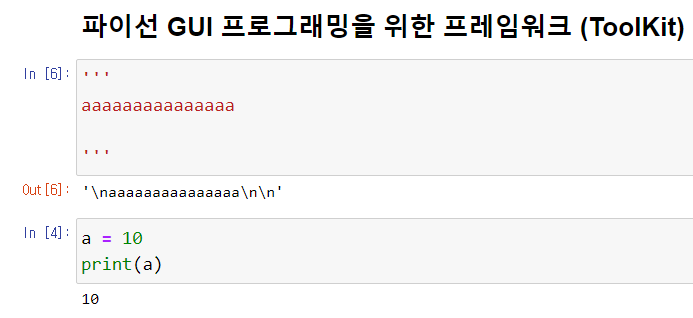


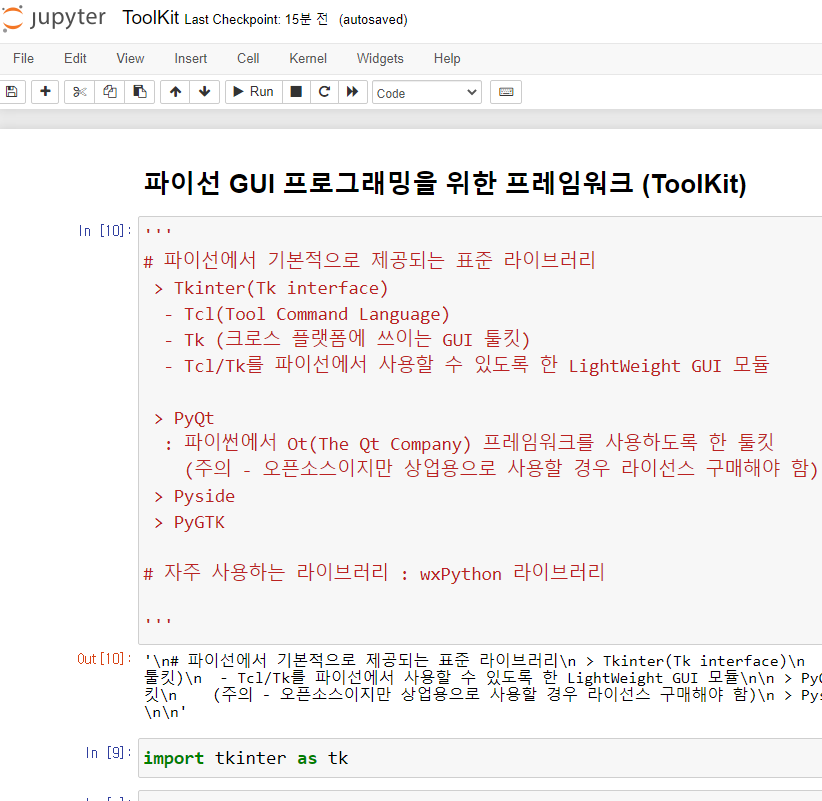
HTML 태그처럼 # 사용





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)

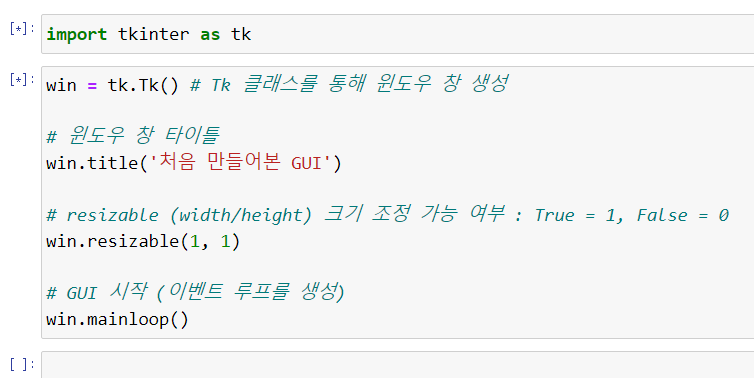
> Pyside

> 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')

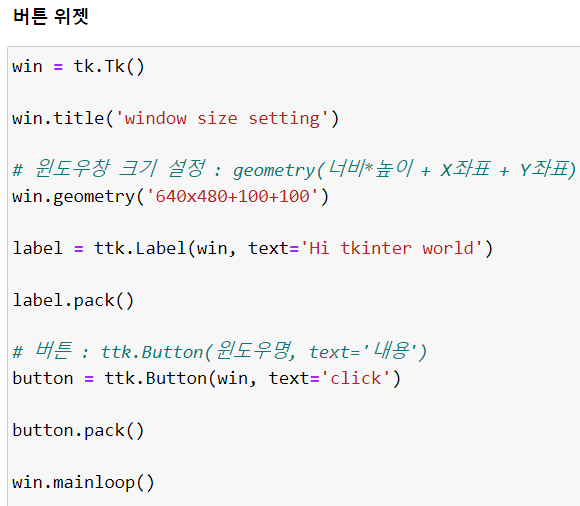
# 윈도우창 크기 설정 : geometry(너비\*높이 + X좌표 + Y좌표)

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()

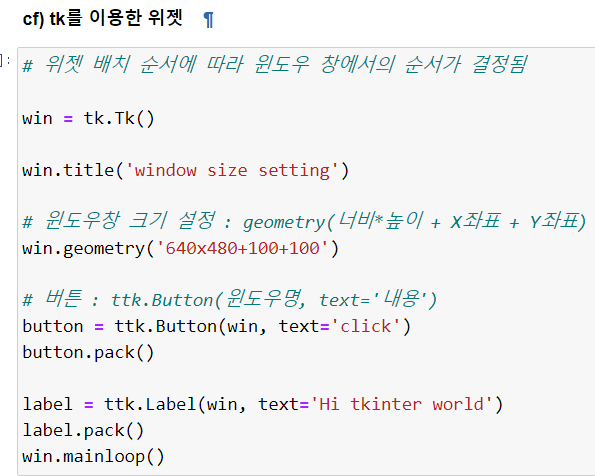
# 버튼 : ttk.Button(윈도우명, text='내용')

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')

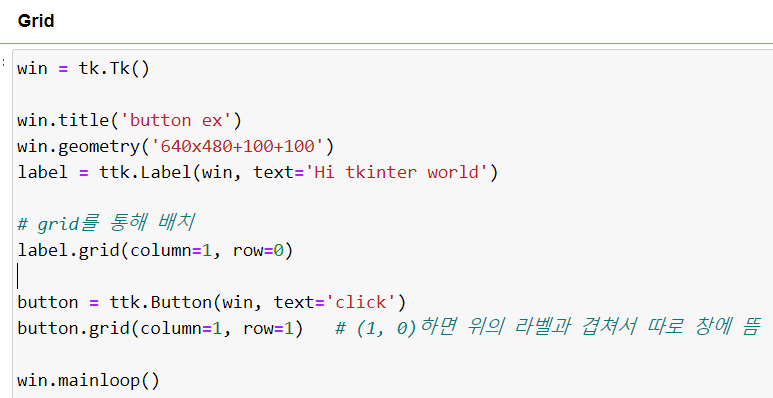
button = ttk.Button(win, text='click')

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')

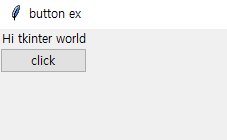
# grid를 통해 배치

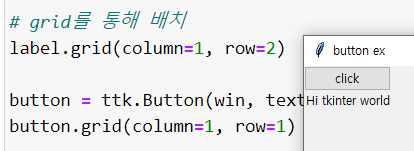
label.grid(column=0, row=0)

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)

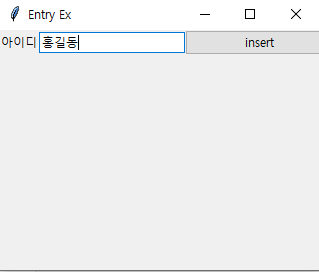
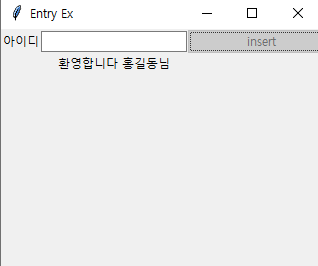
button = ttk.Button(win, text='click', command=clickMe)

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()