|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | matplotlib(4장) GUI\_OOP\_Classes\_imported\_tooltip.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  from tkinter import messagebox as msg  from time import sleep  import Tooltip as tt    def **create\_ToolTip**(widget, text):  toolTip=tt.ToolTip(widget)  def **enter**(event):  toolTip.show\_tip(text)  def **leave**(event):  toolTip.hide\_tip()  widget.bind(*"<Enter>"*,enter)  widget.bind(*"<Leave>"*,leave)  class **OOP**():  (~~)  def **create\_widgets**(*self*):  (~~)    # (tab1-라벨프레임 내) 스크롤드 테스트 위젯  scrol\_w = 30  scrol\_h =3  *self*.scr = scrolledtext.ScrolledText(mighty, width=scrol\_w, height=scrol\_h, wrap=tk.WORD)  *self*.scr.grid(column=0, columnspan=3, row=3, sticky=*'WE'*)  create\_ToolTip(*self*.scr,*'This is a ScrolledText widget'*) | | |  | | --- | | 4장 마지막 내용인 ‘데이터와 클래스’에 관한 내용입니다. |   Tooltip클래스 내용은 Tooltip.py파일에 따로 저장하였다.  그리고 imported\_tooltip파일에서는 Tooltip.py를 먼저 가져오고 tt라고 부르기도 했다.  즉 create\_Tooltip함수에서 기존에는 ToolTip(widget)을 써서 클래스를 인스턴스화했지만, 이제는 tt.을 앞에 붙여서  tt로 불리는 파이썬 파일 안에 있는 클래스를 가져온다고 작성해야 한다.  세 번째 노란 강조 줄에는 평소대로 소스를 작성해도 된다. create\_Tooltip함수는 같은 파이썬 파일에 있기 때문이다. |
| |  | | --- | | matplotlib(5장) Matplotlib\_first\_chart.py | | import numpy as np  import matplotlib.pyplot as plt  from pylab import show  x=np.arange(0,5,0.1)  y=np.sin(x)  plt.plot(x,y)  show() | | |  | | --- | | ( 5장 시작 )  Matplotlib: 수학 계산을 간편하게 하고 시각적 차트 등을 쉽게 그릴 수 있는 환경을 제공하는 파이썬 프로그램 |     numpy라는 모듈을 import하고 np라고 부르기로 하였다. numpy는 등간격의 정수 리스트를 생성하는 arrange함수와 sin값을 구할 수 있는 sin함수를 담고 있다.  Matplotlib.pyplot라는 모듈을 import하고 plt라고 부르기로 하였다. 위 모듈은 그래프 만들기 함수인 plot를 담고 있는 모듈이다.  pylab이라는 모듈을 import하고 show함수를 불러왔다. show함수는 사용자가 작성한 차트를 화면에 실제로 나타내는 역할을 한다.  위를 토대로 해석해보면,  0부터 5까지 0,1간격으로 작성한 배열을 x로 두고, 이 배열에 대한 sin값을 y로 두었다.  이후 x,y배열을 토대로 그래프를 만들고 화면에 내보냈다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | matplotlibMatplotlib\_second\_chart.py | | from pylab import show,arange,sin,plot,pi  a=arange(0.0,2.0,0.01)  b=sin(2\*pi\*a)  plot(a,b)  show() | | |  | | --- | |  |     위에서 numpy,matplotlib.pyplot모듈이 가지고 있는 함수는 pylab모듈에도 다 있는 함수이다.  그래서 이번에는 pylab모듈을 import하고 show,arrange,sin,plot함수를 불러왔다.  추가적으로 원주율에 해당하는 pi상수도 불러왔다.  0.0부터 2.0까지 0.01간격으로 작성한 배열을 a로 두고 이 배열에 2pi를 곱하고 sin값을 구한 것을 b라고 하였다.  이후 plot함수를 통해 a,b배열을 가지고 그래프 를 만들고 이걸 화면에 내보냈다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | matplotlibMatplotlib\_labels.py | | import tkinter as tk  from matplotlib.figure import Figure  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  fig=Figure(figsize=(6,4),facecolor=*'white'*)  axis=fig.add\_subplot(211)  xvalues=[1,2,3,4]  yvalues=[5,7,6,8]  axis.plot(xvalues,yvalues)  axis.set\_xlabel(*"Horizontal Label"*)  axis.set\_ylabel(*"Vertical Label"*)  axis.grid(linestyle=*'-'*)  def **\_destroyWindow**():  root.quit()  root.destroy()  root=tk.Tk()  root.withdraw()  root.protocol(*"WM\_DELETE\_WINDOW"*,\_destroyWindow) # x창 치면 콜백함수를 실행하라  canvas=FigureCanvasTkAgg(fig,master=root)  canvas.\_tkcanvas.pack(side=tk.TOP,fill=tk.BOTH,expand=1)  root.update()  root.deiconify()  root.mainloop() | | |  | | --- | | **(추가 설명)**  * figure객체   차트 틀을 담은 객체   * 차트 틀   차트를 여러 개 담을 수 있는 공간 |     대표적으로 GUI를 제공하는 모듈인 tkinter을 불러오고 matplotlib.figure라는 모듈애서 Figure클래스를 불러왔다. 이 클래스는 차트 틀을 제공하고, 차트의 설정과 다양화를 가능하게 하는 소스 블록이다.  이 클래스를 이용하면 범례, 축 제목, 차트의 위치, 차트 디자인 등을 만들 수 있다.  이후 matplotlib.backends.backend\_tkagg모듈에서 FigureCanvasTkAgg클래스를 불러왔다. 이 클래스는  차트를 gui폼 창에 삽입하는 역할을 한다.  Figure클래스를 인스턴스화하면서 차트 틀 사이즈를 6px,4px로 하고 (차트 틀 배경)을 흰색으로 설정만 하였다. 이때 figure(차트 틀)은 fig객체에 저장한다. 이후 add\_subplot메소드를 이용해서 행 방향으로 2 개,열 방향으로 1개만큼 차트를 만들고 이 중 1번째 차트를 axis객체에 저장하였다.  이후 xvalues,yvalues리스트를 만든 후 plot함수를 이용해서 차트(axis객체)에 x값, y값에 따른 그래프를 추가하였다. 밑에 set\_xlabel은 x축 제목을 만드는 메소드이다. grid메소드에서 차트에 실선 스타일을 주었다.  노란색 소스로 넘어가면 FigureCanvasTkAgg클래스를 인스턴스화하면서 만들어진 root폼 창에 (차트를 담은)차트틀을 삽입하는 내용이다.  Root폼 창을 업데이트해야 성공적으로 최종적으로 삽입이 완료된다. (update메소드 이용) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | matplotlibMatplotlib\_labels\_four.py | | (~~)  fig=Figure(figsize=(12,8),facecolor=*'white'*)  xvalues=[1,2,3,4]  yvalues=[5,7,6,8]  axis1=fig.add\_subplot(221)  axis2=fig.add\_subplot(222,sharex=axis1,sharey=axis1)  axis3=fig.add\_subplot(223,sharex=axis1,sharey=axis1)  axis4=fig.add\_subplot(224,sharex=axis1,sharey=axis1)  axis1.plot(xvalues,yvalues)  axis1.set\_xlabel(*"Horizontal Label"*)  axis1.set\_ylabel(*"Vertical Label"*)  axis1.grid(linestyle=*'-'*)  axis2.plot(xvalues,yvalues)  axis2.set\_xlabel(*"Horizontal Label"*)  axis2.set\_ylabel(*"Vertical Label"*)  axis2.grid(linestyle=*'-'*)  axis3.plot(xvalues,yvalues)  axis3.set\_xlabel(*"Horizontal Label"*)  axis3.set\_ylabel(*"Vertical Label"*)  axis3.grid(linestyle=*'-'*)  axis4.plot(xvalues,yvalues)  axis4.set\_xlabel(*"Horizontal Label"*)  axis4.set\_ylabel(*"Vertical Label"*)  axis4.grid(linestyle=*'-'*)  (~~) | | |  | | --- | | **(이어서)** mainloop을 통해 실체화하기 전 deiconify메소드를 통해 정상적인 형태의 윈도우로 만들 필요성이 있다. 차트가 삽입되면 비정상으로 여겨지기 때문이다. |   차트 설정과 리스트는 동일하다.  add\_subplot메소드를 이용해서 행 방향으로 2 개,열 방향으로 2개만큼 차트를 만들고 이 중 1번째 차트를 axis1객체에 저장하였다.  같은 방식으로 2번째~4번째 차트를 axis2~axis4에 저장하였다.  차트를 4개 만들었으니 x값,y값도 각 객체마다 줘야 되고 축 제목과 grid설정도 각각해야 한다.  왼쪽 소스를 보면 다 같게 차트를 설정하였으니 행과 열만 다른 채 다 같은 것을 알 수 있다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | matplotlibex\_add\_subplot.py | | import tkinter as tk  from matplotlib.figure import Figure  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  import matplotlib.pyplot as plt  fig=plt.figure(figsize=(12,9))  ax1=fig.add\_subplot(221,projection=*'polar'*)  ax1.set\_xlabel(*'ax1 Label'*)  ax2=fig.add\_subplot(222,facecolor=*'red'*)  ax2.set\_xlabel(*'ax2 Label'*)  ax3=fig.add\_subplot(223)  ax3.set\_xlabel(*'ax3 Label'*)  ax4=fig.add\_subplot(224,frameon=False)  ax4.set\_xlabel(*'ax4 Label'*)  (~~) | | |  | | --- | |  |     matplotlib.figure모듈에 있는 Figure클래스는  matplotlib.pyplot모듈에 있는 figure클래스와 같은 역할을 한다.  그래서 사용자는 편한 모듈을 이용하면 된다.  하늘색 강조 부분을 보면 figure클래스를 이용한 모습이다.  다음은 초록색 강조 부분을 보자.  subplot메소드를 통해 차트를 만들 때 projection속성을 줄 수 있다.  이 속성은 차트의 모양을 결정한다. polar을 옵션으로 주면 원형 스타일로 모양이 변한다.  facecolor속성은 차트의 배경색을 결정한다.  frameon속성은 차트 테두리의 유무를 결정한다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | matplotlibEx\_add\_plot.py | | import tkinter as tk  from matplotlib.figure import Figure  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  fig=Figure(figsize=(12,8),facecolor=*'white'*)  axis=fig.add\_subplot(211)  xvalues=[1,2,3,4]  yvalues=[5,6,7,8]  axis.plot(xvalues,yvalues)  axis.grid(linestyle=*'--'*,c=*'r'*)  axis=fig.add\_subplot(212)  xvalues2=[1,2,3,4]  yvalues2=[5,6,7,8]  axis.plot(xvalues2,yvalues2)  axis.grid()  (~~) | | |  | | --- | |  |     하늘색 부분은 grid메소드를 통해 차트에 속성을 주는 소스가 있다.  Linestyle속성은 (그래프 수치를 좀 더 수월하게 보게 하는 배경 선)을 결정한다.  옵션을 ‘-‘로 주면 실선으로 결정하고  ‘—‘로 주면 점선으로 결정한다.  c속성은 차트 배경선의 색깔을 결정한다.  r은 red라는 의미이다.  정리해보면 차트 틀의 배경 색은 figure가 인스턴스화되면서 설정하고  차트의 베경 색은 subplot메소드를 통해 차트가 만들어지면서 설정하고  차트 배경선의 색은 grid메소드를 통해 평소 속성을 주듯이 설정하면 된다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | matplotlibMatplotlib\_chart\_with\_legend.py | | import tkinter as tk  from matplotlib.figure import Figure  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  fig=Figure(figsize=(12,5),facecolor=*'white'*)  axis=fig.add\_subplot(111)  xvalues=[1,2,3,4]  yvalues0=[6,7.5,8,7.5]  yvalues1=[5.5,6.5,8,6]  yvalues2=[6.5,7,8,7]  t0, =axis.plot(xvalues, yvalues0, color=*"r"*)  t1, =axis.plot(xvalues, yvalues1, color=*"b"*)  t2, =axis.plot(xvalues, yvalues2, color=*"purple"*)  axis.set\_xlabel(*"Horizontal Label"*)  axis.set\_ylabel(*"Vertical Label"*)  axis.grid()  fig.legend((t0,t1,t2),(*"First line"*,*"second Line"*,*"Third Line"*),*"upper right"*)  (~~) | | |  | | --- | |  |     왼쪽 소스도 리스트 값을 토대로 세 개의 차트가 만든 것이다.  하지만 아래 하늘 색 강조 부분을 보면 legend라는 메소드가 쓰인 것을 알 수 있다.  legend메소드는 차트 틀(fig)에 범례를 추가하는 기능이 있다.  범례는 그래프를 설명해주는 역할을 한다.  첫 번째 인수에는 설명하려는 그래프 목록(그래프 객체 이름들)을 작성하면 되고 두 번째 인수는 각각의 그래프에 대한 설명이 나와 있다.  예를 들어 t0그래프는 Firstline이라고 설명을 쓸 수 있다.  다만 범례를 주기 전 주의할 점이 있다.  그래프 객체 이름에 ‘,’가 필수적으로 붙여야 된다는 점이다. |

**[5] MATPLOTLIB**

**새로운 위젯 및 속성 정리**

Matplotlib: 수학 계산을 간편하게 하고 시각적 차트 등을 쉽게 그릴 수 있는 환경을 제공하는 파이썬 프로그램

**Figure클래스:** 생성자로 차트 틀을 제공하고, 차트의 설정과 다양화를 가능하게 하는 소스 블록

**Figure클래스 생성자의 (차트 틀 위젯):** 여러 개의 차트를 담을 수 있는 공간

- figsize: 차트 틀의 가로,세로 크기를 결정하는 속성이다.

- facecolor: 차트 틀 배경 색을 결정하는 속성이다.

- frameon: 차트 틀 테두리를 결정하는 속성이다. (true일 때 생성)

**subplot메소드의 (차트 위젯):** 여러 개의 그래프를 담을 수 있는 공간

- projection: 차트의 모양을 결정하는 속성이다. (polar옵션일 때 원형 스타일)

- facecolor: 차트 배경 색을 결정하는 속성이다.

- frameon: 차트 테두리를 결정하는 속성이다. (true일 때 생성)

**= grid메소드 =**

- linestyle: 차트 배경선(그래프 수치를 좀 더 수월하게 보게 하는 배경 선)의 스타일을 결정한다.

(옵션이 ‘-‘일 때는 점선으로, ‘—‘일 때는 실선으로 표시한다.)

- c: 차트 배경선의 색상을 결정하는 속성이다.

(옵션이 ‘r’일 때는 red, ‘g’일 때는 green, ‘b’일 때는 blue이다. 나머지 색은 풀네임을 입력)

**plot메소드의 (그래프 위젯):** x,y값들을(수치를) 그래픽으로 나타낸 것

- color: 그래프의 색상을 결정한다.

**legend메소드의 (차트 범례 위젯):** 그래프를 설명해주는 틀 위에 있는 위젯