19장. wxPython

파이썬 정복

Contents

❖ 목차

1. 윈도우

❖ 소개

- Tkinter는 간단한 유틸리티용 이상의 용도에서 한계 있음

- wxPython
  - wxWidget
    - 1992년 발표된 크로스 플랫폼 GUI 툴킷
    - 윈도우, 맥, 리눅스 모두 지원
    - C++로 작성
    - 오픈소스
    - http://www.wxPython.org 혹은 명령행에서 pip로 설치

1. 윈도우

❖ App

- 응용 프로그램
  - 윈도우 소유하며 운영체제나 사용자로부터 전달되는 이벤트 처리

- 메인 윈도우 생성
  - Mainloop 메서드
    - 각종 메시지를 디폴트 처리

wxApp

```
import wx

app = wx.App()
frame = wx.Frame(None)

frame.Show(True)
app.MainLoop()
```

1. 윈도우

- MyAPP 클래스에서 OnInit 재정의하여 프레임 생성
- 차일드 위젯 배치 및 이벤트 핸들러 연결

wxApp2

```
import wx

class MyApp(wx.App):
    def OnInit(self):
        frame = wx.Frame(None)
        frame.Show(True)
        return True

app = MyApp()
app.MainLoop()
```

1. 윈도우

❖ 윈도우의 속성

- GUI 화면에 나타나는 객체는 클래스 계층을 이룸
- 이 계층 파악하고 클래스 분석할 수 있어야

- **Frame 생성자**

```
Frame(parent, id=ID_ANY, title="", pos=DefaultPosition,
      size=DefaultSize, style=DEFAULT_FRAME_STYLE, name=FrameNameStr)
```

1. 윈도우

- 다양한 속성 조절할 수 있음
  - 크기
    - » 윈도우가 차지할 폭과 높이
  - 위치
    - » 윈도우가 배치될 좌표 지정
  - 배경색
    - » SetBackgroundColour 함수
    - » r, g, b, a 색상 요소값 가지는 Colour 객체 전달
  - 타이틀 텍스트
    - » 윈도우 타이틀 바의 문자열 지정
  - 스타일
    - » 윈도우 모양과 동작 지정하는 각종 플래그 집합
    - » 조합에 따라 다양한 윈도우 생성

1. 윈도우

플래그	설명
ICONIZE, MINIMIZE	최소화 상태로 시작한다.
CAPTION	타이틀 바를 표시하고 캡션 문자열을 표시한다. 이 속성이 있어야 최소, 최대화, 닫기 버튼을 배치할 수 있다.
MINIMIZE_BOX	최소화 버튼을 배치한다.
MAXIMIZE	최대화 상태로 시작한다.
MAXIMIZE_BOX	최대화 버튼을 배치한다.
CLOSE_BOX	닫기 버튼을 배치한다.
STAY_ON_TOP	다른 윈도우보다 항상 위에 있는다.
SYSTEM_MENU	시스템 메뉴를 표시한다.
RESIZE_BORDER	크기 조정이 가능한 경계선을 사용한다.
FRAME_TOOL_WINDOW	타이틀 바가 작은 형태로 생성하며 작업 표시줄에 나타나지 않는다.
FRAME_NO_TASKBAR	작업 표시줄에 나타나지 않는다.
FRAME_FLOAT_ON_PARENT	부모보다 항상 위에 있는다.
FRAME_SHAPED	SetShape 메서드로 모양을 변경할 수 있다.
CLIP_CHILDREN	다시 그릴 때 차일드 영역을 제외하여 깜박임을 제거한다.
DEFAULT_FRAME_STYLE	여러 가지 속성의 조합으로 가장 일반적인 프레임 윈도우를 정의한다. MINIMIZE_BOX | MAXIMIZE_BOX | RESIZE_BORDER | SYSTEM_MENU | CAPTION | CLOSE_BOX | CLIP_CHILDREN

1. 윈도우

❖ 이벤트 핸들러

- 특정 이벤트 처리하는 함수
- 이벤트에 대한 핸들러 지정
  - 이벤트 발생할 때마다 핸들러 함수 호출

```
Bind(이벤트명, 핸들러)
```

이벤트	설명
EVT_KEY_DOWN	키를 눌렀다.
EVT_KEY_UP	키를 뗐다.
EVT_CHAR	문자를 입력했다.
EVT_LEFT_DOWN	마우스 왼쪽 버튼을 눌렀다.
EVT_LEFT_UP	마우스 왼쪽 버튼을 뗐다.
EVT_LEFT_DCLICK	마우스 왼쪽 버튼을 더블클릭했다.
EVT_SIZE	윈도우의 크기가 변경되었다.
EVT_PAINT	윈도우를 다시 그린다.

2. 위젯

❖ 위젯 배치

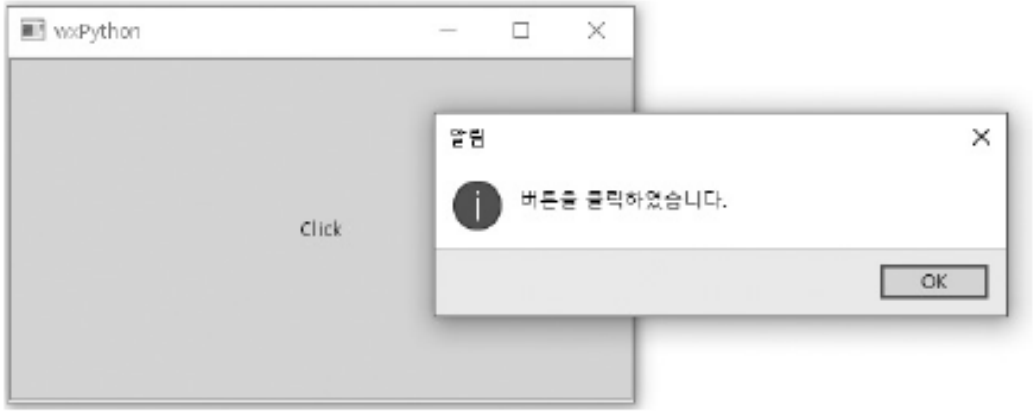
- 사용자 대면 및 실질적 작업 처리 위해 프레임에 다양한 위젯 배치

Button

```
import wx

app = wx.App()
frame = wx.Frame(None, title = "wxPython")

btn = wx.Button(frame, label="Click")
def OnClick(event):
    wx.MessageBox("버튼을 클릭하였습니다.", "알림", wx.OK)
btn.Bind(wx.EVT_BUTTON, OnClick)

frame.Show(True)
app.MainLoop()
```

- 위젯의 첫 번째 인수는 항상 부모 윈도우
  - 자신이 속할 윈도우 객체 지정
- 나머지 속성은 필요한 것만 키워드 인수로 지정

2. 위젯



- **버튼 생성문 다음에 아래 코드 추가하여 문자열 표시 스태틱 배치**

```
lbl = wx.StaticText(frame, label="Text")
```

2. 위젯

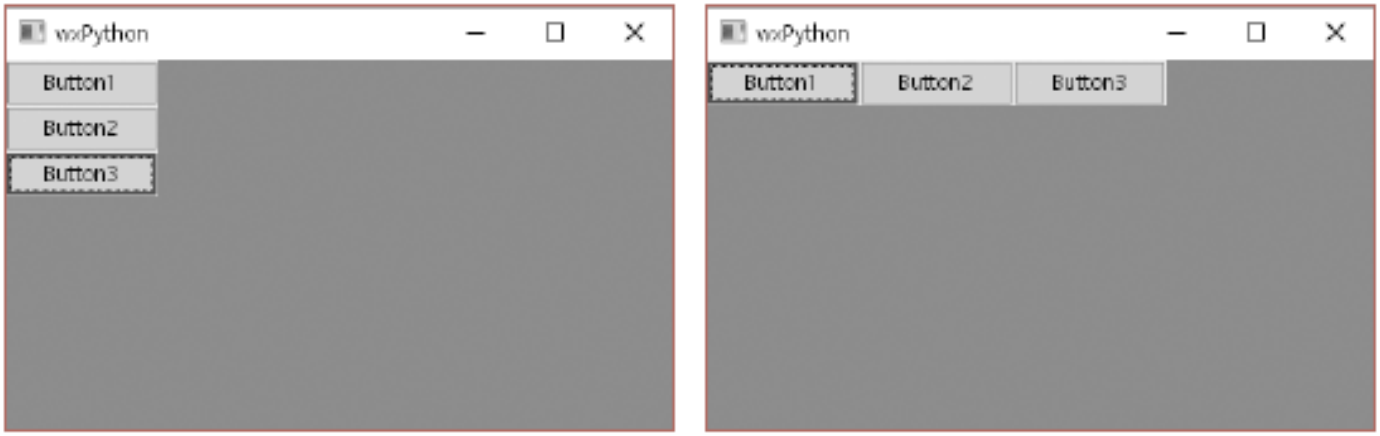
❖ 사이저

- 미리 정한 규칙과 속성에 따라 차일드의 위치와 크기 자동조정
- Boxsizer
  - 위젯을 수직 혹은 수평으로 일렬 배치

BoxSizer

```
import wx

app = wx.App()
frame = wx.Frame(None, title = "wxPython")

btn1 = wx.Button(frame, label="Button1")
btn2 = wx.Button(frame, label="Button2")
btn3 = wx.Button(frame, label="Button3")
box = wx.BoxSizer(wx.VERTICAL)
frame.SetSizer(box)
box.Add(btn1)
box.Add(btn2)
box.Add(btn3)

frame.Show(True)
app.MainLoop()
```

2. 위젯



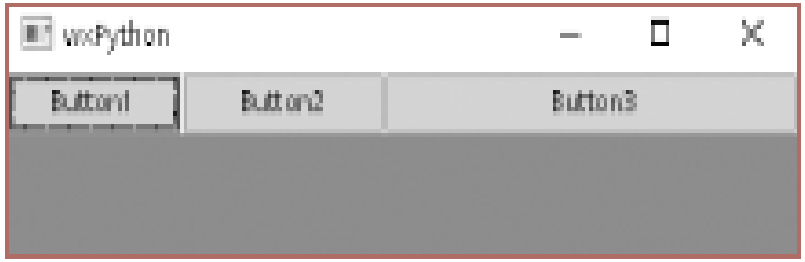
- **배치 옵션 통해 여백 및 배치 방식 추가 조정**

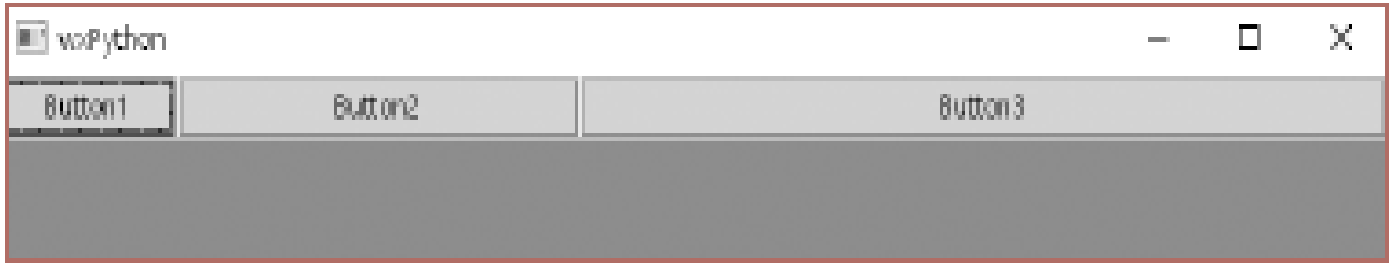
```
Add (window, proportion=0, flag=0, border=0, userData=None)
```

BoxSizer2

```
box.Add(btn1, proportion = 0)
box.Add(btn2, proportion = 1)
box.Add(btn3, proportion = 2)
```

2. 위젯


wxPython
Button1
Button2
Button3


wxPython
Button1
Button2
Button3

2. 위젯

flag 인수

- 사이저의 여러 옵션 지정하는 플래그 조합

플래그	설명
LEFT, TOP, RIGHT, BOTTOM, ALL	border의 폭을 적용할 면을 지정한다.
EXPAND	위젯이 자신의 공간을 모두 채운다.
SHAPED	EXPAND와 같되 종횡비는 유지한다.
FIXED_MINSIZE	최소한의 크기를 유지한다.
RESERVE_SPACE_EVEN_IF_HIDDEN	숨겨진 상태에서도 자리를 차지한다.
ALIGN_LEFT, ALIGN_RIGHT ALIGN_CENTER ALIGN_TOP ALIGN_BOTTOM ALIGN_CENTER_VERTICAL ALIGN_CENTER_HORIZONTAL	위젯의 정렬 방식을 지정한다. CENTER는 영국식 영어인 CENTRE로 써도 상관없다.

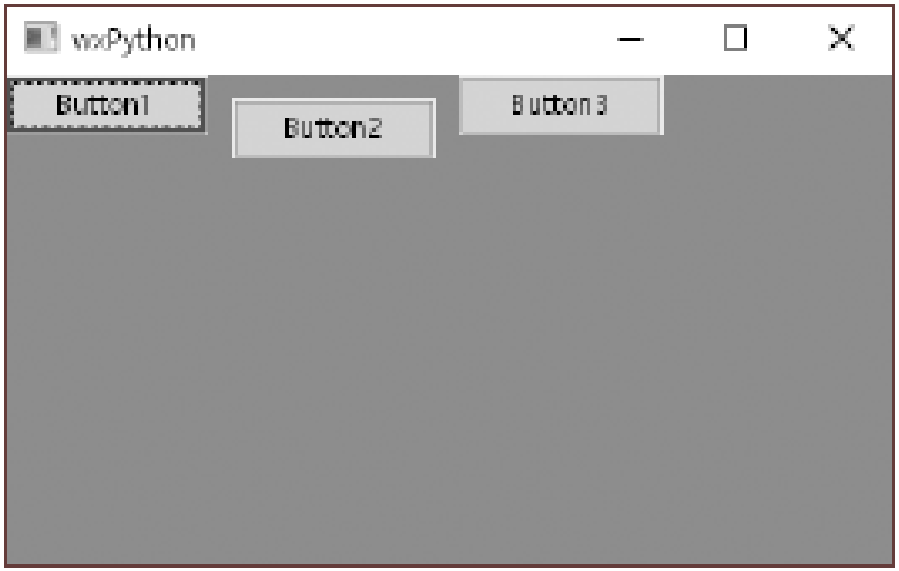
2. 위젯

border 인수

- 위젯의 외부 여백 폭 지정

BoxSizer3

```
box.Add(btn1, border = 10)
box.Add(btn2, border = 10, flag = wx.ALL)
box.Add(btn3, border = 10)
```

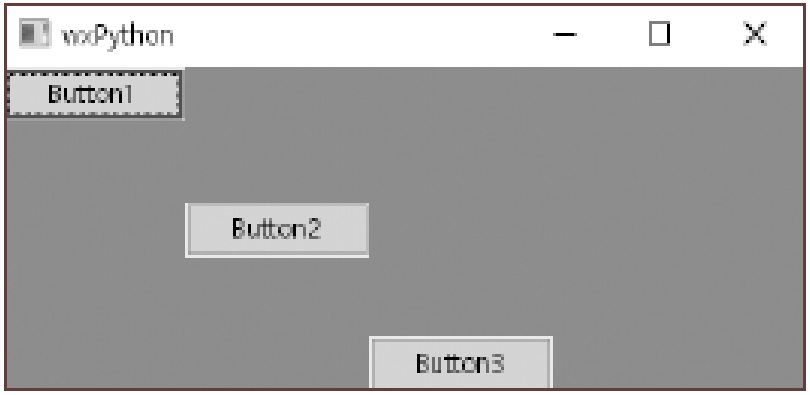


2. 위젯

- 정렬은 박스의 방향과 반대로만 지정 가능

BoxSizer4

```
box.Add(btn1)
box.Add(btn2, flag = wx.ALIGN_CENTER)
box.Add(btn3, flag = wx.ALIGN_BOTTOM)
```



2. 위젯

❖ 패널

- 프레임과 같은 기능이되 직접 눈에 보이지 않음
  - 한 컨테이너 내에서 수평, 수직 두 가지 정렬을 동시에 적용할 수 없음
  - 프레임에 직접 배치하기보다 중간에 패널 둠

Panel

```
import wx

app = wx.App()
frame = wx.Frame(None, title = "wxPython")

panelhorz = wx.Panel(frame)
btn1 = wx.Button(panelhorz, label="Button1")
btn2 = wx.Button(panelhorz, label="Button2")
btn3 = wx.Button(panelhorz, label="Button3")
```

2. 위젯

```
sizerhorz = wx.BoxSizer(wx.HORIZONTAL)
sizerhorz.Add(btn1)
sizerhorz.Add(btn2)
sizerhorz.Add(btn3)
panelhorz.SetSizer(sizerhorz)

panelvert = wx.Panel(frame)
static1 = wx.StaticText(panelvert, label="StaticText1")
static2 = wx.StaticText(panelvert, label="StaticText2")
static3 = wx.StaticText(panelvert, label="StaticText3")
sizervert = wx.BoxSizer(wx.VERTICAL)
sizervert.Add(static1)
sizervert.Add(static2)
sizervert.Add(static3)
panelvert.SetSizer(sizervert)

box = wx.BoxSizer(wx.VERTICAL)
frame.SetSizer(box)
box.Add(panelhorz, border = 10, flag = wx.DOWN)
box.Add(panelvert, border = 10, flag = wx.UP)

frame.Show(True)
app.MainLoop()
```

2. 위젯


wxPython
Button1
Button2
Button3
StaticText1
StaticText2
StaticText3

2. 위젯

❖ 텍스트

▪ TextCtrl 위젯

• 문자열을 입력받음

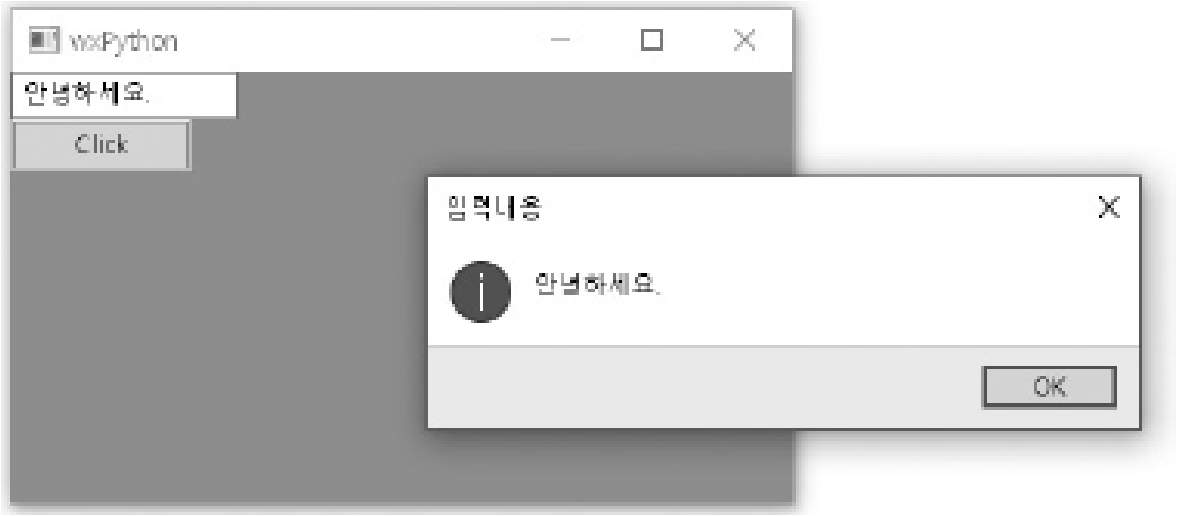
스타일	설명
TE_MULTILINE	여러 줄을 입력받는다.
TE_PASSWORD	입력받은 문자열을 *로 표시한다.
TE_READONLY	읽기만 가능하며 편집은 할 수 없다.
TE_LEFT, TE_CENTRE, TE_RIGHT	정렬 방식을 지정한다. 디폴트는 왼쪽이다.
TE_NOHIDESEL	포커스를 잃어도 선택 영역을 숨기지 않는다.
TE_RICH	Win32 환경에서 64K 이상의 텍스트를 편집할 수 있다.

2. 위젯

Text

```
import wx

app = wx.App()
frame = wx.Frame(None, title = "wxPython")

text = wx.TextCtrl(frame)
btn = wx.Button(frame, label="Click")
def onClick(event):
    str = text.GetValue()
    wx.MessageBox(str, "입력내용", wx.OK)
btn.Bind(wx.EVT_BUTTON, onClick)

box = wx.BoxSizer(wx.VERTICAL)
frame.SetSizer(box)
box.Add(text)
box.Add(btn)

frame.Show(True)
app.MainLoop()
```

2. 위젯



- 편집 즉시 문자열 알고 싶을 경우 EVT_TEXT 이벤트 처리

2. 위젯

❖ 체크 박스

- CHK_2STATE 스타일
  - 두 가지 상태 중 하나 입력받음
- CHK_3STATE 스타일
  - 세 가지 상태 중 하나 입력받음

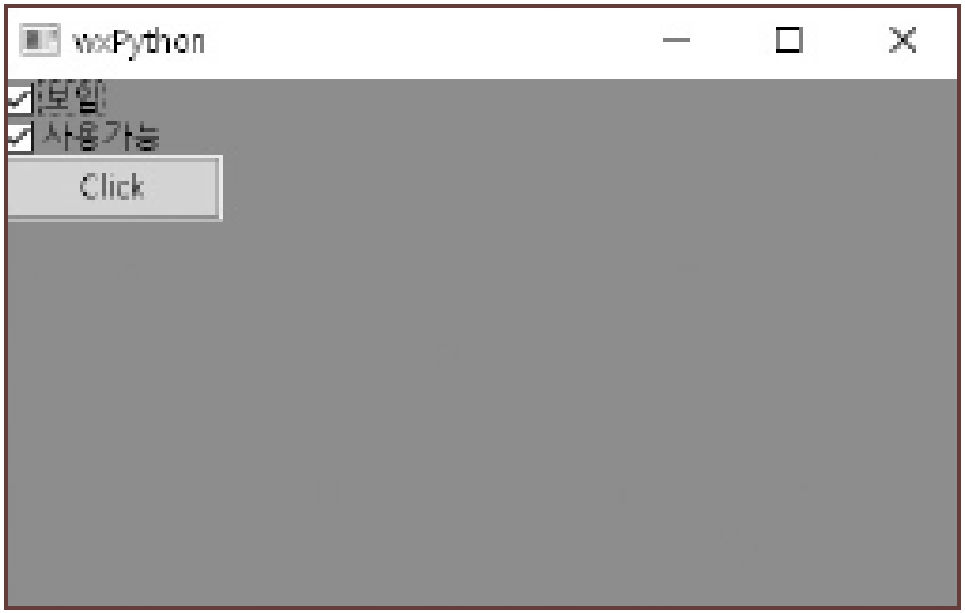
- 변경 즉시 동작 원하는 경우 EVT_CHECKHBOX 이벤트 처리

2. 위젯

CheckBox

```
import wx

app = wx.App()
frame = wx.Frame(None, title = "wxPython")

chkvisible = wx.CheckBox(frame, label = "보임")
chkvisible.SetValue(wx.CHK_CHECKED)
def onvisible(event):
    if chkvisible.GetValue() == wx.CHK_CHECKED:
        btn.Show(True)
    else:
        btn.Show(False)
chkvisible.Bind(wx.EVT_CHECKBOX, onvisible)

chkenable = wx.CheckBox(frame, label = "사용가능")
chkenable .SetValue(wx.CHK_CHECKED)
def onenable(event):
    if chkenable.GetValue() == wx.CHK_CHECKED:
        btn.Enable(True)
    else:
        btn.Enable(False)
chkenable.Bind(wx.EVT_CHECKBOX, onenable)

btn = wx.Button(frame, label="Click")
def onClick(event):
    wx.MessageBox("클릭했습니다.", "알림", wx.OK)
btn.Bind(wx.EVT_BUTTON, onClick)

box = wx.BoxSizer(wx.VERTICAL)
frame.SetSizer(box)
box.Add(chkvisible)
box.Add(chkenable)
box.Add(btn)

frame.Show(True)
app.MainLoop()
```

2. 위젯



2. 위젯

❖ 라디오 버튼

- 선택 가능한 여러 값 중 하나 고름

RadioButton

```
import wx

app = wx.App()
frame = wx.Frame(None, title = "wxPython")

red = wx.RadioButton(frame, label = "빨강", style = wx.RB_GROUP)
def onred(event):
    frame.SetBackgroundColour(wx.Colour(255, 0, 0, 0))
    frame.Refresh()
red.Bind(wx.EVT_RADIOBUTTON, onred)

green = wx.RadioButton(frame, label = "초록",)
def ongreen(event):
    frame.SetBackgroundColour(wx.Colour(0, 255, 0, 0))
    frame.Refresh()
green.Bind(wx.EVT_RADIOBUTTON, ongreen)
```

2. 위젯

```
blue = wx.RadioButton(frame, label = "파랑",)
def onblue(event):
    frame.SetBackgroundColour(wx.Colour(0, 0, 255, 0))
    frame.Refresh()
blue.Bind(wx.EVT_RADIOBUTTON, onblue)

yellow = wx.RadioButton(frame, label = "노랑",)
def onyellow(event):
    frame.SetBackgroundColour(wx.Colour(255, 255, 0, 0))
    frame.Refresh()
yellow.Bind(wx.EVT_RADIOBUTTON, onyellow)
yellow.SetValue(True)

box = wx.StaticBoxSizer(wx.VERTICAL, frame, "배경색")
frame.SetSizer(box)
box.Add(red)
box.Add(green)
box.Add(blue)
box.Add(yellow)

frame.SetBackgroundColour(wx.Colour(255, 255, 0, 0))
frame.Show(True)
app.MainLoop()
```

2. 위젯

❖ 메뉴

- 프로그램 기능 전체 선택받는 장치
- MenuBar
  - 윈도우 상단에 표시되는 메뉴 바
- MenuItem
  - Menu에 Append하고 Menu는 Menubar에 Append
- SetMenuBar 메서드
  - 위에서 만든 메뉴 바를 붙임

2. 위젯

MenuBar

```
import wx

app = wx.App()
frame = wx.Frame(None, title = "wxPython")

menubar = wx.MenuBar()
file = wx.Menu()
menubar.Append(file, "파일")
about = wx.MenuItem(id = wx.ID_ANY, text="소개")
file.Append(about)
file.AppendSeparator()
exit = wx.MenuItem(id = wx.ID_ANY, text="종료")
file.Append(exit)
frame.SetMenuBar(menubar)

def onAbout(event):
    wx.MessageBox("메뉴를 사용하는 프로그램입니다.", "알림", wx.OK)
frame.Bind(wx.EVT_MENU, onAbout, about)

def onExit(event):
    frame.Close()
frame.Bind(wx.EVT_MENU, onExit, exit)

frame.Show(True)
app.MainLoop()
```

Thank You !

파이썬 정복