

# Windows Application With QT Framework {PySide6}

# การพัฒนาโปรแกรม GUI

- Tkinter
- PyQt
- PySide

# การพัฒนาโปรแกรม GUI

- GUI ย่อมาจาก Graphical User Interface
- เป็นการใช้กราฟิกทำงานโดยต้องบวกกับผู้ใช้เพื่อรับคำสั่งและควบคุมโปรแกรม โดยการใช้ไอคอนรูปภาพ ปุ่ม เมนู และการตกลแต่งส่วนการทำงานบนหน้าจอที่สวยงาม
- ช่วยให้ใช้งานโปรแกรมได้ง่ายและรวดเร็วขึ้น
- สำหรับภาษาโปรแกรมจะมีเฟรมเวิร์ก (Framework หรือกรอบการทำงาน) ที่สนับสนุนในการสร้างโปรแกรมแบบ GUI เพื่อใช้ในงานด้านต่าง ๆ
- เช่น การสร้างเกมส์ แอโนเมชัน กราฟิกทั้งแบบ 2 มิติ/3 มิติ และโปรแกรมประยุกต์ต่าง ๆ

# เฟรมเวิร์කสำหรับสร้างโปรแกรม GUI

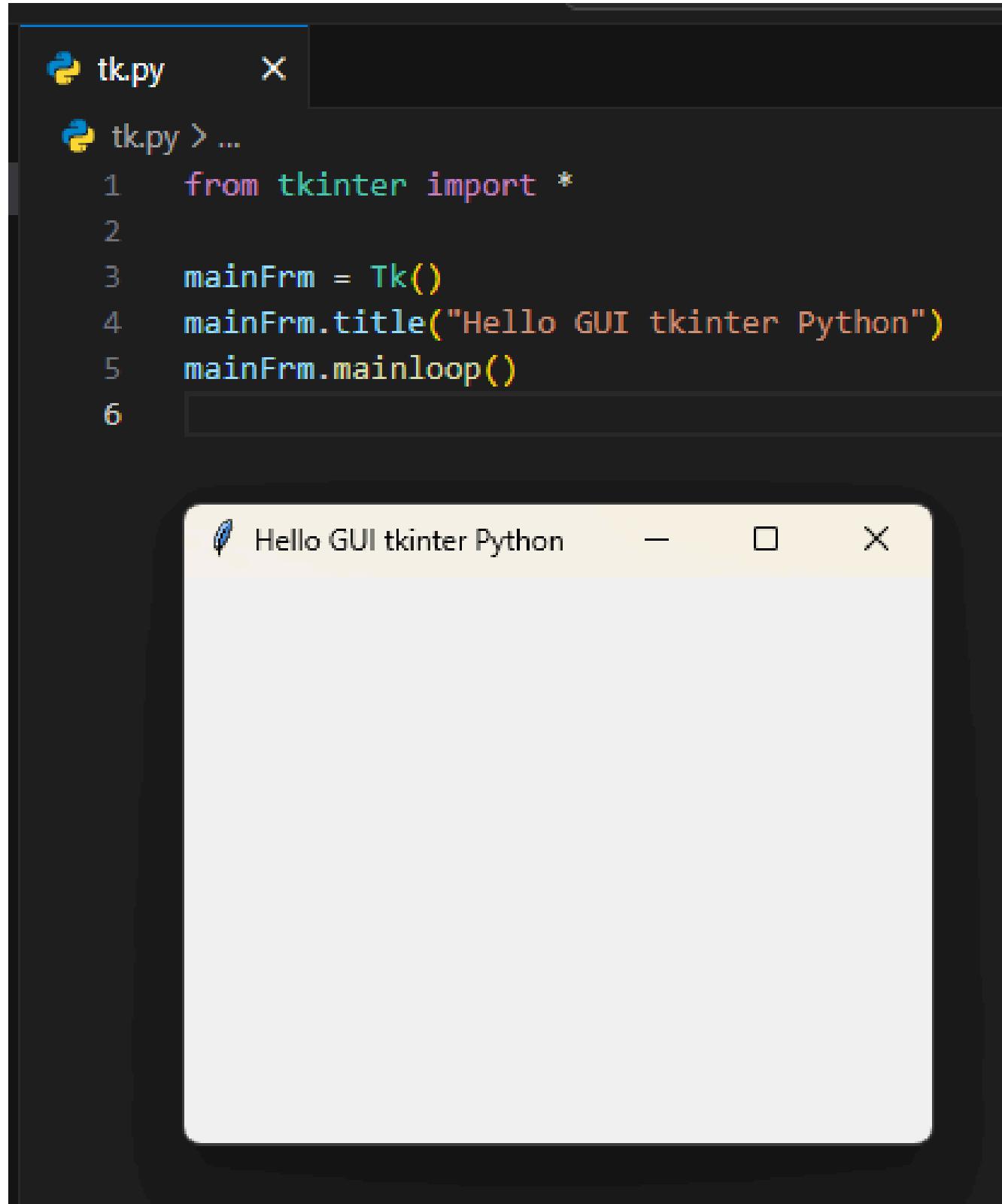
- เฟรมเวิร์ก (Framework) คือ เครื่องมือหรือชุดคำสั่งสนับสนุนการพัฒนาโปรแกรมสำหรับงานด้านใดด้านหนึ่ง โดยมีเครื่องมือ พังก์ชัน ไลบรารี หรือส่วนติดต่อผู้ใช้งาน พร้อมกับว่างมาตรฐานในการสร้างที่มีกรอบชัดเจน
- รวมทั้งมีส่วนแปลภาษาหรือแปลโปรแกรมให้สามารถนำไปใช้งานได้ตามกรอบที่วางไว้อย่างถูกต้อง ซึ่งช่วยให้เราพัฒนาโปรแกรมได้ง่าย และรวดเร็วขึ้น ไม่ต้องมาออกแบบและเขียนโค้ดเองทั้งหมดให้ยุ่งยาก
- ดังนั้นในการสร้างโปรแกรมแบบ GUI ด้วยภาษาไพรอนนี้ ก็จะมีเฟรมเวิร์กร่วมไว้เป็นไลบรารีให้นำมาใช้งานกันหลายชุดด้วยกัน

# Tkinter

- Tkinter เป็นแพ็คเกจ GUI มาตรฐานของภาษา Python ที่ถูกติดตั้งมาพร้อมชุดพัฒนาโปรแกรมภาษา Python สำหรับ Windows และ Mac OS X
- กำหนดให้เชื่อมต่อกับเครื่องมือ Tk GUI toolkit ซึ่งสร้างมาจากภาษา Tcl/Tk
- ความสามารถโดยรวมคล้ายกับวิดเจ็ต (Widgets) โปรแกรมขนาดเล็ก ประกอบด้วย ปุ่ม เช็คบัตตอน เรดิโอบัตตอน ลิสต์บี็อกซ์ สกรอลบี็อกซ์ โปรแกรมสารพัด และส่วนการทำงานอื่นๆ ซึ่งเหมาะสมสำหรับผู้เริ่มต้นสร้างโปรแกรม GUI ในภาษา Python

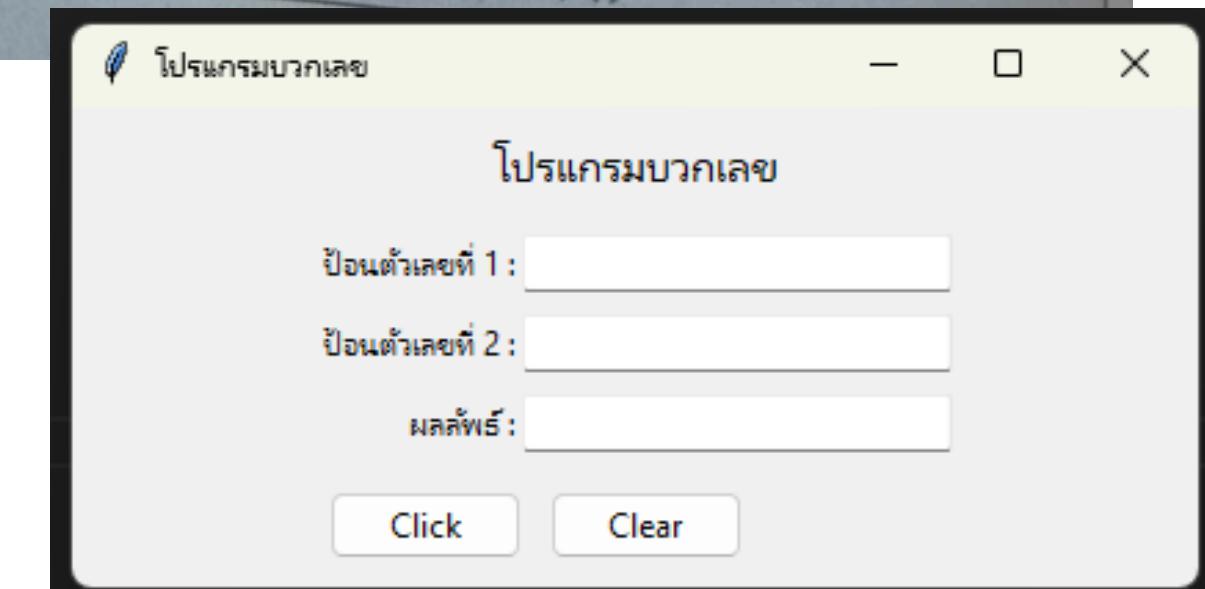
# Tkinter

ตัวอย่างที่ 12.2 คำสั่งโปรแกรมการสร้าง Frame เป็น parent และสร้าง widget เป็น child



```
from tkinter import * # โหลดโมดูล tkinter
from tkinter import ttk # นำเข้าคลาส ttk
mainFrm = Tk()
mainFrm.title("โปรแกรมบวกเลข")
# สร้าง Label widget
lblTitle = ttk.Label(mainFrm, text="โปรแกรมบวกเลข", font='20')
lblNum1 = ttk.Label(mainFrm, text="ป้อนตัวเลขที่ 1 :")
lblNum2 = ttk.Label(mainFrm, text="ป้อนตัวเลขที่ 2 :")
lblTotal = ttk.Label(mainFrm, text="ผลลัพธ์ :")
# จัดวางตำแหน่งให้กับ Label แต่ละตัวด้วยเมธอด grid()
lblTitle.grid(column=0, columnspan=2, padx = 150, pady=10)
lblNum1.grid(column=0, row=2, pady=5, sticky="NE")
lblNum2.grid(column=0, row=3, pady=5, sticky="NE")
lblTotal.grid(column=0, row=4, pady=5, sticky="NE")
# สร้าง Entry widget และจัดวางตำแหน่งด้วยเมธอด grid()
entNum1 = ttk.Entry(mainFrm, width=25).grid(column=1, row=2, sticky="W")
entNum2 = ttk.Entry(mainFrm, width=25).grid(column=1, row=3, sticky="W")
entTotal = ttk.Entry(mainFrm, width=25).grid(column=1, row=4, sticky="W")
# สร้าง Button widget
btnClick = ttk.Button(mainFrm, text="Click", width=10)
btnClear = ttk.Button(mainFrm, text="Clear", width=10)
# จัดวางตำแหน่งให้ Button widget แต่ละตัวด้วยเมธอด grid()
btnClick.grid(column=0, row=5, pady=10, sticky="NE")
btnClear.grid(column=1, row=5, padx=10, sticky="W")
mainFrm.mainloop() # แสดงหน้า GUI ด้วยเมธอด mainloop()
```

กีมา จากรังสือคู่มือการเขียนโปรแกรมภาษา Python ฉบับปรับปรุง โดย ณัฐวัตร คำภักดิ์



# PyQT

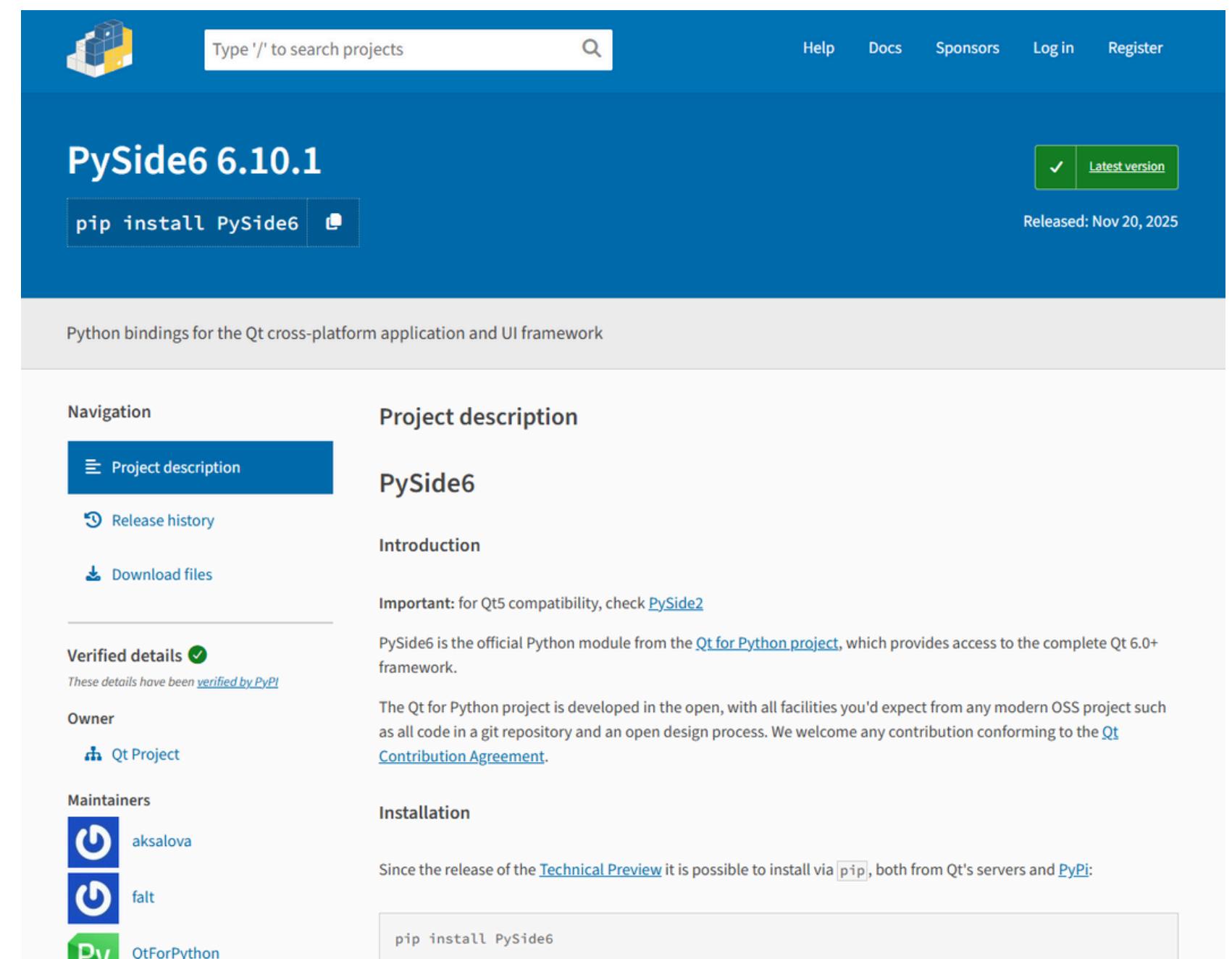
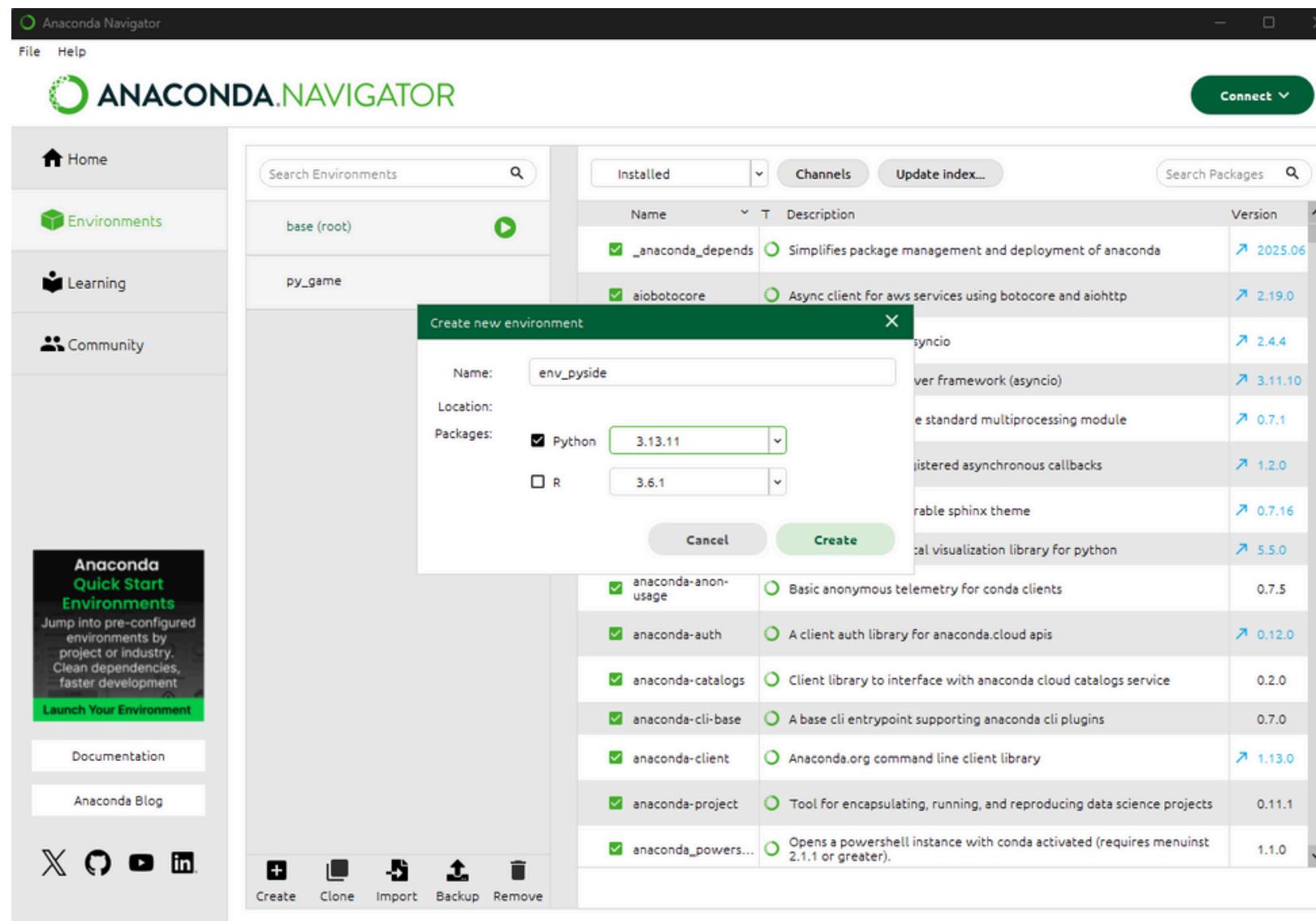
- PyQT เป็นไลบรารี Qt สำหรับพัฒนาโปรแกรม GUI ที่สามารถทำงานบนแพลตฟอร์มได้ทั้ง Windows, OS X, Linux, iOS และ Android มีทั้งเฟรมเวิร์กของ C++ และไฟร่อน PyQT
- ประกอบด้วยโมดูลและคลาสมากกว่าหนึ่งพันคลาส
- นอกจากระบบเครื่องมือการสร้างกราฟิกโดยต้องผู้ใช้งานแล้ว ยังสนับสนุนการทำงานกับระบบเครือข่าย Unicode, regular expressions, ฐานข้อมูล SQL, SVG, OpenGL, XML, เว็บเบราว์เซอร์ และมัลติมีเดียเฟรมเวิร์ก เป็นต้น
- ซึ่งเป็นไลบรารีที่ใช้งานเกี่ยวกับหน้าจอเดสก์ท็อป PyQT จะมีทั้งแบบโอเพ่นซอร์ส และมีライเซนส์

# PySide

- PySide คือชุดเครื่องมือ (binding) ที่ช่วยให้เราเขียนโปรแกรมสร้างส่วนติดต่อผู้ใช้ (GUI) สำหรับแอปพลิเคชันด้วยภาษา Python โดยใช้ライบรารี Qt ซึ่งเป็นเฟรมเวิร์กพัฒนาแอปพลิเคชันที่ได้รับความนิยมสูง ทำให้สร้างแอปพลิเคชันที่มีหน้าตาสวยงาม
- และทำงานได้บนหลายแพลตฟอร์ม (Windows, macOS, Linux) โดย PySide เป็นเวอร์ชันที่พัฒนาโดย Qt Company โดยตรง
- ต่างจาก PyQt ที่พัฒนาโดยนักพัฒนาภายนอก
- ทำให้บางครั้ง PySide มีความคล้ายคลึงกับ PyQt มาก จนผู้เรียนสามารถสลับใช้กันได้

# Install PySide

- Create Env Anaconda `env_pyside`
- <https://pypi.org/project/PySide6/>
- `pip install PySide6`



# First Program With PySide

## Creating an application

Let's create our first application! To start create a new Python file — you can call it whatever you like (e.g. `app.py`) and save it somewhere accessible. We'll write our simple app in this file.



We'll be editing within this file as we go along, and you may want to come back to earlier versions of your code, so remember to keep regular backups.

- **<https://www.pythonguis.com/pyside6>**

PYTHON

```
from PySide6.QtWidgets import QApplication, QWidget

# Only needed for access to command line arguments
import sys

# You need one (and only one) QApplication instance per application.
# Pass in sys.argv to allow command line arguments for your app.
# If you know you won't use command line arguments QApplication([]) works too.
app = QApplication(sys.argv)

# Create a Qt widget, which will be our window.
window = QWidget()
window.show() # IMPORTANT!!!! Windows are hidden by default.

# Start the event loop.
app.exec()

# Your application won't reach here until you exit and the event
# Loop has stopped.
```

# ทบทวนหลักการสำคัญของคลาส

```
class myClass(SuperClass):
    def __init__(self):
        super().__init__()
        ...
    
```

การใช้งาน `super` เป็น การเรียกใช้งาน `method` ของ `class` แม่โดยคลาสลูกต้องสืบทอดจากคลาสแม่เท่านั้นถึงจะเรียนใช้งานด้วย `method` ของ คลาสแม่ได้

```
class myClass(SuperClass):
    def __init__(self):
        super().__init__()

    def m1(self):
        pass

    def m2(self, a, b):
        self.m1()                      #เรียกเมธอด m1 () ผ่าน self
        ...
    
```

# สร้าง GUI ที่ถูกต้องตามหลักการของ Qt

```
import sys
from PySide6.QtWidgets import *

class myApp(QWidget):    # ใช้ชื่อคลาสอะไรก็ได้ แต่ให้สืบทอดจาก QWidget
    def __init__(self):
        super().__init__()    # หรือ super(myApp, self).__init__()
        self.resize(320, 240)  # กำหนดขนาดของหน้าต่างโปรแกรม
        self.setWindowTitle("Hello, World!")  # กำหนดชื่อต่างหน้าต่าง

if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = myApp()
    win.show()
    sys.exit(app.exec())
```

# signal slot events

**Signal and Slot** คือกลไกสำคัญที่ใช้สำหรับสื่อสารระหว่าง Object ต่างๆ โดยเฉพาะในการเขียนโปรแกรมแบบ GUI (Graphic User Interface) เพื่อตอบสนองต่อเหตุการณ์ (Events) ที่เกิดขึ้น

## 1. Signal (สัญญาณ)

คือ "ตัวส่ง" สัญญาณออกไปเมื่อมีเหตุการณ์บางอย่างเกิดขึ้น ตัวอย่างเช่น

- ผู้ใช้คลิกปุ่ม (clicked)
- ผู้ใช้พิมพ์ข้อความในช่อง Input (textChanged)
- ตัวจับเวลาทำงานครบกำหนด (timeout)

## 2. Slot (ช่องรับ)

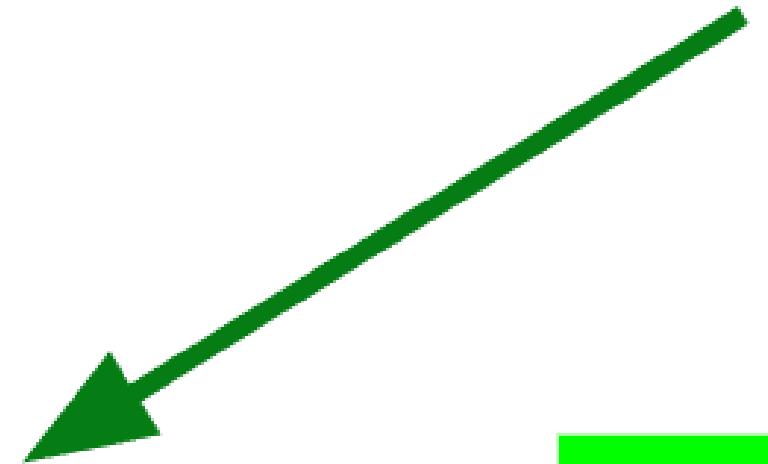
คือ "ฟังก์ชัน" หรือ "Method" ที่เราต้องการให้ทำงานเมื่อได้รับสัญญาณนั้นๆ คือฟังก์ชัน Python ปกติที่เราเขียนขึ้นมาเพื่อประมวลผลข้อมูลนั้นเอง

# signal slot events

```
class MainWindow( . . . ) :  
    . . .  
    btn.clicked.connect(self.btn_clicked)  
    . . .  
  
def btn_clicked(self) :  
    . . .
```

'signal'

'slot'



# Signals, Slots & Events (Example)

```
def window():
    app = QApplication(sys.argv)
    win = QDialog()
    b1 = QPushButton(win)
    b1.setText("Button1")
    b1.move(50,20)
    b1.clicked.connect(b1_clicked)

    b2 = QPushButton(win)
    b2.setText("Button2")
    b2.move(50,50)
    QObject.connect(b2,SIGNAL("clicked()"),b2_clicked)

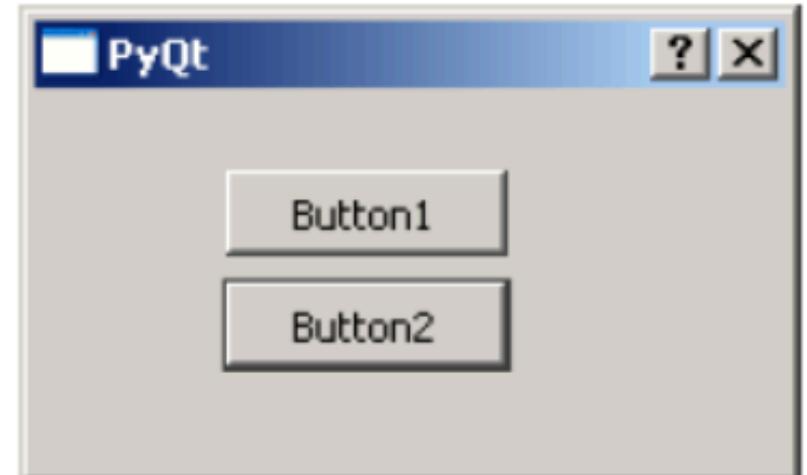
    win.setGeometry(100,100,200,100)
    win.setWindowTitle("PyQt")
    win.show()
    sys.exit(app.exec_())

def b1_clicked():
    print "Button 1 clicked"

def b2_clicked():
    print "Button 2 clicked"

if __name__ == '__main__':
    window()
```

The above code produces the following output –



Output

```
Button 1 clicked
Button 2 clicked
```

# Layout Manager

- Container (QApplication/QMainWindow/QWidget)
- VBox (QVBoxLayout)
- HBox (QHBoxLayout)
- Grid (QGridLayout)
- Nested Layout
- Stacked (QStackedLayout)
- QTabWidget
- QFormLayout

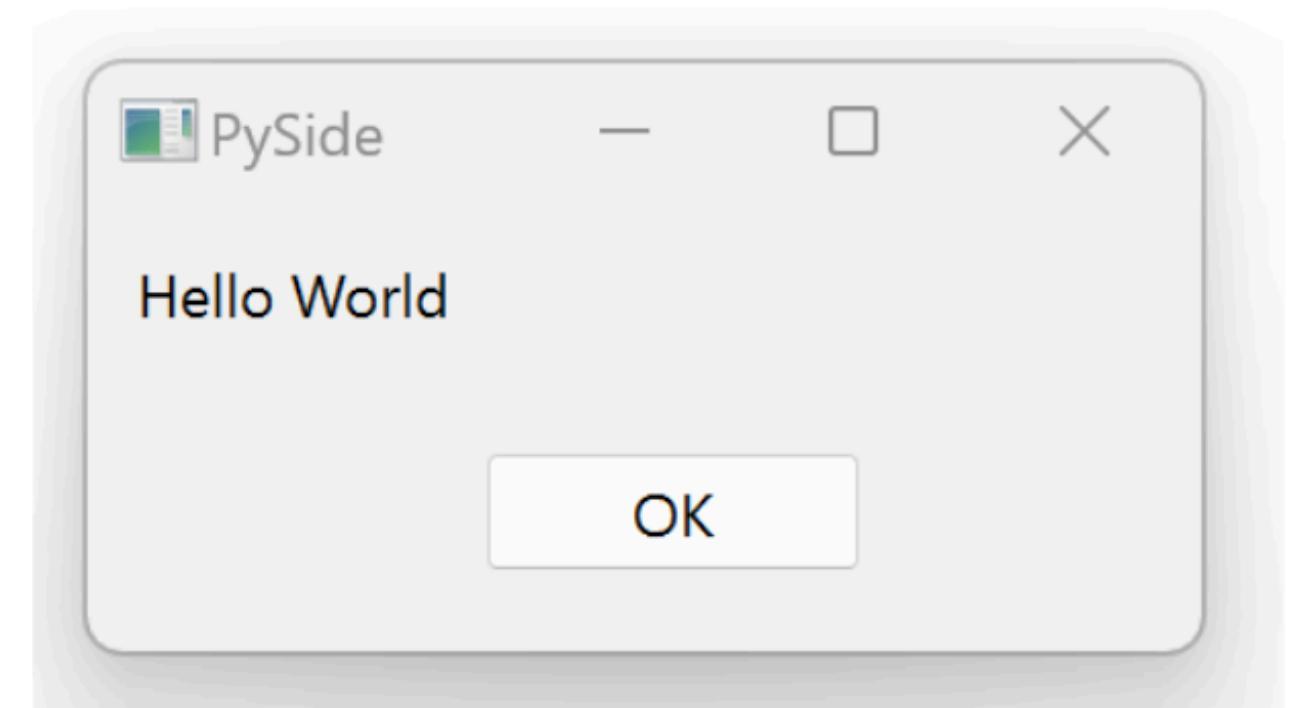
# Layout Manager :Absolute positioning

```
import sys
from PySide6.QtWidgets import *

class myApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("PySide") # กำหนดชื่อตระหัวโปรแกรม
        lbl = QLabel('Hello World', self) # self วินโดว์นี้เป็นคอนเท้นเนอร์
        lbl.move(10, 10)

        btn = QPushButton('OK', self)
        btn.move(80, 50)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = myApp()
    win.show()
    sys.exit(app.exec())
```



# Create Window ជាមួយ QWidget

## layout\_colorwidget.py

```
from PySide6.QtGui import QColor, QPalette
from PySide6.QtWidgets import QWidget

class Color(QWidget):
    def __init__(self, color: str):
        super().__init__()
        self.setAutoFillBackground(True)

        palette = self.palette()
        palette.setColor(QPalette.ColorRole.Window, QColor(color))
        self.setPalette(palette)
```

## app.py

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")

        app = QApplication(sys.argv)
        window = MainWindow()
        window.show()
        app.exec()
```

# Layout (VBox)

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout
from layout_colorwidget import Color

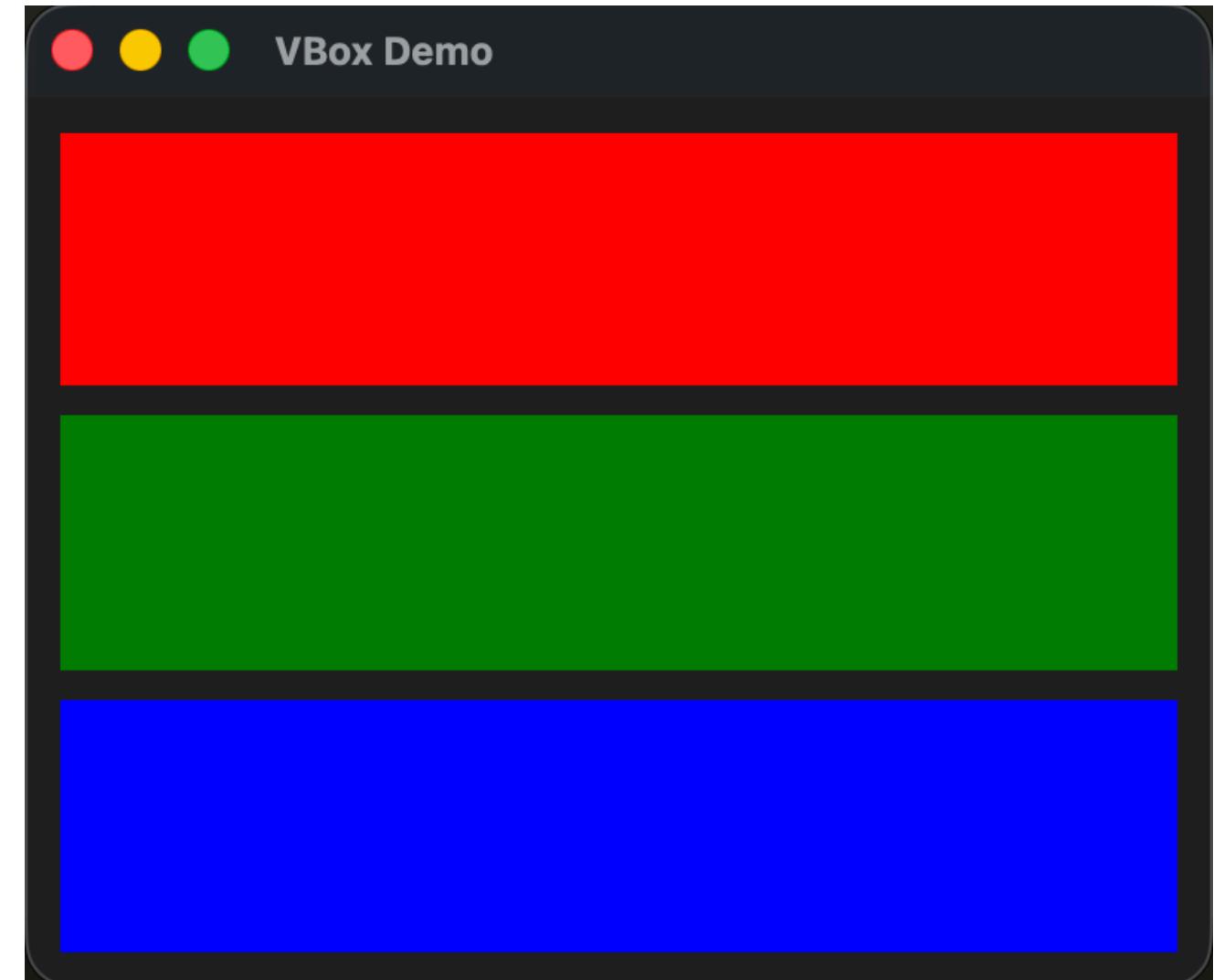
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("VBox Demo")

        layout = QVBoxLayout()
        layout.addWidget(Color("red"))
        layout.addWidget(Color("green"))
        layout.addWidget(Color("blue"))

        container = QWidget()
        container.setLayout(layout)
        self.setCentralWidget(container)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

ให้ บศ ลองเพิ่ม 1 สี (เช่น “yellow”)  
แล้วสังเกตสำลับ



# Layout (HBox)

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow, QWidget, QHBoxLayout
from layout_colorwidget import Color

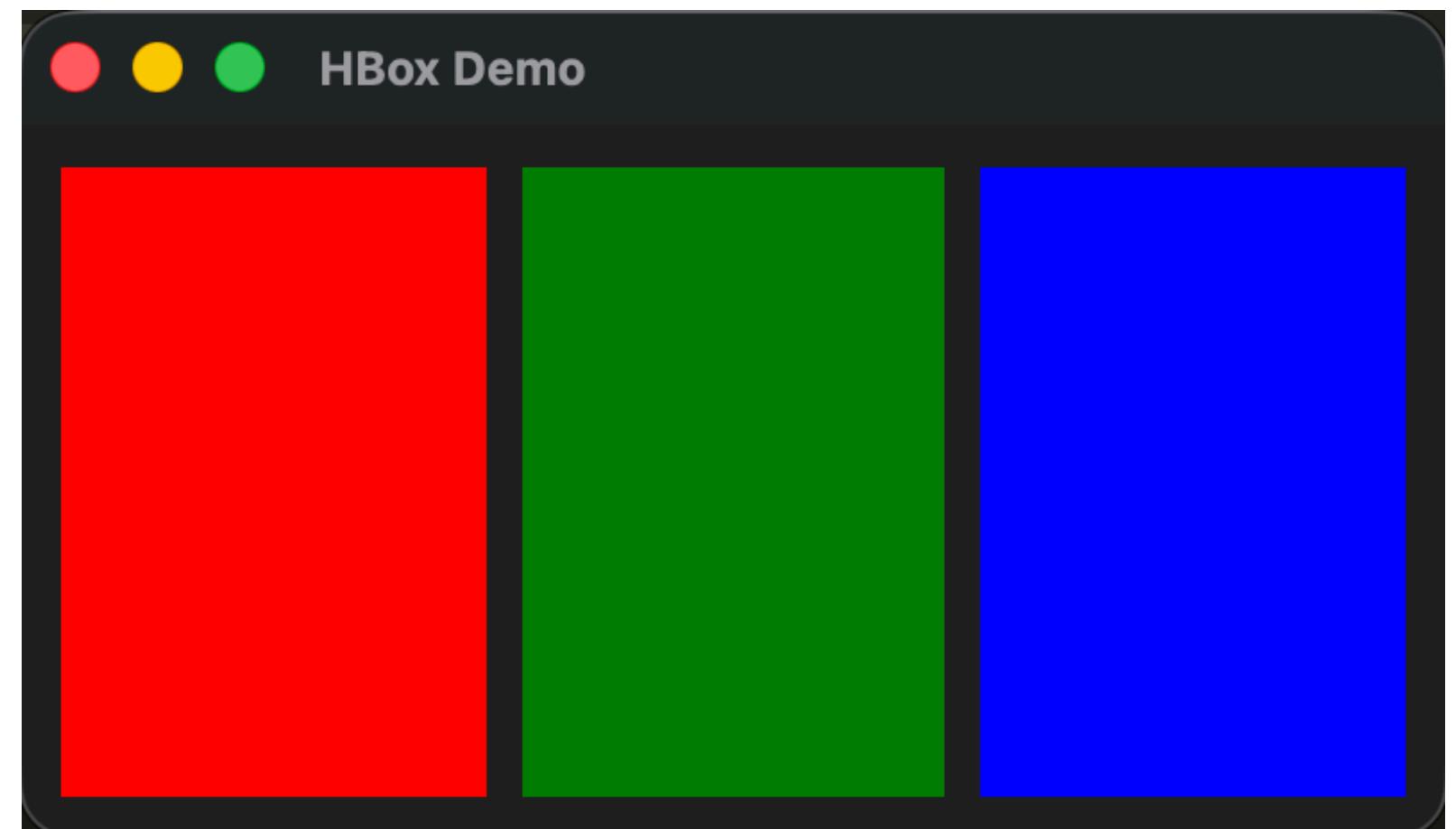
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("HBox Demo")

        layout = QHBoxLayout()
        layout.addWidget(Color("red"))
        layout.addWidget(Color("green"))
        layout.addWidget(Color("blue"))

        container = QWidget()
        container.setLayout(layout)
        self.setCentralWidget(container)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

ถ้าอยากรีบ widget ชิดกันมากขึ้น/ห่างขึ้น  
ปรับ setSpacing()  
ex.layout.setSpacing(10)



# Layout (Grid)

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow, QWidget, QGridLayout
from layout_colorwidget import Color

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Grid Demo")

        layout = QGridLayout()
        layout.addWidget(Color("red"), 0, 0)
        layout.addWidget(Color("green"), 1, 0)
        layout.addWidget(Color("blue"), 1, 1)
        layout.addWidget(Color("purple"), 2, 1)

        container = QWidget()
        container.setLayout(layout)
        self.setCentralWidget(container)

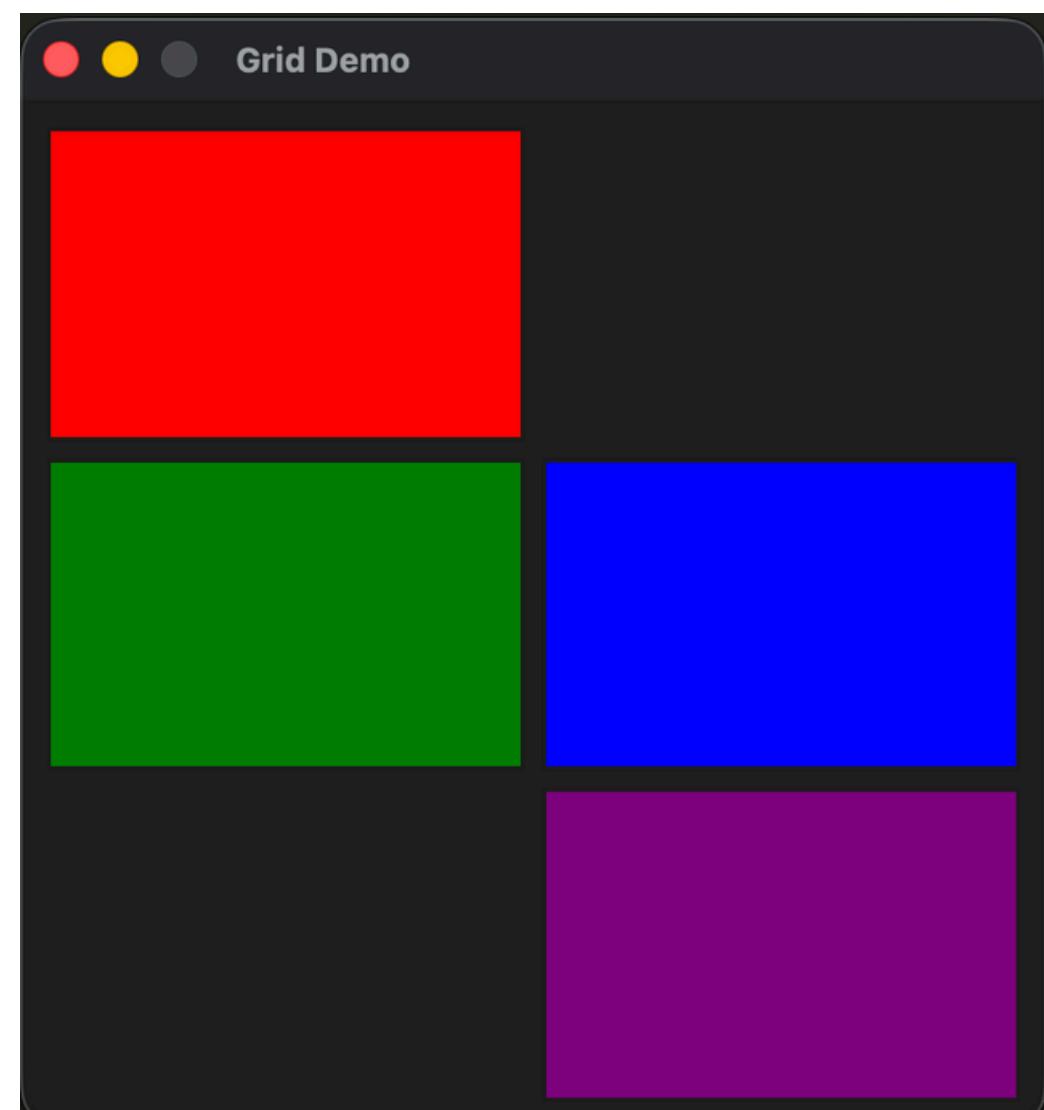
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    app.exec()
```

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

			0,3
	1,1		
		2,2	
3,0			

ให้บักศึกษาวางแผนสีให้เป็น

- รูปตัว L
- รูปขึ้นบันได



# Layout (Grid)

```
import sys
from PySide6.QtWidgets import QApplication, QWidget, QGridLayout, QPushButton

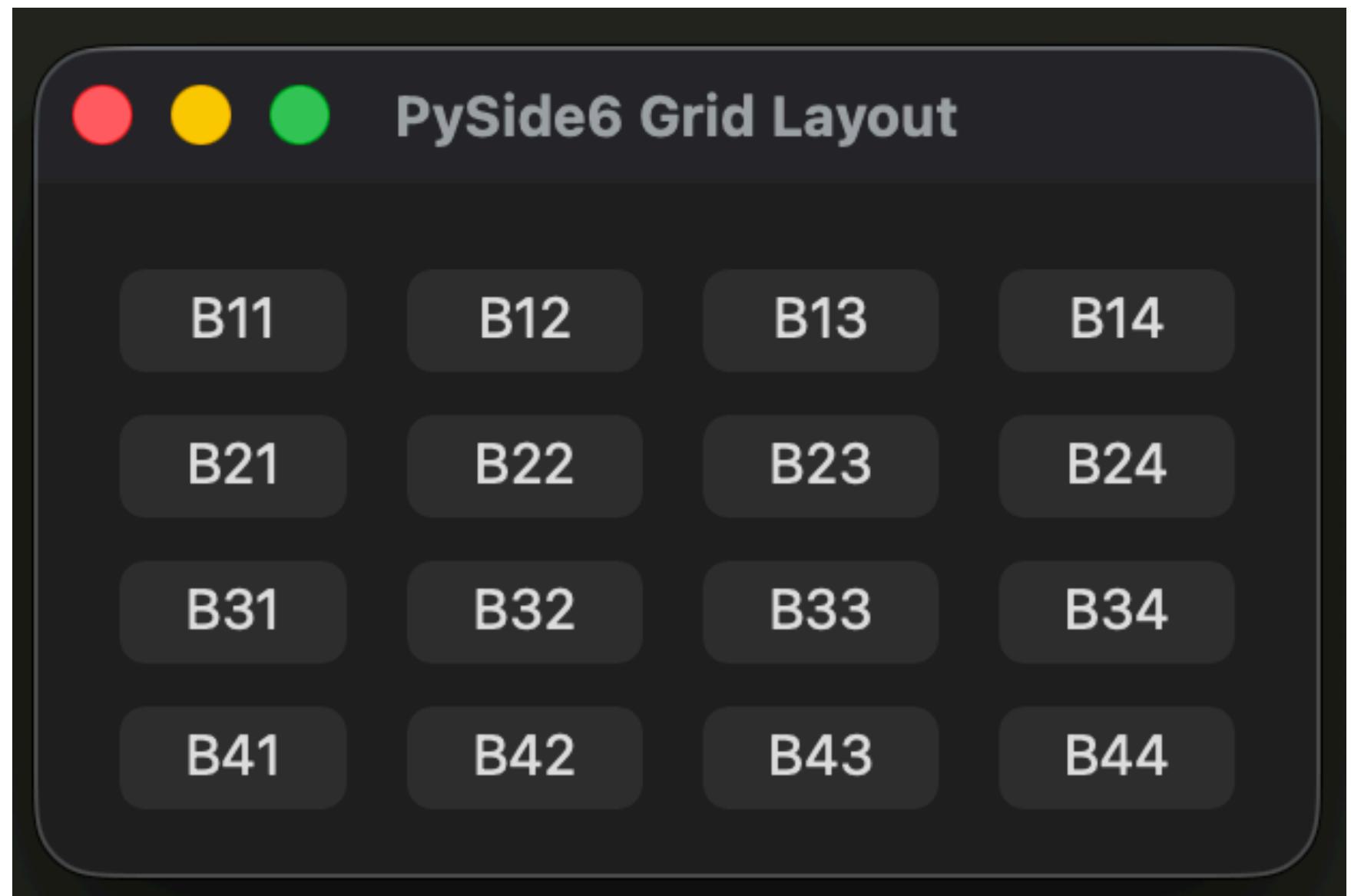
def window():
    app = QApplication(sys.argv)
    win = QWidget()
    grid = QGridLayout()

    for i in range(1, 5):
        for j in range(1, 5):
            button_name = f"B{i}{j}"
            button = QPushButton(button_name)
            grid.addWidget(button, i, j)

    win.setLayout(grid)
    win.setGeometry(100, 100, 300, 150)
    win.setWindowTitle("PySide6 Grid Layout")
    win.show()

    sys.exit(app.exec())

if __name__ == '__main__':
    window()
```



# Nesting layouts

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow, QWidget, QHBoxLayout,
    QVBoxLayout
from layout_colorwidget import Color

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Nesting Demo")

        layout_main = QHBoxLayout()
        layout_left = QVBoxLayout()
        layout_right = QVBoxLayout()

        layout_left.addWidget(Color("red"))
        layout_left.addWidget(Color("yellow"))
        layout_left.addWidget(Color("purple"))

        layout_main.addLayout(layout_left)
        layout_main.addWidget(Color("green"))

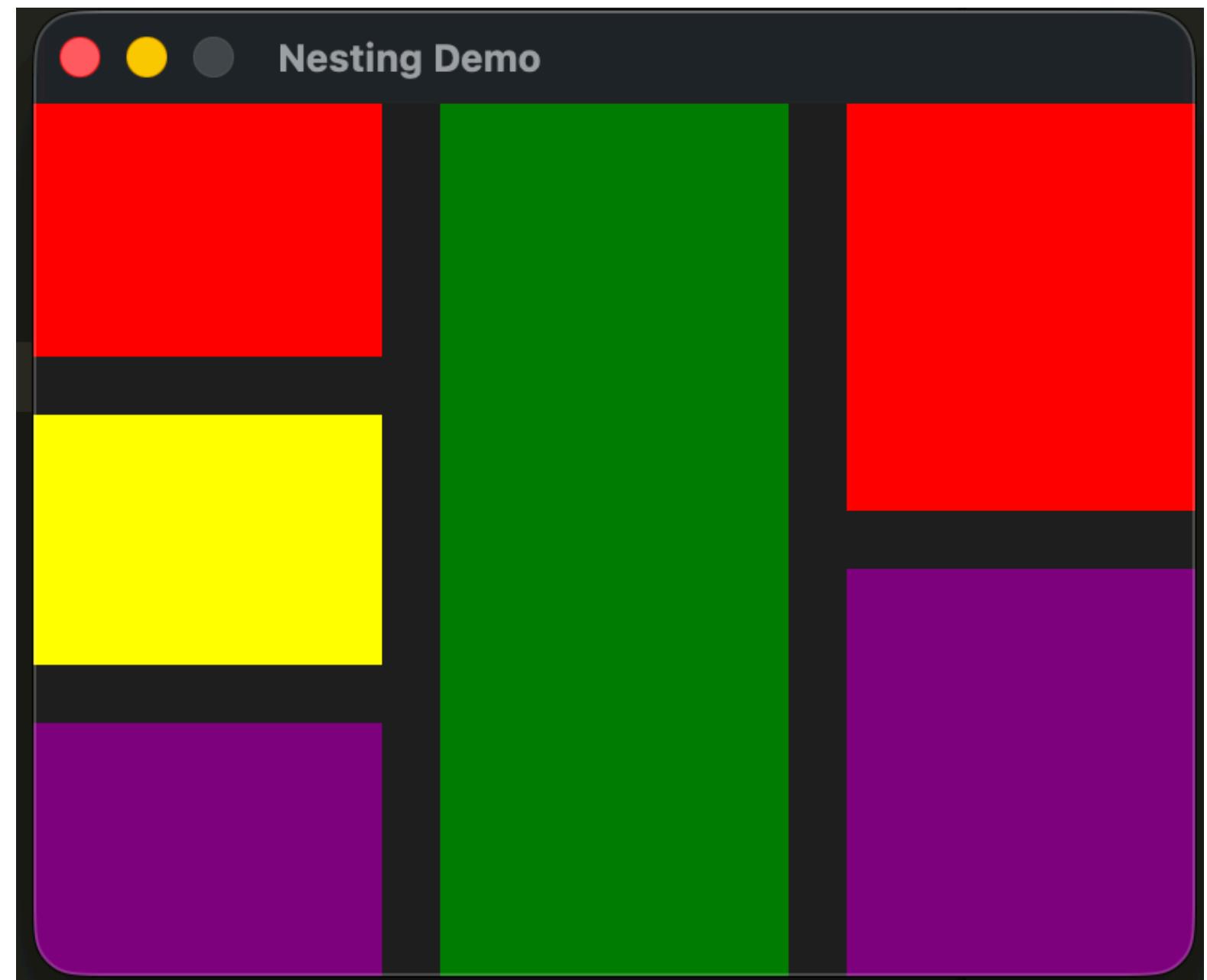
        layout_right.addWidget(Color("red"))
        layout_right.addWidget(Color("purple"))
        layout_main.addLayout(layout_right)

        layout_main.setContentsMargins(0, 0, 0, 0)
        layout_main.setSpacing(20)

        container = QWidget()
        container.setLayout(layout_main)
        self.setCentralWidget(container)

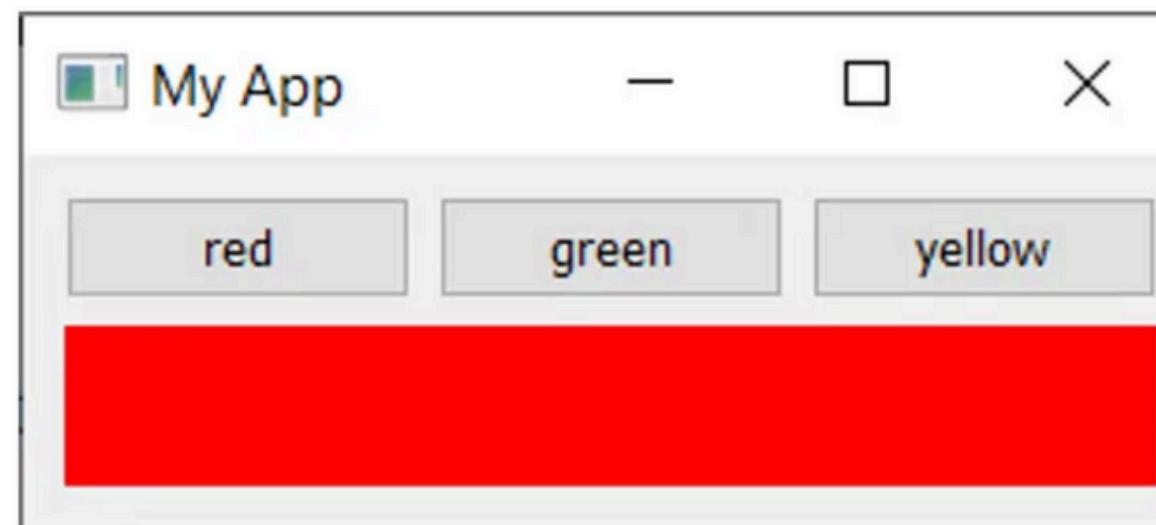
app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

- Layout ใหญ่เป็น “โครงหลัก”
- Layout ย่อยเป็น “กล่องบรรจุ/แควกภายใน”



# Layout Manager

## : Stacked (QStackedLayout)



```
def activate_tab_1(self):
    self.stacklayout.setCurrentIndex(0)

def activate_tab_2(self):
    self.stacklayout.setCurrentIndex(1)

def activate_tab_3(self):
    self.stacklayout.setCurrentIndex(2)
```

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")

        pagelayout = QVBoxLayout()
        button_layout = QHBoxLayout()
        self.stacklayout = QStackedLayout()

        pagelayout.addLayout(button_layout)
        pagelayout.addWidget(self.stacklayout)

        btn = QPushButton("red")
        btn.pressed.connect(self.activate_tab_1)
        button_layout.addWidget(btn)
        self.stacklayout.addWidget(Color("red"))

        btn = QPushButton("green")
        btn.pressed.connect(self.activate_tab_2)
        button_layout.addWidget(btn)
        self.stacklayout.addWidget(Color("green"))

        btn = QPushButton("yellow")
        btn.pressed.connect(self.activate_tab_3)
        button_layout.addWidget(btn)
        self.stacklayout.addWidget(Color("yellow"))

        widget = QWidget()
        widget.setLayout(pagelayout)
        self.setCentralWidget(widget)
```

# QStackedLayout + ปุ่ม

```
import sys
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QWidget,
    QVBoxLayout, QHBoxLayout, QPushButton, QStackedLayout
)
from PySide6.QtGui import QPalette, QColor

class Color(QWidget):
    def __init__(self, color):
        super().__init__()
        self.setAutoFillBackground(True)
        palette = self.palette()
        palette.setColor(QPalette.Window, QColor(color))
        self.setPalette(palette)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

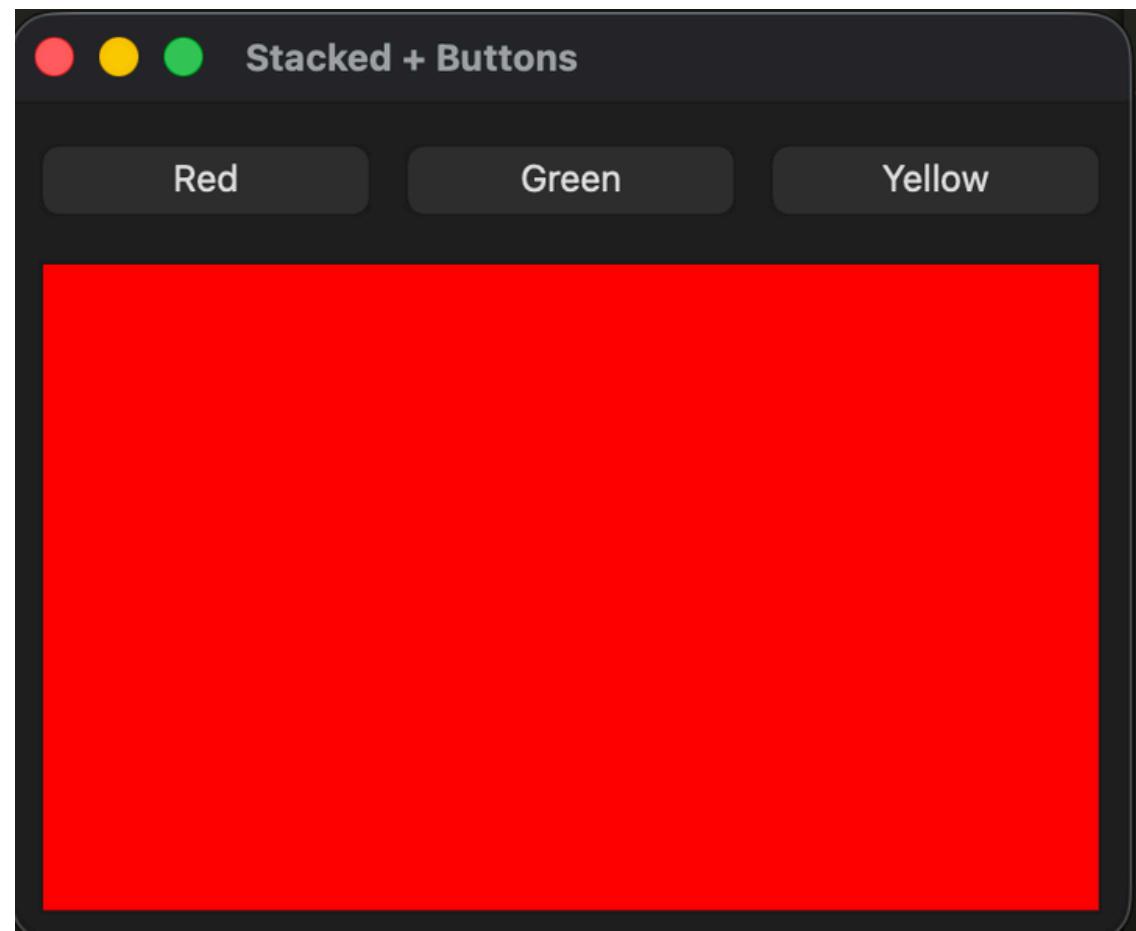
        self.setWindowTitle("Stacked + Buttons")
        self.resize(400, 300)

        page_layout = QVBoxLayout()
        button_layout = QHBoxLayout()
        self.stack = QStackedLayout()
        page_layout.addLayout(button_layout)
        page_layout.addWidget(self.stack)
        btn_red = QPushButton("Red")
        btn_red.pressed.connect(lambda: self.stack.setCurrentIndex(0))
        button_layout.addWidget(btn_red)
        self.stack.addWidget(Color("red"))
        btn_green = QPushButton("Green")
        btn_green.pressed.connect(lambda: self.stack.setCurrentIndex(1))
        button_layout.addWidget(btn_green)
        self.stack.addWidget(Color("green"))
        btn_yellow = QPushButton("Yellow")
        btn_yellow.pressed.connect(lambda: self.stack.setCurrentIndex(2))
        button_layout.addWidget(btn_yellow)
        self.stack.addWidget(Color("yellow"))

        container = QWidget()
        container.setLayout(page_layout)
        self.setCentralWidget(container)

    if __name__ == "__main__":
        app = QApplication(sys.argv)
        window = MainWindow()
        window.show()
        sys.exit(app.exec())
```

- ใช้ lambda ลดจำนวนเมธอด activate\_tab\_1/2/3
- mapping “ปุ่มหนึ่งปุ่ม ต่อ index หนึ่ง index”
- QStackedLayout เหมาะเมื่ออยากรอคัวบตัวคุณเอง



# QTabWidget (แท็บสำเร็จรูป)

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow, QTabWidget
from layout_colorwidget import Color

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("QTabWidget Demo")

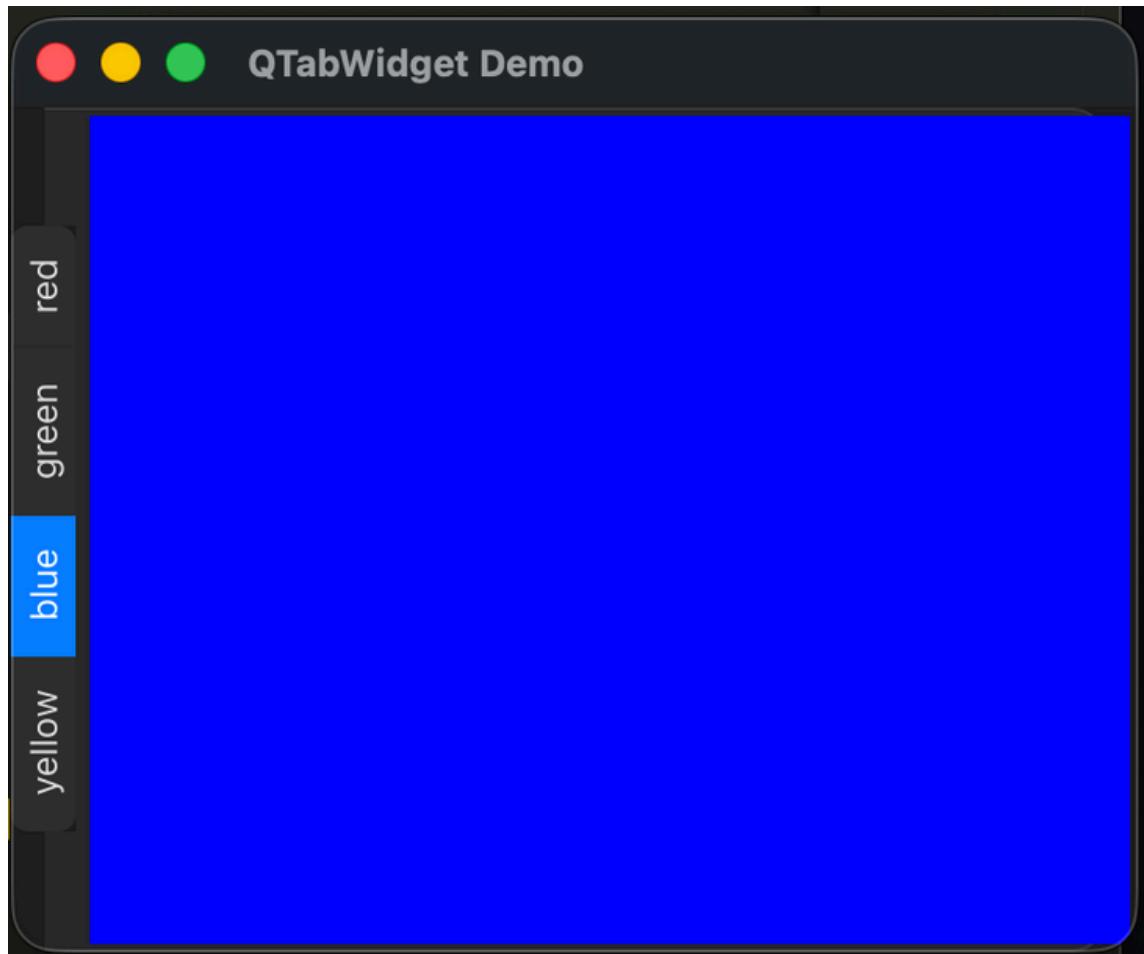
        tabs = QTabWidget()
        tabs.setTabPosition(QTabWidget.West)
        tabs.setMovable(True)

        for color in ["red", "green", "blue", "yellow"]:
            tabs.addTab(Color(color), color)

        self.setCentralWidget(tabs)

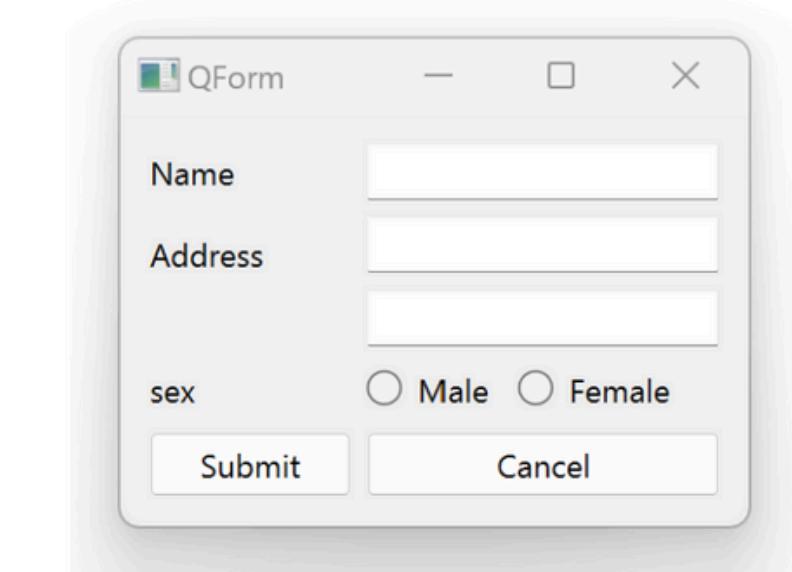
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    app.exec()
```

- ใช้ง่ายกว่า สวยกว่า แต่อาจปรับได้ไม่ยืดหยุ่นเท่า custom
- QTabWidget เหมาะเมื่อ UI เป็นแก๊บมาตรฐาน



# Layout Manager: QFormLayout

```
class myApp(QWidget):  
    def __init__(self):  
        super().__init__()  
        self.setWindowTitle("QForm")  
        l1 = QLabel("Name")  
        nm = QLineEdit()  
  
        l2 = QLabel("Address")  
        add1 = QLineEdit()  
        add2 = QLineEdit()  
        fbox = QFormLayout()  
        fbox.addRow(l1, nm)  
        vbox = QVBoxLayout()  
  
        vbox.addWidget(add1)  
        vbox.addWidget(add2)  
        fbox.addRow(l2, vbox)  
        hbox = QHBoxLayout()
```



```
        r1 = QRadioButton("Male")  
        r2 = QRadioButton("Female")  
        hbox.addWidget(r1)  
        hbox.addWidget(r2)  
        hbox.addStretch()  
        fbox.addRow(QLabel("sex"), hbox)  
        fbox.addRow(QPushButton("Submit"),  
                   QPushButton("Cancel"))  
  
    self.setLayout(fbox)
```

# QFormLayout

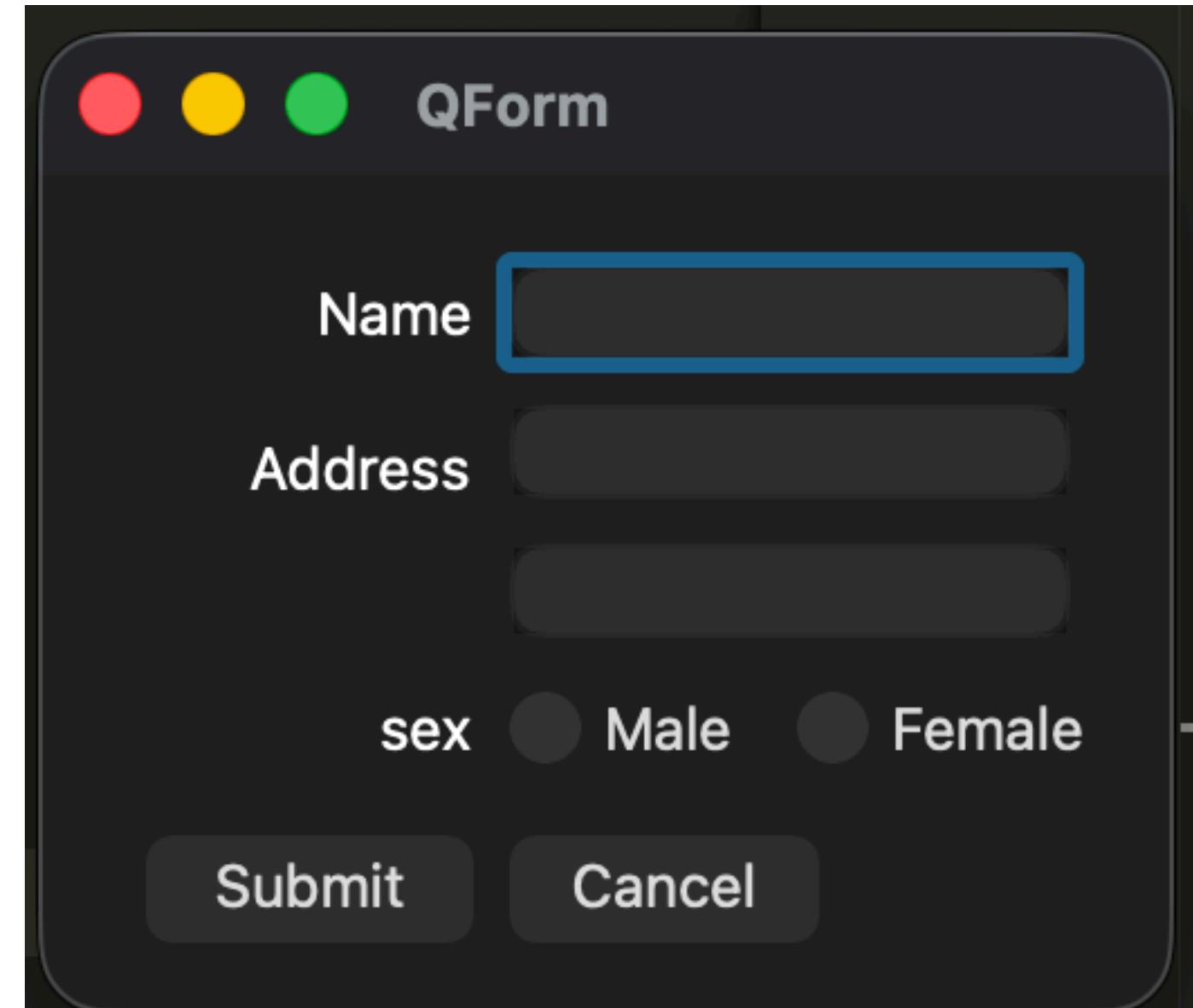
```
import sys
from PySide6.QtWidgets import QApplication, QWidget, QFormLayout, QLabel,
    QLineEdit, QVBoxLayout, QHBoxLayout,
    QRadioButton, QPushButton)

class myApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("QForm")

        fbox = QFormLayout()
        l1 = QLabel("Name")
        nm = QLineEdit()
        fbox.addRow(l1, nm)
        l2 = QLabel("Address")
        add1 = QLineEdit()
        add2 = QLineEdit()
        vbox = QVBoxLayout()
        vbox.addWidget(add1)
        vbox.addWidget(add2)
        fbox.addRow(l2, vbox)
        hbox = QHBoxLayout()
        r1 = QRadioButton("Male")
        r2 = QRadioButton("Female")
        hbox.addWidget(r1)
        hbox.addWidget(r2)
        hbox.addStretch()
        fbox.addRow(QLabel("sex"), hbox)
        fbox.addRow(QPushButton("Submit"), QPushButton("Cancel"))

        self.setLayout(fbox)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = myApp()
    window.show()
    sys.exit(app.exec())
```



# QFormLayout

```
class myApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("QForm with Submit Logic")

        fbox = QFormLayout()
        self.nm = QLineEdit()
        fbox.addRow(QLabel("Name"), self.nm)
        self.add1 = QLineEdit()
        self.add2 = QLineEdit()
        vbox = QVBoxLayout()
        vbox.addWidget(self.add1)
        vbox.addWidget(self.add2)
        fbox.addRow(QLabel("Address"), vbox)
        hbox = QHBoxLayout()
        self.r1 = QRadioButton("Male")
        self.r2 = QRadioButton("Female")
        hbox.addWidget(self.r1)
        hbox.addWidget(self.r2)
        hbox.addStretch()
        fbox.addRow(QLabel("sex"), hbox)
        self.btn_submit = QPushButton("Submit")
        self.btn_cancel = QPushButton("Cancel")
        self.btn_submit.clicked.connect(self.on_submit)
        self.btn_cancel.clicked.connect(self.close)
        fbox.addRow(self.btn_submit, self.btn_cancel)
        self.setLayout(fbox)

# พังก์ชันจัดการเมื่อกดปุ่ม Submit
def on_submit(self):
    name = self.nm.text()
    address = f'{self.add1.text()} {self.add2.text()}'"
    gender = ""
    if self.r1.isChecked():
        gender = "Male"
    elif self.r2.isChecked():
        gender = "Female"
    else:
        gender = "Not selected"

# สร้างข้อความที่จะแสดง
info = f'Name: {name}\nAddress: {address}\nSex: {gender}'

# แสดงกล่องร่องข้อความ (Message Box)
QMessageBox.information(self, "Submission Successful", info)
```

