TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

CHƯƠNG 8

LẬP TRÌNH GIAO DIỆN TRONG PYTHON

Nghệ An, 2019

Chương 8:
LẬP TRÌNH GIAO DIỆN TRONG PYTHON

NỘI DUNG GIẢNG DẠY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

Chương 8:
LẬP TRÌNH GIAO DIỆN TRONG PYTHON

NỘI DUNG GIẢNG DẠY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER

- **Khái niệm về chương trình có giao diện đồ họa**

- Chương trình có giao diện đồ hoạ là một chương trình luôn chạy cho tới khi người dùng thoát chương trình (*có chạy 1 vòng lặp vô hạn để luôn hiển thị giao diện, gọi là main loop*).

Chương trình đồ họa hoạt động dựa trên những tương tác của người dùng và phản ứng với các tương tác đó (*bấm nút nào thì thực hiện công việc tương ứng*). Loại chương trình như vậy thuộc loại mô hình *"Event-driven programming".*

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER

- **Khái niệm về chương trình có giao diện đồ họa**

- Trong chương trình đồ họa, các thao tác của người dùng được gọi là các event, các hành động tương ứng của chương trình (các function) được gọi là các callback, được gắn vào các bộ phận giao diện (gắn callback vào nút bấm thì khi ta bấm nút, callback sẽ được gọi).

- Các bộ phận giao diện như nút bấm, chữ, ô nhập ký tự ... được gọi là các widget.

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER

- **So sánh chương trình có giao diện kiểu dòng lệnh (Command Line Interface - CLI) và chương trình có giao diện đồ họa (Graphical User Interface - GUI)**

CLI	GUI
- Chiếm ít tài nguyên. - Người dùng có nhiều quyền xử lý hệ thống. - Chỉ việc gõ một vài dòng để thực hiện một việc.	- Dễ dàng hơn cho người dùng khi tương tác với ứng dụng. - Có khả năng hoạt động đa nhiệm.

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER

- **Giới thiệu thư viện Tkinter**

- Tkinter là một gói trong Python có chứa module Tk hỗ trợ cho việc lập trình GUI.

- Tk ban đầu được viết cho ngôn ngữ Tcl, sau đó Tkinter được viết ra để sử dụng Tk bằng trình thông dịch Tcl trên nền Python.

- Ngoài Tkinter ra còn có một số công cụ khác giúp tạo một ứng dụng GUI viết bằng Python như wxPython, PyQt, và PyGTK.

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER

Để tạo ra một một khung cửa sổ đơn giản trong Tkinter chỉ cần làm như sau:

```
from tkinter import Tk, Frame, BOTH


class Example(Frame):
 def __init__(self, parent):
 Frame.__init__(self, parent, background="white")
 self.parent = parent
 self.initUI()
```

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER

```
def initUI(self):
self.parent.title("Simple")
self.pack(fill=BOTH, expand=1)


root = Tk()
root.geometry("250x150+300+300")
app = Example(root)
root.mainloop()
```

Chương 8:
LẬP TRÌNH GIAO DIỆN TRONG PYTHON

NỘI DUNG GIẢNG DẠY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

QUẢN LÝ LAYOUT

Trong Tkinter có hai loại widget:

- Các widget thường như nút bấm, textbox…

- Containter, đây là những widget chứa các widget khác (còn đợc gọi là layout).

Trong Tkinter có 3 loại layout:

- **Place** là kiểu layout tự do, tức là bạn sẽ phải tự quy định vị trí cũng như kích thước của các widget.

- **Pack** sắp xếp các widget của bạn theo chiều ngang và chiều dọc.

- **Grid** sắp xếp widget theo dạng bảng.

QUẢN LÝ LAYOUT

- **Place:**

- Khi thiết kế giao diện thường dùng layout kiểu tự do:

- Người lập trình phải tự quy định vị trí và kích thước cho các widget trên màn hình, nếu kích thước cửa sổ thay đổi thì kích thước và vị trí của các widget không thay đổi.

- Thiết kế theo kiểu tự do cũng rất bất tiện khi muốn thay đổi, thêm hay bớt các widget thì hầu như sẽ phải sắp xếp lại toàn bộ widget.

QUẢN LÝ LAYOUT

Ví dụ:

```
import Tkinter as tk
import random
root = tk.Tk()
# width x height + x_offset + y_offset:
root.geometry("170x200+30+30")
languages = ['Python','Perl','C++','Java','Tcl/Tk']
labels = range(5)
```

QUẢN LÝ LAYOUT

Ví dụ:

```
for i in range(5):
    ct = [random.randrange(256) for x in range(3)]
    brightness = int(round(0.299*ct[0] + 0.587*ct[1] + 0.114*ct[2]))
    ct_hex = "%02x%02x%02x" % tuple(ct)
    bg_colour = '#' + "".join(ct_hex)
    l = tk.Label(root,
                 text=languages[i],
```

QUẢN LÝ LAYOUT

Ví dụ:

```
fg='White' if brightness < 120 else 'Black',
                bg=bg_colour)
   l.place(x = 20, y = 30 + i*30, width=120, hei
root.mainloop()
```

QUẢN LÝ LAYOUT

- **Pack**:

Dễ sử dụng nhất trong ba trình quản lý hình học của Tk và Tkinter.

- Thay vì phải khai báo chính xác vị trí của một widget sẽ xuất hiện trên màn hình hiển thị, chúng ta có thể khai báo vị trí của các widget bằng lệnh pack tương đối với nhau. Lệnh pack sẽ chăm sóc các chi tiết.

- Trình quản lý layout này bị hạn chế về khả năng so với các trình quản lý **Grid** và **Place**. Đối với các ứng dụng đơn giản, nó chắc chắn là người quản lý của sự lựa chọn. Ví dụ, các ứng dụng đơn giản như đặt một số vật dụng cạnh nhau hoặc đặt lên nhau.

QUẢN LÝ LAYOUT

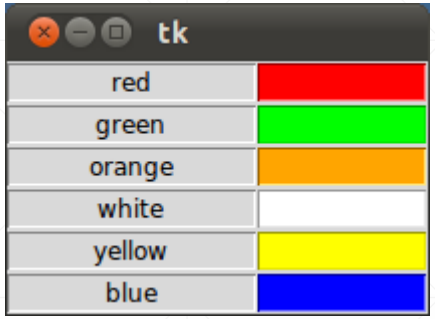
Ví dụ:

```
from Tkinter import *
root = Tk()
Label(root, text="Red Sun", bg="red", fg="white
Label(root, text="Green Grass", bg="green", fg=
Label(root, text="Blue Sky", bg="blue", fg="whit
mainloop()
```

QUẢN LÝ LAYOUT

Ví dụ:

```
from Tkinter import *
colours = ['red','green','orange','white','yellow','blue']
r = 0
for c in colours:
    Label(text=c, relief=RIDGE,width=15).grid(row=r,column=0)
    Entry(bg=c, relief=SUNKEN,width=10).grid(row=r,column=1)
    r = r + 1
mainloop()
```



QUẢN LÝ LAYOUT

- **Grid**:

Trong nhiều trường hợp là sự lựa chọn tốt nhất.

Grid đặt các widget trong bảng 2 chiều, bao gồm một số hàng và cột.

- Vị trí của một widget được xác định bởi một hàng và một số cột.

- Các widget có cùng số cột và số hàng khác nhau sẽ ở trên hoặc dưới nhau. Tương ứng, các vật dụng có cùng số hàng nhưng số cột khác nhau sẽ nằm trên cùng một "dòng" và sẽ nằm cạnh nhau, tức là ở bên trái hoặc bên phải.

Chương 8:
LẬP TRÌNH GIAO DIỆN TRONG PYTHON

NỘI DUNG GIẢNG DẠY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

WIDGET

Widget là các thành phần cấu tạo nên một ứng dụng GUI.

Widget rất đa dạng, có một số widget quan trọng cần phải có của bất kì nền tảng nào kể cả Tkinter ví dụ như button (nút bấm), check box hay scroll bar (thanh cuộn).

Ngoài những widget cơ bản lập trình viên còn có thể tùy chỉnh widget của riêng mình.

WIDGET

- **Checkbutton:** là widget hiển thị hộp đánh dấu.

Ví dụ:

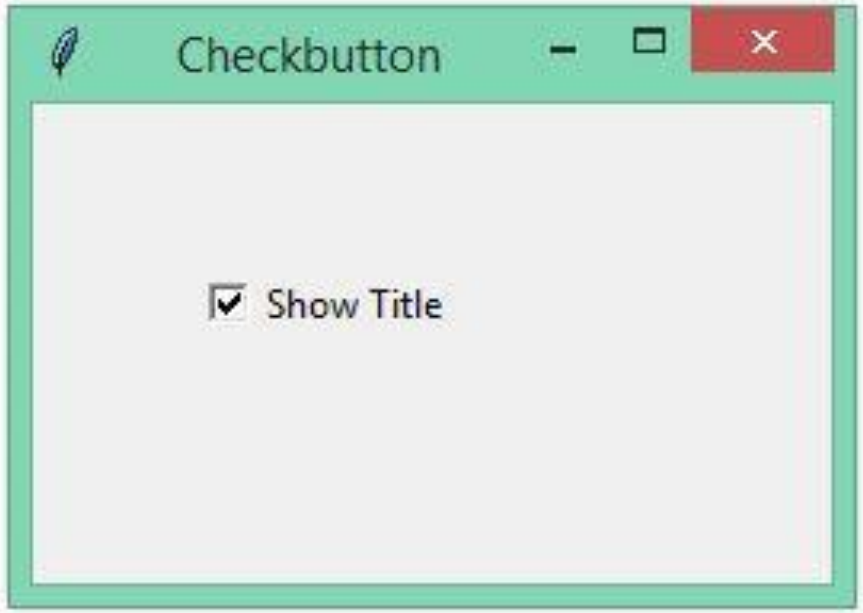
```
from tkinter import Tk, Frame, Checkbutton
from tkinter import BooleanVar, BOTH
 class Example(Frame):
  def __init__(self, parent):
    Frame.__init__(self, parent)
      self.parent = parent
    self.initUI()
```

WIDGET

```
    def initUI(self):
        self.parent.title("Checkbutton")
        self.pack(fill=BOTH, expand=True)
        self.var = BooleanVar()
        cb = Checkbutton(self, text="Show Title", variable=self.var,
command=self.onClick)

        cb.select()
        cb.place(x=50, y=50)
```

WIDGET

```
def onClick(self):
    if self.var.get() == True:
      self.master.title("Checkbutton")
    else:
      self.master.title("")
root = Tk()
root.geometry("250x150+300+300")
app = Example(root)
root.mainloop()
```



WIDGET

- **Label**: dùng để hiển thị text hoặc hình ảnh.

Ví dụ: dùng Label để hiển thị ảnh lên màn hình.

```
from PIL import Image, ImageTk
from tkinter import Tk, Frame, Label
class Example(Frame):
  def __init__(self, parent):
    Frame.__init__(self, parent)
    self.parent = parent
    self.initUI()
```

WIDGET

```
def initUI(self):
    self.parent.title("Label")
    self.img = Image.open("C:\\tatras.jpg")
    tatras = ImageTk.PhotoImage(self.img)
    label = Label(self, image=tatras)
    label.image = tatras

    label.pack()
    self.pack()
```

WIDGET

```
def setGeometry(self):
    w, h = self.img.size
    self.parent.geometry(("%dx%d+300+300") % (w, h))


root = Tk()
ex = Example(root)
ex.setGeometry()
root.mainloop()
```

WIDGET

- **Scale:** hiển thị một thanh cuộn gắn với một khoảng giá trị nào đó.

Ví dụ:

```
from tkinter import Tk, BOTH, IntVar, LEFT
from tkinter.ttk import Frame, Label, Scale, Style
class Example(Frame):
  def __init__(self, parent):
    Frame.__init__(self, parent)
    self.parent = parent
    self.initUI()
```

WIDGET

```
def initUI(self):
    self.parent.title("Scale")
    self.style = Style()
    self.style.theme_use("default")
    self.pack(fill=BOTH, expand=1)
    scale = Scale(self, from_=0, to=100, command=self.onScale)
    scale.pack(side=LEFT, padx=15)
    self.var = IntVar()
    self.label = Label(self, text=0, textvariable=self.var)
    self.label.pack(side=LEFT)
```

WIDGET

```
def onScale(self, val):
    v = int(float(val))
    self.var.set(v)


root = Tk()
ex = Example(root)
root.geometry("250x100+300+300")
root.mainloop()
```

WIDGET

• **Listbox**: cho phép hiển thị một danh sách các item. Người dùng có thể chọn một hoặc nhiều item.

```
from tkinter.ttk import Frame, Label
from tkinter import Tk, BOTH, Listbox, StringVar, END
class Example(Frame):
  def __init__(self, parent):
    Frame.__init__(self, parent)
    self.parent = parent
    self.initUI()
```

WIDGET

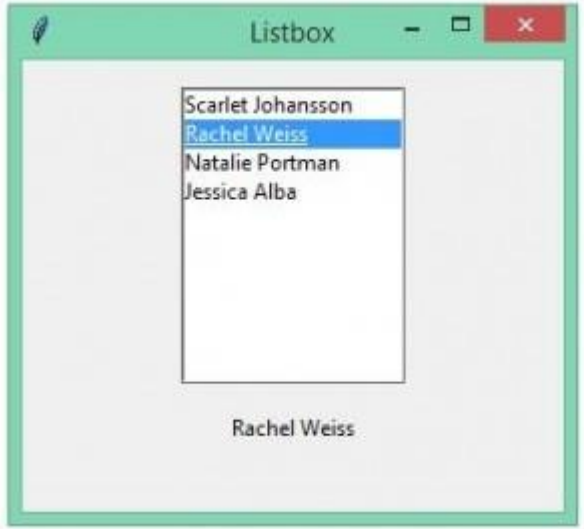
```
    def initUI(self):
        self.parent.title("Listbox")
        self.pack(fill=BOTH, expand=1)

        acts = ["Scarlet Johansson", "Rachel Weiss", "Natalie Portman",
"Jessica Alba"]

        lb = Listbox(self)
```

WIDGET

```
for i in acts:
        lb.insert(END, i)
    lb.bind("<<ListboxSelect>>", self.onSelect)


    lb.pack(pady=15)


    self.var = StringVar()
    self.label = Label(self, text=0, textvariable=self.var)
    self.label.pack()
```

WIDGET

```
def onSelect(self, val):
    sender = val.widget
    idx = sender.curselection()
    value = sender.get(idx)
    self.var.set(value)
root = Tk()
ex = Example(root)
root.geometry("300x250+300+300")
root.mainloop()
```



THẢO LUẬN NHÓM

NỘI DUNG:

Cách tổ chức giao diện người dùng trong một số hệ điều hành thông dụng.

BÀI TẬP

NỘI DUNG:

Tạo một chương trình GUI giới thiệu về thông tin cá nhân.

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

Đọc các tài liệu về nội dung mục 8.4; 8.5 và 8.6.

Chương 8:
LẬP TRÌNH GIAO DIỆN TRONG PYTHON

NỘI DUNG GIẢNG DẠY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

8.7. Bài tập

MENU

Trình bày tất cả các lệnh và chức năng của ứng dụng, có sẵn cho người dùng thông qua giao diện người dùng.

Menu trong GUI được trình bày với sự kết hợp của văn bản và ký hiệu để thể hiện các lựa chọn bằng chuột (hoặc ngón tay trên màn hình cảm ứng) trên một trong các biểu tượng hoặc văn bản, một hành động sẽ được bắt đầu.

Một hành động hoặc hoạt động (lựa chọn) có thể, là mở hoặc lưu tệp, hoặc thoát hoặc thoát khỏi ứng dụng, ...

MENU

- **Ví dụ:** cách tạo menu trong Tkinter.

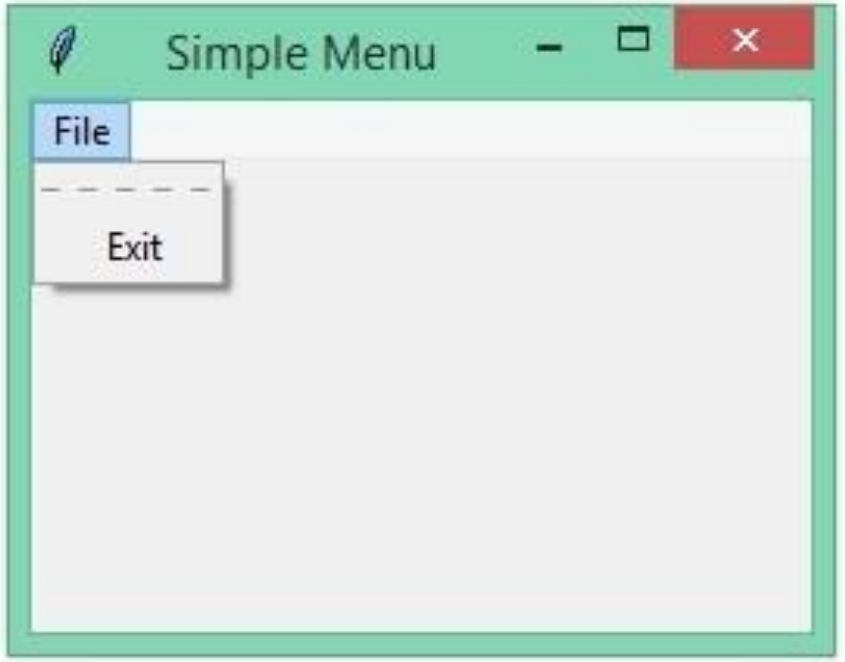
```
from tkinter import Frame, Tk, Menu


class Example(Frame):
  def __init__(self, parent):
    Frame.__init__(self, parent)

    self.parent = parent
    self.initUI()
```

MENU

```
def initUI(self):
    self.parent.title("Simple Menu")

    menuBar = Menu(self.parent)
    self.parent.config(menu=menuBar)

    fileMenu = Menu(menuBar)
    fileMenu.add_command(label="Exit", command=self.onExit)
    menuBar.add_cascade(label="File", menu=fileMenu)
```

MENU

```
def onExit(self):
    self.quit()


root = Tk()
root.geometry("250x150+300+300")
app = Example(root)
root.mainloop()
```



MENU

- **Tạo menu con:** tạo một menu con từ một menu cha.

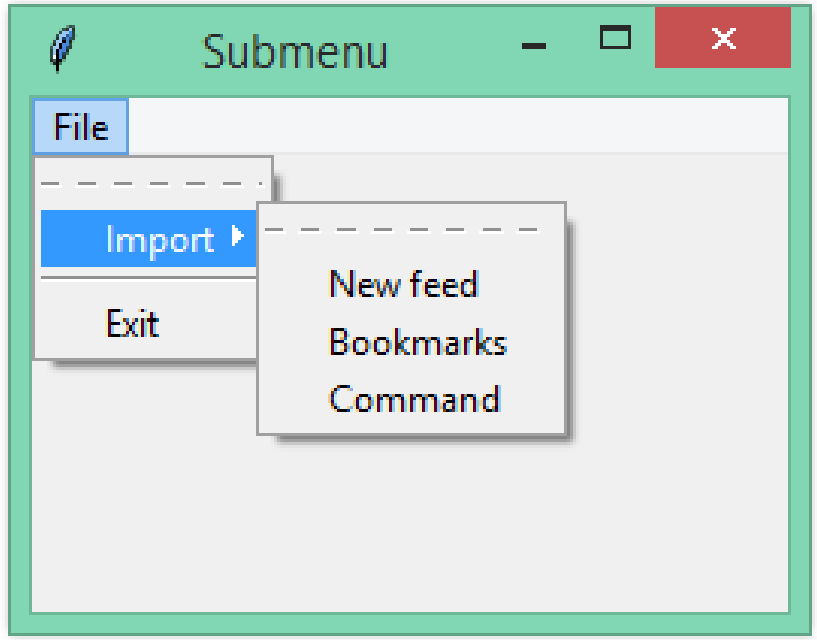
```
from Tkinter import Tk, Frame, Menu
class Example(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent

        self.initUI()
```

MENU

```
def initUI(self):
        self.parent.title("Submenu")
        menubar = Menu(self.parent)
        self.parent.config(menu=menubar)
        fileMenu = Menu(menubar)
        submenu = Menu(fileMenu)
        submenu.add_command(label="New feed")
        submenu.add_command(label="Bookmarks")
        submenu.add_command(label="Mail")
        fileMenu.add_cascade(label='Import', menu=submenu)
```

MENU

```
fileMenu.add_separator()
        fileMenu.add_command(label="Exit", command=self.onExit)
        menubar.add_cascade(label="File", menu=fileMenu)
    def onExit(self):
        self.quit()
root = Tk()
root.geometry("250x150+300+300")
app = Example(root)
root.mainloop()
```



MENU

• **Popup menu:** còn được gọi là menu ngữ cảnh là menu được hiện ra khi click chuột lên cửa sổ.

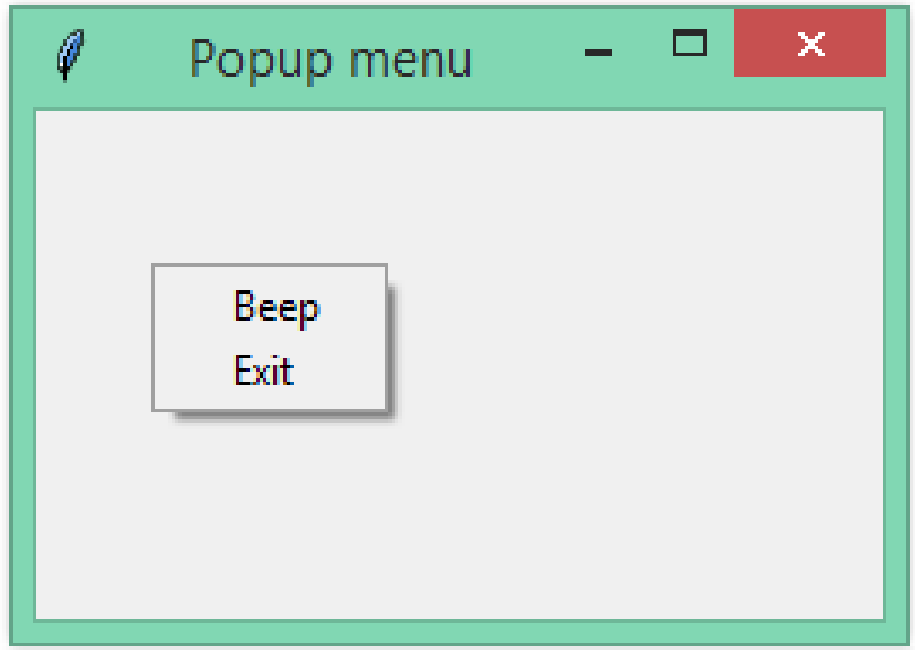
```
from Tkinter import Tk, Frame, Menu
class Example(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)

        self.parent = parent
        self.initUI()
```

MENU

```
def initUI(self):
        self.parent.title("Popup menu")
        self.menu = Menu(self.parent, tearoff=0)
        self.menu.add_command(label="Beep", command=self.bell())
        self.menu.add_command(label="Exit", command=self.onExit)
        self.parent.bind("<Button-3>", self.showMenu)
        self.pack()
def showMenu(self, e):
        self.menu.post(e.x_root, e.y_root)
```

MENU

```
def onExit(self):
        self.quit()



root = Tk()
root.geometry("250x150+300+300")
app = Example(root)
root.mainloop()
```



Chương 8:
LẬP TRÌNH GIAO DIỆN TRONG PYTHON

NỘI DUNG GIẢNG DẠY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

8.7. Bài tập

HỘP THOẠI

- Có thể được sử dụng để hiển thị các hộp thông báo, hiển thị cảnh báo hoặc lỗi hoặc widget để chọn tệp và màu sắc.

- Ngoài ra còn có các hộp thoại đơn giản, yêu cầu người dùng nhập chuỗi, số nguyên hoặc số float.

HỘP THOẠI

• **Messagebox:** dùng để hiển thị thông báo cho người dùng và đôi khi còn dùng để đưa ra yêu cầu chọn lựa cho người dùng.

```
from tkinter.ttk import Frame, Button
from tkinter import Tk, BOTH
import tkinter.messagebox as mbox
class Example(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()
```

HỘP THOẠI

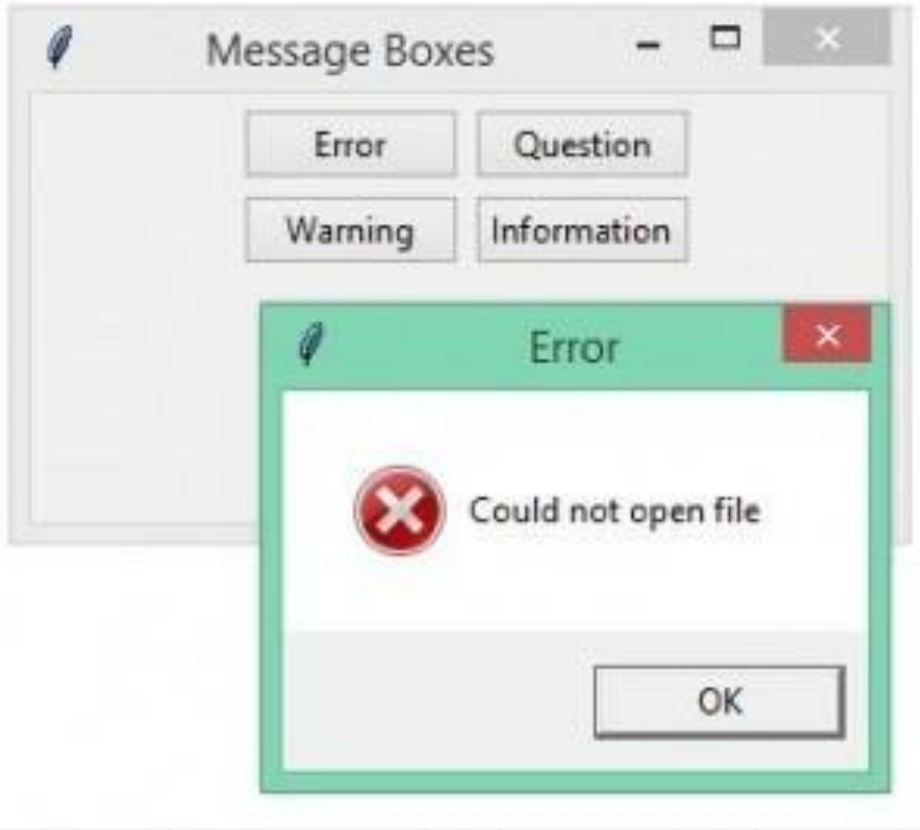
```
def initUI(self):
  self.parent.title("Message Boxes")
  self.pack()
  error = Button(self, text="Error", command=self.onError)
  error.grid(padx=5, pady=5)
  warning = Button(self, text="Warning", command=self.onWarn)
   warning.grid(row=1, column=0)
   question = Button(self, text="Question", command=self.onQuest)
   question.grid(row=0, column=1)
   inform = Button(self, text="Information", command=self.onInfo)
   inform.grid(row=1, column=1)
```

HỘP THOẠI

```
def onError(self):
        mbox.showerror("Error", "Could not open file")
    def onWarn(self):
        mbox.showwarning("Warning", "Deprecated function call")

    def onQuest(self):
        mbox.askquestion("Question", "Are you sure to quit?")

    def onInfo(self):
        mbox.showinfo("Information", "Download completed")
```

HỘP THOẠI

```
root = Tk()
ex = Example(root)
root.geometry("300x150+300+300")
root.mainloop()
```



HỘP THOẠI

• **Hộp thoại chọn màu (Color chooser):** Các hệ điều hành hay có sẵn hộp thoại chọn màu cho chúng ta sử dụng.

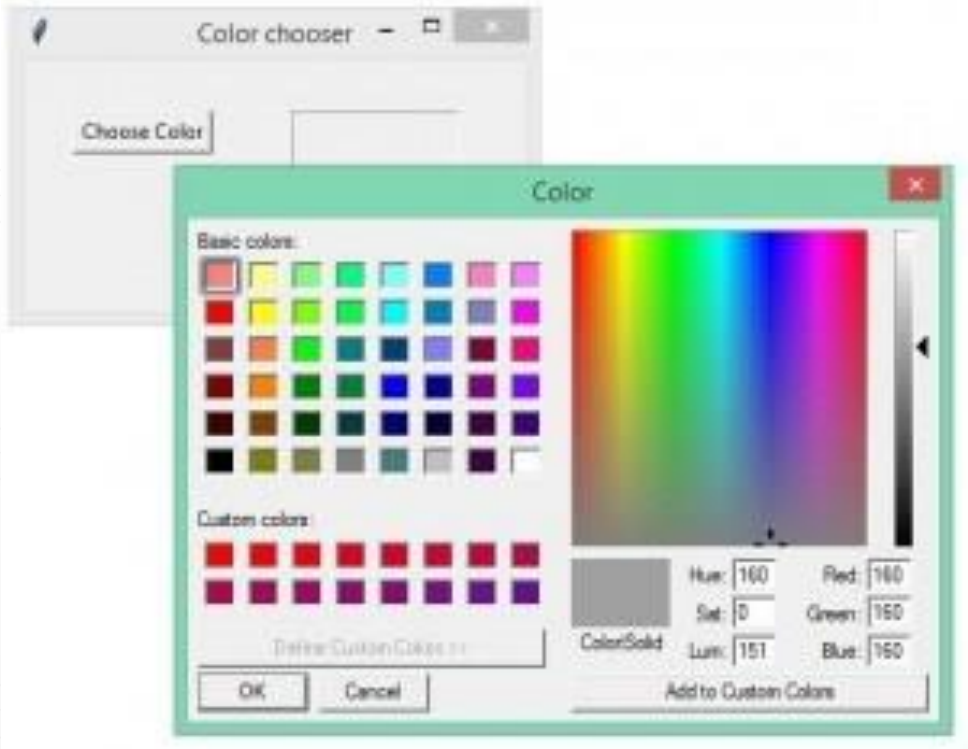
```
from tkinter import Tk, Frame, Button, BOTH, SUNKEN
from tkinter.colorchooser import askcolor


class Example(Frame):
  def __init__(self, parent):
      Frame.__init__(self, parent)
      self.parent = parent
      self.initUI()
```

HỘP THOẠI

```
  def initUI(self):
        self.parent.title("Color chooser")
        self.pack(fill=BOTH, expand=1)
        self.btn = Button(self, text="Choose Color",
command=self.onChoose)
        self.btn.place(x=30, y=30)

        self.frame = Frame(self, border=1, relief=SUNKEN, width=100,
height=100)
        self.frame.place(x=160, y=30)
```

HỘP THOẠI

```
def onChoose(self):
    (rgb, hx) = askcolor()
    self.frame.config(bg=hx)

root = Tk()
ex = Example(root)
root.geometry("300x150+300+300")
root.mainloop()
```



HỘP THOẠI

• **Hộp thoại chọn file (File Dialog):** Ví dụ: dùng hàm Open của module tkinter.filedialog để mở một File Dialog.

```
from tkinter import Frame, Tk, BOTH, Text, Menu, END
from tkinter.filedialog import Open


class Example(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()
```

HỘP THOẠI

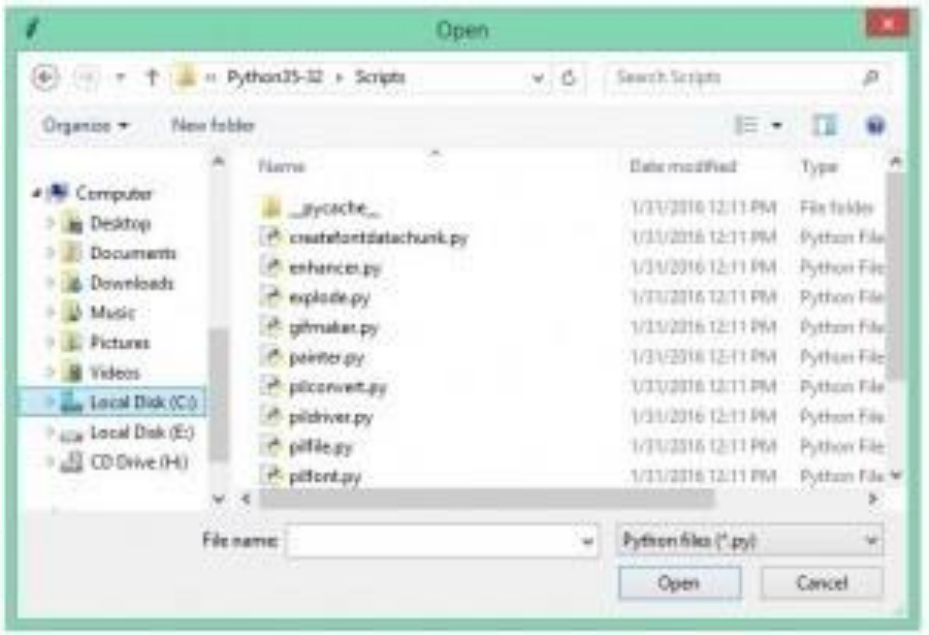
```
def initUI(self):
        self.parent.title("File dialog")
        self.pack(fill=BOTH, expand=1)
        menubar = Menu(self.parent)
        self.parent.config(menu=menubar)
        fileMenu = Menu(menubar)
        fileMenu.add_command(label="Open", command=self.onOpen)
        menubar.add_cascade(label="File", menu=fileMenu)
        self.txt = Text(self)
        self.txt.pack(fill=BOTH, expand=1)
```

HỘP THOẠI

```
def onOpen(self):
        ftypes = [('Python files', '*.py'), ('All files', '*')]
        dlg = Open(self, filetypes = ftypes)
        fl = dlg.show()
    if fl != '':
        text = self.readFile(fl)
        self.txt.insert(END, text)
    def readFile(self, filename):
      f = open(filename, "r")
      text = f.read()
      return text
```

HỘP THOẠI

```
root = Tk()
ex = Example(root)
root.geometry("300x250+300+300")
root.mainloop()
```



Chương 8:
LẬP TRÌNH GIAO DIỆN TRONG PYTHON

NỘI DUNG GIẢNG DẠY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

8.7. Bài tập

ĐỒ HỌA

- **Vẽ đoạn thẳng**:

Để vẽ đoạn thẳng thì chúng ta dùng phương thức create_line() của lớp Canvas.

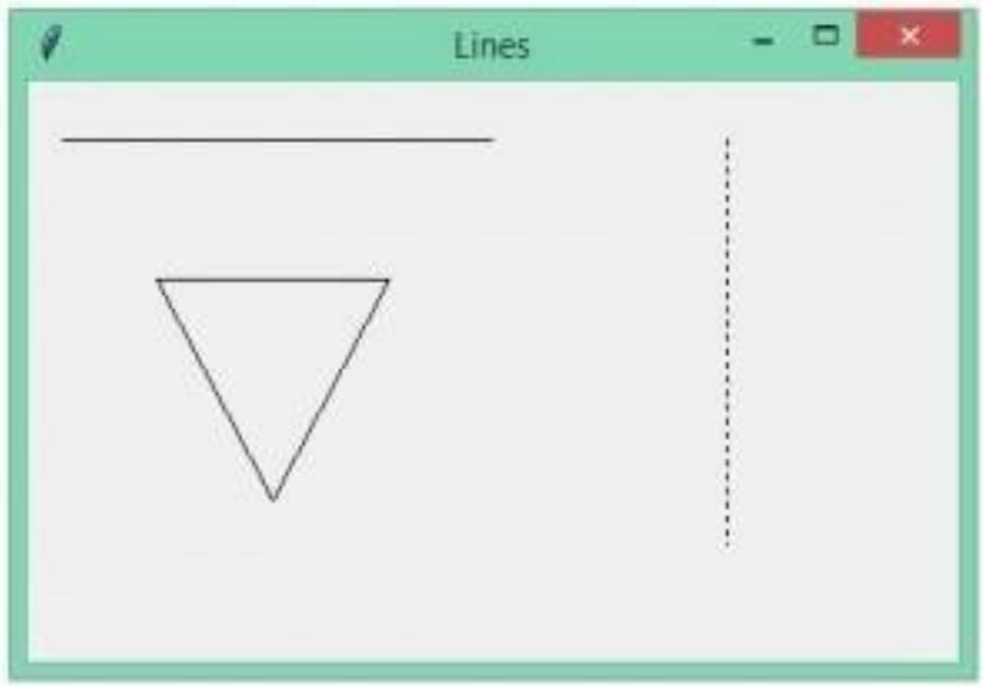
```
from tkinter import Tk, Canvas, Frame, BOTH


class Example(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()
```

ĐỒ HỌA

```
def initUI(self):
        self.parent.title("Lines")
        self.pack(fill=BOTH, expand=1)

        canvas = Canvas(self)
        canvas.create_line(15, 25, 200, 25)
        canvas.create_line(300, 25, 300, 200, dash=(4, 2))
        canvas.create_line(55, 85, 155, 85, 105, 180, 55, 85)

        canvas.pack(fill=BOTH, expand=1)
```

ĐỒ HỌA

```
root = Tk()
ex = Example(root)
root.geometry("400x250+300+300")
root.mainloop()
```



ĐỒ HỌA

- **Vẽ màu**:

Màu trong máy tính là màu RGB, là tổ hợp của 3 giá trị đỏ (Red), xanh lá (Green) và xanh lam (Blue).

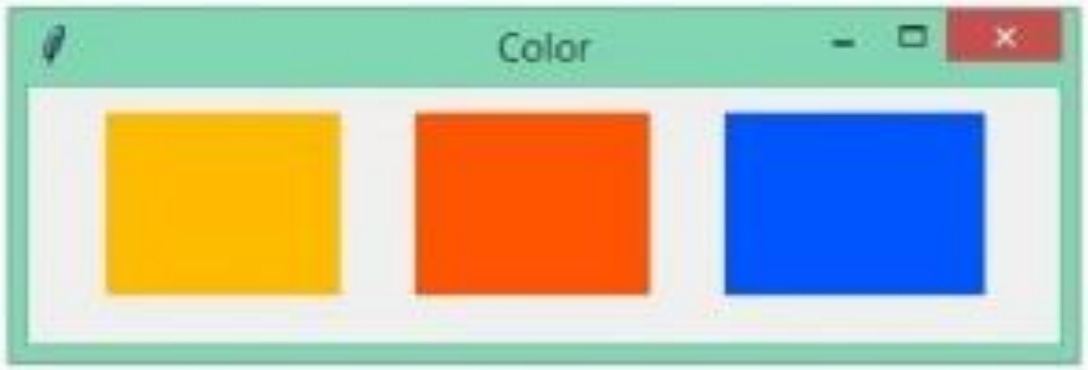
```
from tkinter import Tk, Canvas, Frame, BOTH


class Example(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()
```

ĐỒ HỌA

```
  def initUI(self):
        self.parent.title("Colors")
        self.pack(fill=BOTH, expand=1)
        canvas = Canvas(self)
        canvas.create_rectangle(30,  10,  120,  80,  outline="#fb0",
fill="#fb0")
        canvas.create_rectangle(150,  10,  240,  80,  outline="#f50",
fill="#f50")
        canvas.create_rectangle(270,  10,  370,  80,  outline="#05f",
fill="#05f")
        canvas.pack(fill=BOTH, expand=1)
```

ĐỒ HỌA

```
root = Tk()
ex = Example(root)
root.geometry("400x100+300+300")
root.mainloop()
```



ĐỒ HỌA

- **Vẽ một số đối tượng hình học khác:**

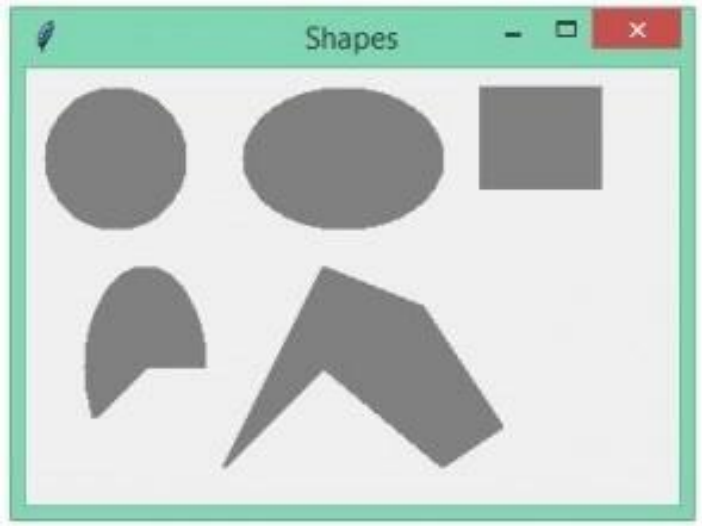
```
from tkinter import Tk, Canvas, Frame, BOTH

class Example(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()
```

ĐỒ HỌA

```
  def initUI(self):
        self.parent.title("Shapes")
        self.pack(fill=BOTH, expand=1)
        canvas = Canvas(self)
        canvas.create_oval(10, 10, 80, 80, outline="gray", fill="gray",
width=2)
        canvas.create_oval(110, 10, 210, 80, outline="gray",
fill="gray", width=2)
        canvas.create_rectangle(230, 10, 290, 60, outline="gray",
fill="gray", width=2)
```

ĐỒ HỌA

```
  canvas.create_arc(30, 200, 90, 100, start=0, extent=210,
outline="gray", fill="gray", width=2)
        points = [150, 100, 200, 120, 240, 180, 210, 200, 150, 150,
100, 200]
        canvas.create_polygon(points, outline="gray", fill="gray",
width=2)
        canvas.pack(fill=BOTH, expand=1)
  root = Tk()
  ex = Example(root)
  root.geometry("330x220+300+300")
  root.mainloop()
```



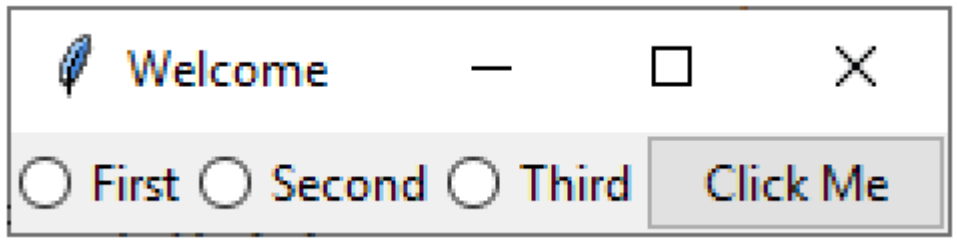
BÀI TẬP

NỘI DUNG:

1. Tạo một chương trình GUI thực hiện chức năng của một máy tính (Calculator) đơn giản.

2. Tạo một chương trình GUI thực hiện giải các phương trình bậc nhất và bậc 2 với các hệ số do người dùng nhập vào từ bàn phím. Lưu kết quả giải phương trình vào file.

3. Xây dựng chương thực hiện giao diện



TỔNG KẾT HỌC PHẦN

Sau khi học xong học phần: **Kỹ thuật lập trình** các sinh viên cần nắm được các kiến thức và kỹ năng như sau:

- Kiến thức và kỹ năng căn bản về lập trình bao gồm hai phương pháp lập trình: lập trình có cấu trúc và lập trình hướng đối tượng.
- Xây dựng các thuật toán và thực hiện lập trình bằng ngôn ngữ Pythonđể giải quyết vấn đề.
- Thể hiện phong cách lập trình chuyên nghiệp
- Kỹ năng về làm việc nhóm để viết chương trình phần mềm giải quyết các vấn đề trong thực tiễn.
- Sử dụng công cụ GITHUB để lưu trữ mã chương trình và làm việc nhóm.

CHÚC CÁC BẠN THÀNH CÔNG!