

# GUI Toolkit

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH



# GUI



# What

- GUI
  - Graphical User Interface
  - Interactive with graphical components
    - Windows, scrollbars, buttons, textboxes,
  - Sexy (?)
- CLI
  - Command Line Interface
  - Writing commands in terminal
  - Wait for response from system
  - Old school, boring (?)



# CLI vs GUI

| Feature        | CLI       | GUI             |
|----------------|-----------|-----------------|
| Learning curve | Steep     | Easy            |
| Flexibility    | Very high | Limited         |
| Memory         | Low       | Higher          |
| Speed          | Fast      | Slower          |
| Interact       | Keyboard  | Keyboard, Mouse |
| Theming        | Limited   | Easy            |

# Why

- User friendly, intuitive for new comers
- Easy to learn, no need to remember comments
- Better multitasking



# Python GUI Toolkit





# Toolkits

| Feature        | Tkinter | PyQt | Kivy   | wxPython  |
|----------------|---------|------|--------|-----------|
| Included?      | Yes     | No   | No     | No        |
| Cross platform | Yes     | Yes  | Yes    | Yes       |
| Backend        | Tcl/Tk  | Qt   | OpenGL | wxWidgets |



# Tkinter

- Simplicity
- Flexibility
- Focusing on new comers
- TODO: image of student management system here





# Tkinter

- Window
- Widgets
- Layout
- Window event loop



# Tkinter: Window

- Types: Main window and sub window
- Important attributes: title, size
- Main window

```
import tkinter as tk
window = tk.Tk()
window.title("Student Information System")
window.geometry("800x600")
```

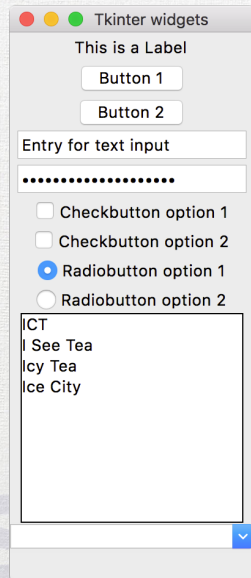
- Sub window

```
sub = tk.Toplevel(window)
sub.title("Students")
sub.geometry("600x400")
```

1. Window
2. Widgets
3. Layout
4. Window event loop

# Tkinter: Widgets

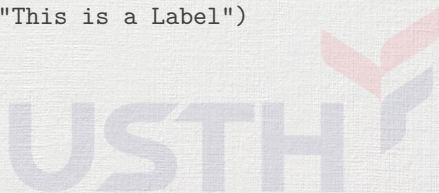
- Everything is widget
  - Frame
  - Label
  - Buttons
  - Entry
  - Check Button
  - Radio Button
  - List Box
  - ComboBox
  - Menu
  - ...
- Important attributes
  - Dimension: `width = 400, height = 300`
  - Background color: `bg = "green"`



# Tkinter: Widgets

1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

- Frame
  - A container for other widgets
  - `tk.Frame(window, width = 100, height = 100)`
- Label
  - Show texts
  - `tk.Label(window, text = "This is a Label")`



# Tkinter: Widgets

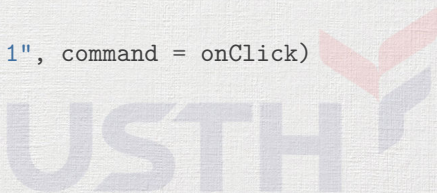
1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

- Button
  - Clickable
  - Handle click: `command = onClickFunc`

```
from tkinter import messagebox
```

```
def onClick():  
    messagebox.showinfo(message="Button 1 clicked")
```

```
tk.Button(window, text = "Button 1", command = onClick)
```





# Tkinter: Widgets

1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

- Entry

- Input texts
- Can be used for password field (with `show = "*"` )

```
entry = tk.Entry(window)
entry.insert(-1, "Entry for text input")
```

- Checkbutton

- Checkboxes
- 2 states: check and uncheck
- `tk.Checkbutton(window, text = "Checkbutton option 1")`

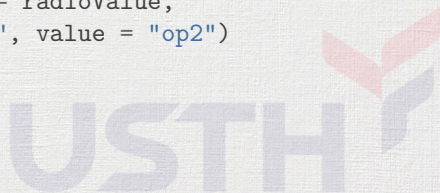


# Tkinter: Widgets

1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

- Radiobutton
  - Single choice among options
  - Text != Value

```
radioValue = tk.StringVar(value = "op1")
tk.Radiobutton(window, variable = radioValue,
    text = "Radiobutton option 1", value = "op1")
tk.Radiobutton(window, variable = radioValue,
    text = "Radiobutton option 2", value = "op2")
```



# Tkinter: Widgets

1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

- Listbox
  - A list of items
  - Selectable item
  - Get selected item: `listbox.get(tk.ACTIVE)`

```
icts = ["ICT", "I See Tea", "Icy Tea", "Ice City"]  
listbox = tk.Listbox(window)  
for i in icts:  
    listbox.insert(icts.index(i), i)
```

inserts each item at the index corresponding to its position in the icts list.

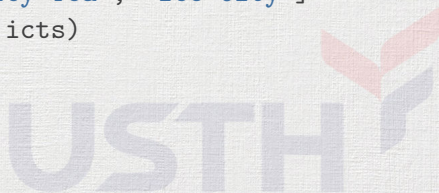


# Tkinter: Widgets

1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

- Combobox
  - A list of selectable items
  - Initially collapsed, can be expanded
  - Get selected item: `combobox.get()`

```
from tkinter import ttk
icts = ["ICT", "I See Tea", "Icy Tea", "Ice City"]
ttk.Combobox(window, values = icts)
```



# Tkinter: Layout

1. Window
2. Widgets
3. **Layout**
4. Window event loop
5. Message box

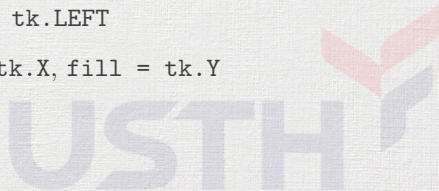
- Geometry manager
  - Handle placements (positions) of widgets on windows
  - Main container: `tk.Frame`
  - **Widget methods** for geometry management
    - `.pack()`
    - `.place()`
    - `.grid()`



# Tkinter: Layout

- `.pack()`
  - Packing algorithm
  - Similar to HTML `div`
  - Default
    - Vertically align
    - Horizontally centered
  - Alignment direction: `side = tk.LEFT`
  - Automatic expand: `fill = tk.X, fill = tk.Y`

1. Window
2. Widgets
3. **Layout**
4. Window event loop
5. Message box





# Tkinter: Layout

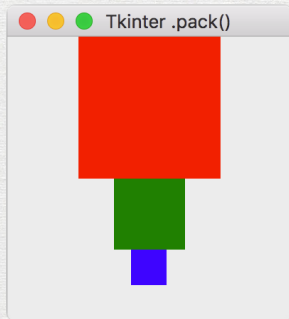
```
# default window
```

```
tk.Frame(window, width = 100, ..., bg="red").pack()  
tk.Frame(window, width = 50, ..., bg="green").pack()  
tk.Frame(window, width = 25, ..., bg="blue").pack()
```

```
# secondary window with fill
```

```
tk.Frame(sub, width = 100, ..., bg="red").pack(fill = tk.X)  
tk.Frame(sub, width = 50, ..., bg="green").pack(fill = tk.X)  
tk.Frame(sub, width = 25, ..., bg="blue").pack(fill = tk.X)
```

1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box





# Tkinter: Layout

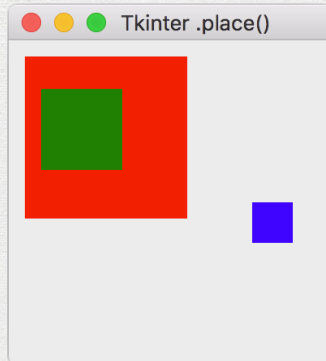
- `.place()`
  - Similar to HTML position: absolute
  - Unit: pixels
  - Absolute values
    - Position: `x = 10, y = 10`
    - Dimension: `width = 400, height = 300`
  - Relative values `[0...1]`
    - Position: `relx = 0.1, rely = 0.1`
    - Dimension: `relwidth = 0.5, relheight = 0.7`

1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

# Tkinter: Layout

```
tk.Frame(window, bg="red").place(  
    x = 10, y = 10, width = 100, height = 100)  
tk.Frame(window, bg="green").place(  
    x = 20, y = 30, width = 50, height = 50)  
tk.Frame(window, bg="blue").place(  
    x = 150, y = 100, width = 25, height = 25)
```

Output:



1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

# Tkinter: Layout

- `.grid()`
  - Similar to HTML table
  - Position: `column = 0, row = 2`
  - Stretching: `sticky = tk.EW`
  - Padding: `padx = 3, pady = 3`
  - Spanning
    - `columnspan = 3`
    - `rowspan = 2`

1. Window
2. Widgets
3. **Layout**
4. Window event loop
5. Message box



# Tkinter: Layout

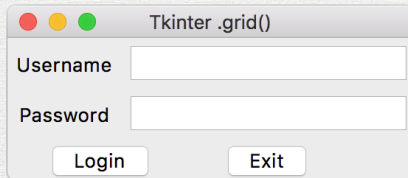
1. Window
2. Widgets
3. Layout
4. Window event loop
5. Message box

```
tk.Label(window, text = "Username").grid(  
    column = 0, row = 0, sticky = tk.EW, padx = 3, pady = 3)  
tk.Label(window, text = "Password").grid(  
    column = 0, row = 1, sticky = tk.EW, padx = 3, pady = 3)  
  
tk.Entry(window).grid(  
    column = 1, row = 0, sticky = tk.EW, padx = 3, pady = 3, columnspan = 4)  
tk.Entry(window).grid(  
    column = 1, row = 1, sticky = tk.EW, padx = 3, pady = 3, columnspan = 4)  
  
tk.Button(window, text = "Login").grid(  
    column = 0, row = 2, columnspan = 2)  
tk.Button(window, text = "Exit").grid(  
    column = 2, row = 2, columnspan = 2)
```



# Tkinter: Layout

1. Window
2. Widgets
3. **Layout**
4. Window event loop
5. Message box





# Tkinter: Window loop

- A **blocking** method
- Handles input, output events
- `window.mainloop()`





Practice!



## Practical work 9: GUI'ed management system

- Copy your pw8 directory to pw9 directory
- Upgrade your user interface to GUI using Tkinter
- Push your work to corresponding forked Github repository

