

Introduction to Python GUI

CS311: Computer Programming II

ผู้ช่วยศาสตราจารย์สิรินธร จิยาัคกดี

AGENDA

1

Python Built-in Functions

2

Python Built-in Modules

3

Python Tkinter Module

4

Python GUI Basic Example



BUILT-IN FUNCTIONS

BUILT-IN FUNCTIONS

- THE PYTHON INTERPRETER HAS A NUMBER OF FUNCTIONS AND TYPES BUILT INTO IT THAT ARE ALWAYS AVAILABLE.

Built-in Functions				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	



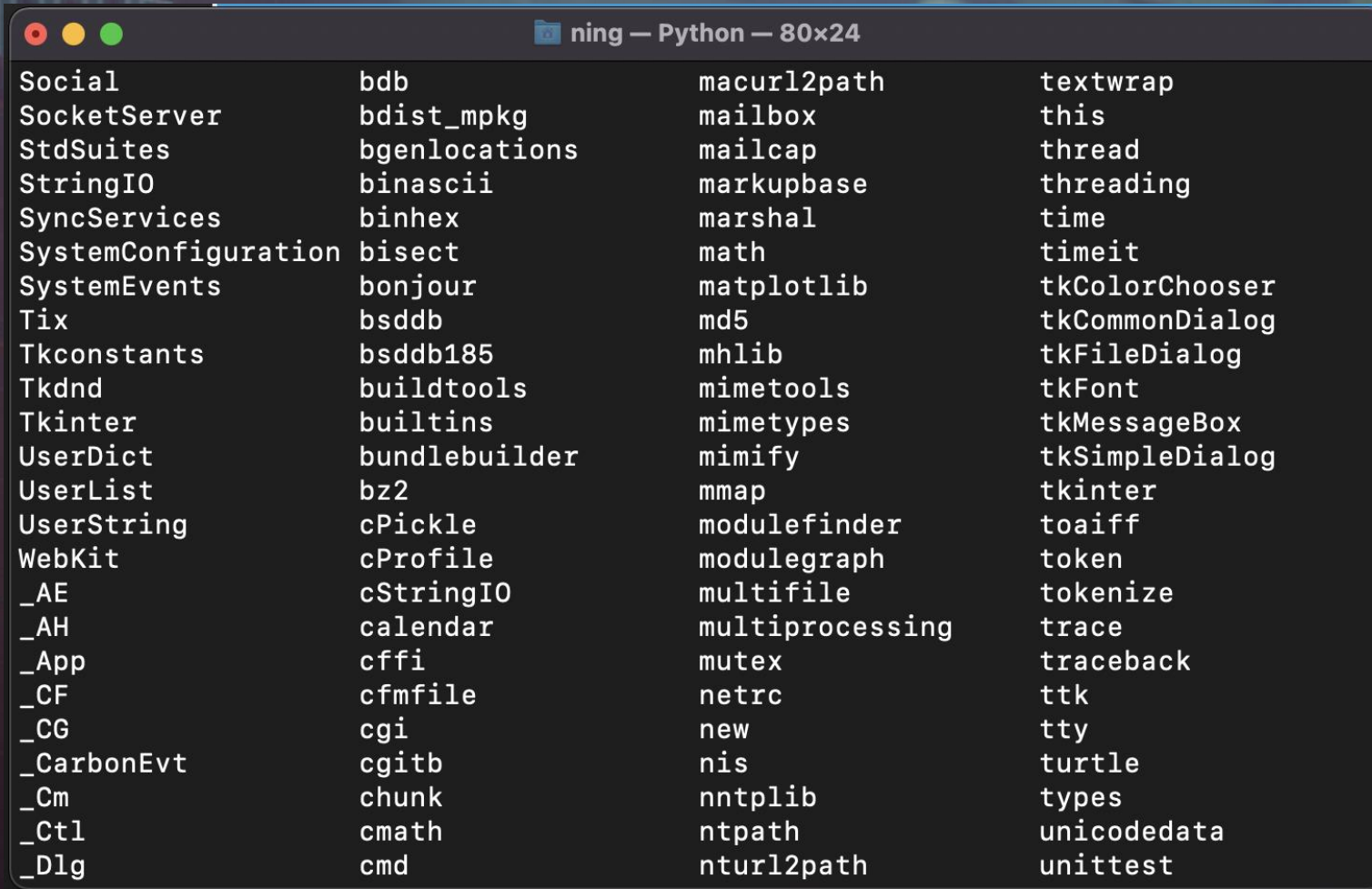
2

BUILT-IN MODULES

MODULES

- MODULES ARE BASICALLY FILES WHICH CONTAIN DEFINITIONS THAT **PROVIDE HELPFUL FUNCTIONALITY**. THEORETICALLY, ALL OF YOU COULD WRITE ANY PROGRAM YOU WANTED TO WRITE ON YOUR OWN, WITHOUT ANY MODULES OR OUTSIDE HELP. BUT WHY REINVENT THE WHEEL IF YOU DON'T HAVE TO? THIS IS THE GENERAL GIST BEHIND MODULES. YOU ALL HAVE EXPERIENCE USING MODULES SUCH AS **TKINTER AND CSV**. THESE MODULES ARE PART OF A LIBRARY OF STANDARD MODULES THAT CAME PACKAGED WITH YOUR INSTALL WHICH THE DEVELOPERS HAVE DEEMED USEFUL OR NECESSARY FOR YOUR DAILY PYTHON USAGE.

BUILT-IN MODULES NAME IN PYTHON

A terminal window titled 'ning — Python — 80x24' displays a list of Python built-in modules. The modules are organized into four columns. The first column lists modules like Social, SocketServer, StdSuites, StringIO, SyncServices, SystemConfiguration, SystemEvents, Tix, Tkconstants, Tkndnd, Tkinter, UserDict, UserList, UserString, WebKit, and various _ modules. The second column lists modules like bdb, bdist_mpkg, bgenlocations, binascii, binhex, bisect, bonjour, bsddb, bsddb185, buildtools, builtins, bundlebuilder, bz2, cPickle, cProfile, cStringIO, calendar, cffi, cfmfile, cgi, cgitb, chunk, cmath, and cmd. The third column lists modules like macurl2path, mailbox, mailcap, markupbase, marshal, math, matplotlib, md5, mhlib, mimetools, mimetypes, mimify, mmap, modulefinder, modulegraph, multifile, multiprocessing, mutex, netrc, new, nis, nntplib, ntpath, and nturl2path. The fourth column lists modules like textwrap, this, thread, threading, time, timeit, tkColorChooser, tkCommonDialog, tkFileDialog, tkFont, tkMessageBox, tkSimpleDialog, tkinter, toaiff, token, tokenize, trace, traceback, ttk, tty, turtle, types, unicodedata, and unittest.

```
ning — Python — 80x24
Social      bdb         macurl2path textwrap
SocketServer bdist_mpkg  mailbox     this
StdSuites   bgenlocations mailcap     thread
StringIO    binascii    markupbase  threading
SyncServices binhex      marshal     time
SystemConfiguration bisect     math        timeit
SystemEvents bonjour     matplotlib  tkColorChooser
Tix          bsddb       md5          tkCommonDialog
Tkconstants bsddb185    mhlib        tkFileDialog
Tkndnd       buildtools  mimetools    tkFont
Tkinter      builtins    mimetypes    tkMessageBox
UserDict     bundlebuilder mimify        tkSimpleDialog
UserList     bz2         mmap          tkinter
UserString   cPickle     modulefinder toaiff
WebKit       cProfile    modulegraph  token
_AE          cStringIO   multifile    tokenize
_AH          calendar   multiprocessing trace
_App         cffi        mutex         traceback
_CF          cfmfile     netrc         ttk
_CG          cgi          new           tty
_CarbonEvt  cgitb       nis           turtle
_Cm          chunk        nntplib       types
_Ctl         cmath        ntpath        unicodedata
_Dlg         cmd          nturl2path    unittest
```


IMPORT MODULE IN PYTHON

- THREE DIFFERENT WAYS TO **IMPORT** MODULES IN PYTHON

1

```
import module_name
```

2

```
import module_name as new_name
```

3

```
from module_name import *
```


EXAMPLE1 : CALLING FUNCTION FROM OTHER FILE USING IMPORT

example1.py

```
import module1  
print("----- Get started -----")  
module1.display()  
module1.displayname()  
print("-"*30)  
name = module1.getname()  
print("Hi ",name)  
print("-"*30)
```

module1.py

```
def display():  
    print("Hello World")  
def displayname():  
    print("Sirinthorn Cheyasak")  
def getname():  
    name = input("Enter name : ")  
    return(name)
```

EXAMPLE2 : CALLING FUNCTION FROM OTHER FILE USING `IMPORT .. AS ..`

example2.py

```
import module1 as me  
print("----- Get started -----")  
me.display()  
me.displayname()  
print("-"*30)  
name = me.getname()  
print("Hi ",name)  
print("-"*30)
```

module1.py

```
def display():  
    print("Hello World")  
def displayname():  
    print("Sirinthorn Cheyasak")  
def getname():  
    name = input("Enter name : ")  
    return(name)
```

EXAMPLE3 : CALLING FUNCTION FROM OTHER FILE USING FROM .. IMPORT..

example3.py

```
from module1 import *  
print("----- Get started -----")  
display()  
displayname()  
print("-"*30)  
name = getname()  
print("Hi ",name)  
print("-"*30)
```

module1.py

```
def display():  
    print("Hello World")  
def displayname():  
    print("Sirinthorn Cheyasak")  
def getname():  
    name = input("Enter name : ")  
    return(name)
```

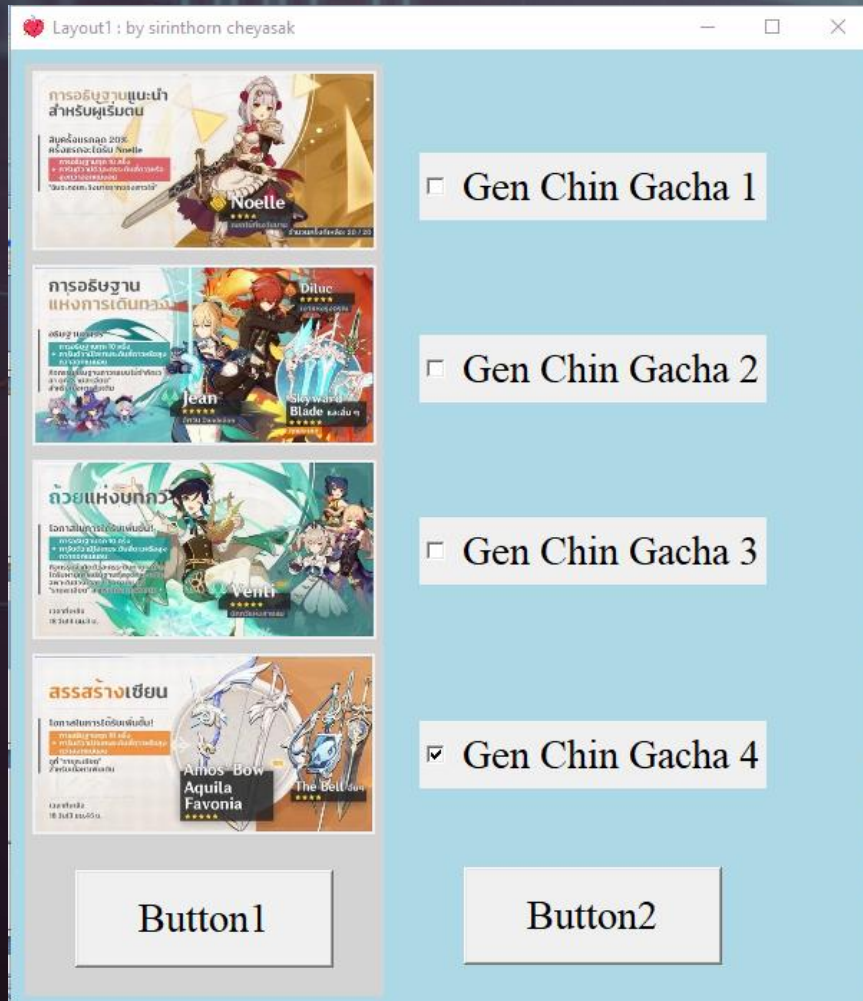



PYTHON GUI MODULE

GRAPHICAL USER INTERFACE (GUI)

- GUI is a **program interface** that takes advantage of the **computer's graphics** capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex **command languages**. On the other hand, many user find that they work more effectively with a command-driven interface, especially if they already know the command language.

GRAPHICAL USER INTERFACE (GUI)



Menu by Sirinthorn Cheyasak

1 : Select Gen Chin Gacha 1

2 : Select Gen Chin Gacha 2

3 : Select Gen Chin Gacha 3

4 : Select Gen Chin Gacha 4

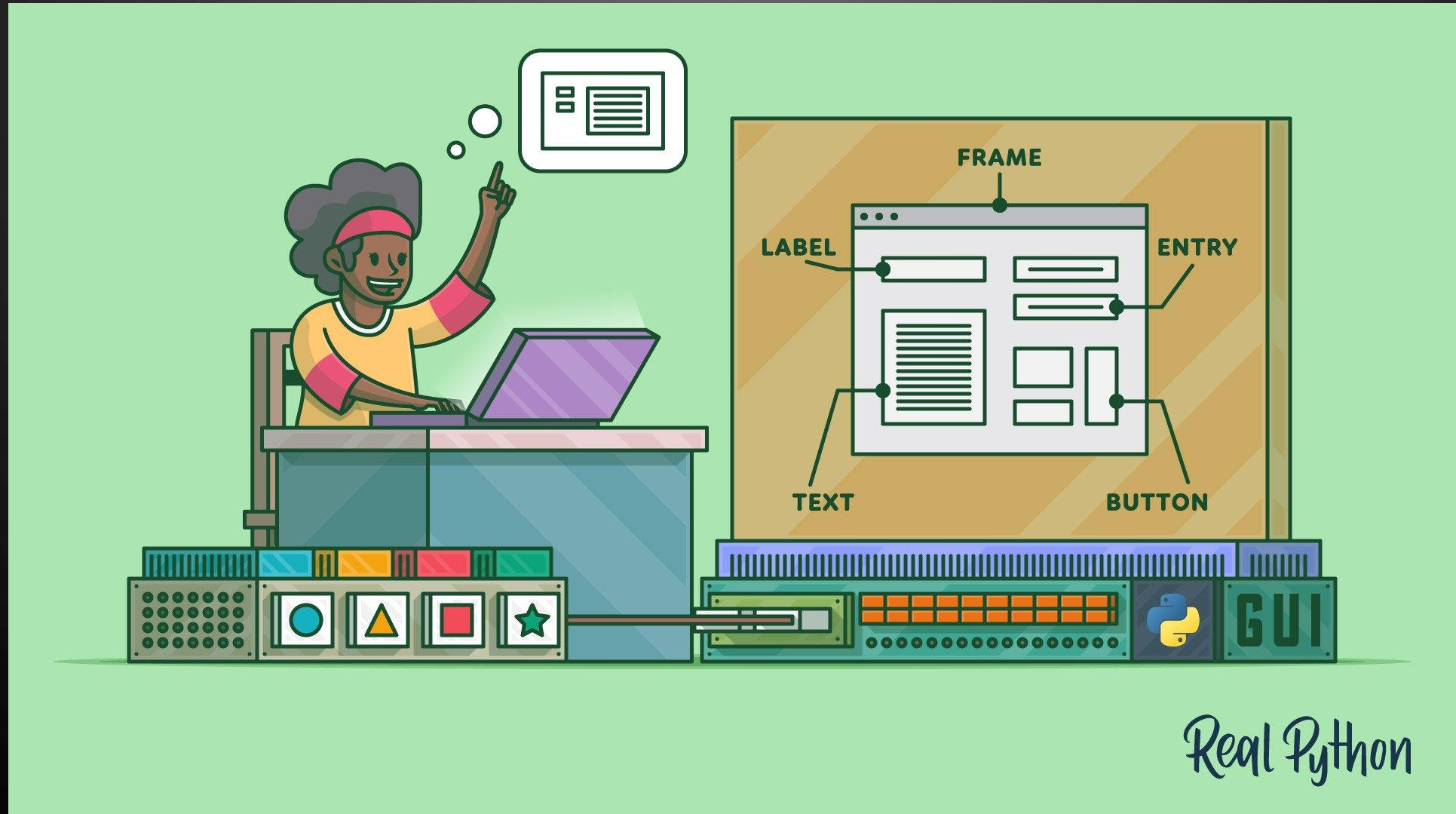
Enter menu(1-4) : █

GUI vs non-GUI

GRAPHICAL USER INTERFACE (GUI)

- Python has a huge number of GUI frameworks available for it, from **Tkinter** (traditionally bundled with Python, using **tk**) to a number of other cross-platform solutions, as well as bindings to platform-specific (also known as "native") technologies.

PYTHON GUI MODULE



Real Python

PYTHON GUI USING TKINTER

- Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
- Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –
 1. **Import** the **Tkinter** module.
 2. **Create** the GUI application **main window**.
 3. **Add one or more** of the above-mentioned **widgets** to the GUI application.
 4. Enter the main **event** loop to **take action** against each event triggered by the user.



1

WAYS TO IMPORT TKINTER MODULE

- **IMPORT** THE **TKINTER** MODULE.
- MAIN WINDOW IS A CONTAINER WHICH CONTAIN MANY OF GUI WIDGETS

1

```
import tkinter
```

2

```
import tkinter as tk
```

3

```
from tkinter import *
```



CREATING TKINTER MAIN WINDOW

Create the GUI application **main window**.

Main Window

widget

widget

widget

widget



CREATING TKINTER MAIN WINDOW

- **TK():** TO CREATE A MAIN WINDOW, TKINTER OFFERS A METHOD TK(). THE BASIC CODE USED TO CREATE THE MAIN WINDOW OF THE APPLICATION IS:

```
top = Tk()
```

- **MAINLOOP():** THERE IS A METHOD KNOWN BY THE NAME MAINLOOP() IS USED WHEN YOUR APPLICATION IS READY TO RUN. MAINLOOP() IS AN INFINITE LOOP USED TO RUN THE APPLICATION, WAIT FOR AN EVENT TO OCCUR AND PROCESS THE EVENT AS LONG AS THE WINDOW IS NOT CLOSED.

```
top.mainloop()
```




EX.

CREATE MAIN WINDOW EXAMPLE

2

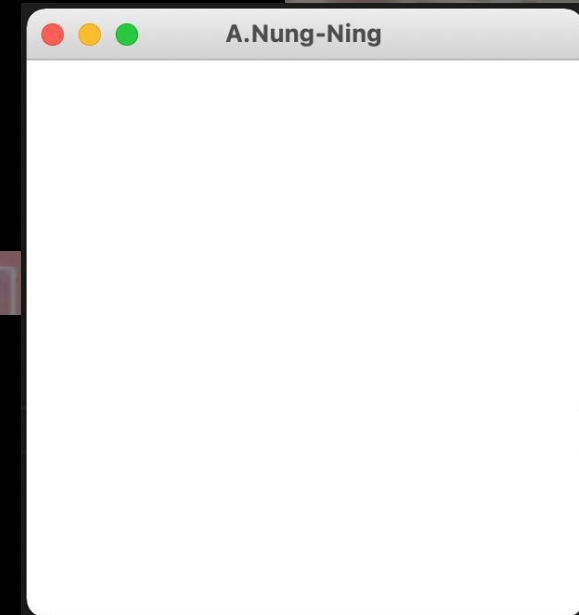
```
import tkinter as tk
top = tk.Tk()
top.title("A.Nung-Ning")
top.wm_geometry("300x300")

top.mainloop()
```

3

```
from tkinter import *
top = Tk()
top.title("A.Nung-Ning")
top.wm_geometry("300x300")

top.mainloop()
```



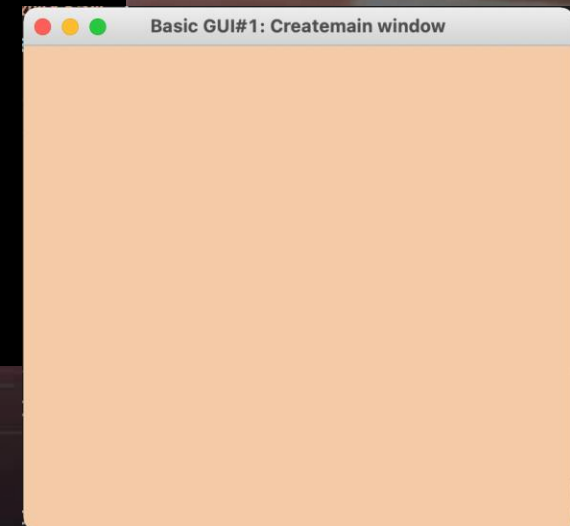


EX.

SETTING MAIN WINDOW PROPERTIES

- **title()** method sets the window **title**.
- **geometry()** method sets size of the window
- **configure()** method configures the Window

```
from tkinter import *  
top = Tk()  
top.title("Basic GUI#1: Createmain window")  
top.wm_geometry('500x350')  
top.configure(bg="#F5CBA7")  
top.mainloop()
```





CREATING TKINTER WIDGETS

- **Tkinter** provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.
- **Syntax :**

```
objName = tkname.widget(parent, option=value, ...)
```

objName	คือ ชื่อออบเจ็กต์ที่ต้องการสร้าง
tkname	คือ คลาสโมดูล tkinter
widget	คือ widget ที่ต้องการสร้างบน window/frame
parent	คือ main window หรือ widget ที่ต้องการให้เป็น parent
option=value	คือ คุณสมบัติที่ต้องการกำหนดให้กับ widget



Common widgets

LIST OF COMMON WIDGETS

- Common widgets are:
 - **Label** สำหรับแสดงข้อความ
 - **Entry** สำหรับรับข้อมูล (Textbox)
 - **Button** ปุ่มสำหรับให้ผู้ใช้แสดงความต้องการ
 - **Text** สำหรับแสดงข้อความใดหลายบรรทัด
 - **Image** สำหรับแสดงรูปภาพ
 - **Radio Button** แสดงทางเลือกให้ผู้ใช้เลือกได้ 1 ทางเลือก
 - **Checkbox** แสดงทางเลือกให้ผู้ใช้เลือกได้หลายทางเลือก

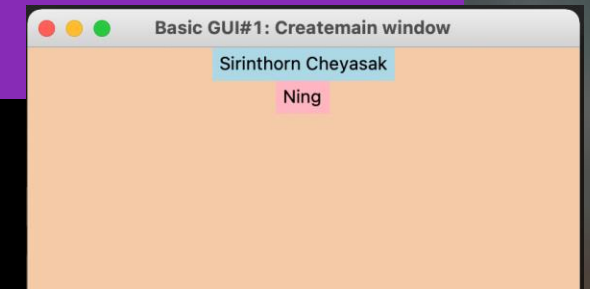
COMMON WIDGETS : LABEL

■ Syntax :

```
widgetname = Label(parent, attribute=value,...)
```

```
from tkinter import *
```

```
top = Tk()
top.title("Basic GUI#1: Createmain window")
top.wm_geometry('400x350')
top.configure(background="#F5CBA7")
name = Label(top,text="Sirinthorn
             Cheyasak",anchor="center",bg='lightblue')
nickname = Label(text="Ning",bg='lightpink')
name.pack()
nickname.pack()
top.mainloop()
```



LABEL ATTRIBUTE AND VALUES

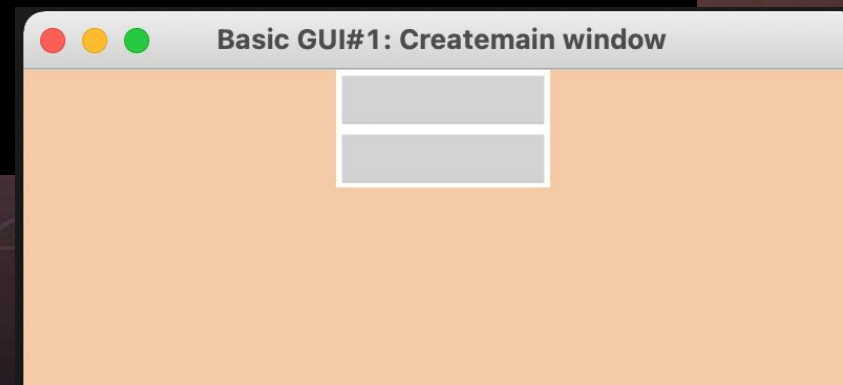
Attribute/Option	Description
bg	The normal background color displayed behind the label and indicator.
fg	Normal foreground (text) color.
font	Font option specifies in what font that text will be displayed.
width	Width of the label.
height	Height of the label.
text	To display one or more lines of text in a label widget, set this option to a string containing the text. Internal newlines (" <code>\n</code> ") will force a line break.
padx	Extra space added to the left and right of the text within the widget. Default is 1.
pady	Extra space added above and below the text within the widget. Default is 1.

COMMON WIDGETS : ENTRY

- Syntax :

```
widgetname = Entry(parent, attribute=value,...)
```

```
from tkinter import *  
  
top = Tk()  
top.title("Basic GUI#1: Createmain window")  
top.geometry('400x350')  
top.configure(background="#F5CBA7")  
box1 = Entry(width=10,bg='lightgrey')  
box2 = Entry(width=10,bg='lightgrey')  
box1.pack()  
box2.pack()  
top.mainloop()
```



ENTRY ATTRIBUTE AND VALUES

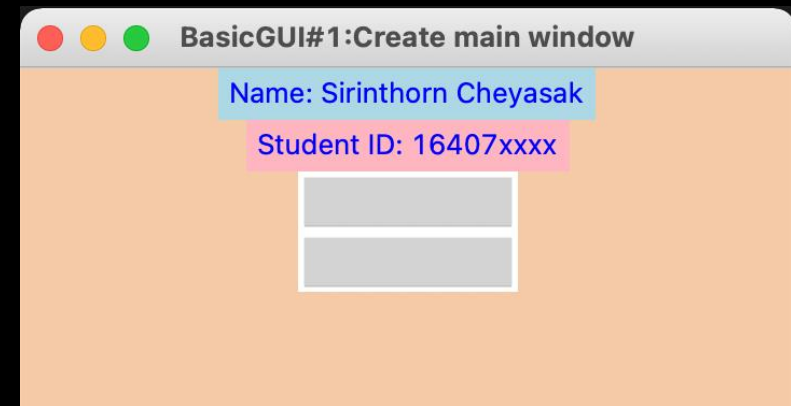
Attribute/Option	Description
bg	Normal background color.
fg	Normal foreground (text) color.
state	The default is state=NORMAL, but you can use state=DISABLED to gray out the control and make it unresponsive. If the cursor is currently over the checkbutton, the state is ACTIVE.
show	Normally, the characters that the user types appear in the entry. To make a .password. entry that echoes each character as an asterisk, set show="*".
bd	The size of the border around the indicator. Default is 2 pixels.
command	Function or method to be called every time the user changes the state of this checkbutton.

PYTHON GUI EXAMPLE1

```
from tkinter import *
def createwindow():
    root = Tk()
    root.title("BasicGUI#1:Create main window")
    root.wm_geometry('360x250')
    root.configure(bg='#F5CBA7')
    return root

def createwidget(root):
    name = Label(root,text="Name: Sirinthorn
                  Cheyasak",fg="blue",bg="lightblue")
    id = Label(root,text="Student ID: 16407xxxx",fg="blue",bg="lightpink")
    name.pack()
    id.pack()
    box1 = Entry(width=10,bg='lightgrey')
    box2 = Entry(width=10,bg='lightgrey')
    box1.pack()
    box2.pack()

root = createwindow()
createwidget(root)
root.mainloop()
```



COMMON WIDGETS : BUTTON

- Syntax :

```
widgetname = Button(parent, attribute=value,...)
```

```
from tkinter import *  
  
top = Tk()  
top.title("Basic GUI#1: Createmain window")  
top.wm_geometry('400x350')  
top.configure(background="#F5CBA7")  
button1 = Button(top,text="Click Me1")  
button2 = Button(top,text="Click Me2",  
                 width=10,highlightbackground='pink')  
button1.pack()  
button2.pack()  
top.mainloop()
```



BUTTON ATTRIBUTE AND VALUES

Attribute/Option	Description
bg	Normal background color.
fg	Normal foreground (text) color.
width	Width of the button in letters (if displaying text) or pixels (if displaying an image).
height	Height of the button in text lines (for textual buttons) or pixels (for images).
text	Button title
image	Image to be displayed on the button (instead of text).
compound	Set a position of image on the button
command	Function or method to be called when the button is clicked.



Geometry Management

GEOMETRY MANAGEMENT

- All Tkinter widgets have access to the specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: **pack**, **grid**, and **place**.
 - The **pack()** Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.
 - The **grid()** Method – This geometry manager organizes widgets in a table-like structure in the parent widget.
 - The **place()** Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.



Geometry management using pack method

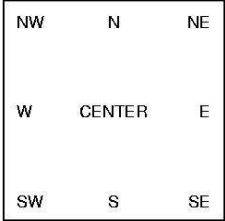
GEOMETRY MANAGEMENT: **PACK()**

- THIS GEOMETRY MANAGER ORGANIZES WIDGETS IN BLOCKS BEFORE PLACING THEM IN THE PARENT WIDGET.
- **SYNTAX:**

```
widgetVar.pack(option=value, ...)
```

widgetVar	คือ ตัวแปร widget ที่ต้องการจัดตำแหน่ง
pack()	คือ method สำหรับจัดตำแหน่ง
option=value	คือ คุณสมบัติการจัดตำแหน่ง widget

LIST OF OPTIONS/ATTRIBUTES : **PACK()**

Option name	Meaning	Using
anchor	<p>กำหนดตำแหน่ง widget ภายใน frame ได้แก่</p> <ul style="list-style-type: none"> - N (กลางด้านบน), S (กลางด้านล่าง) - E (กลางด้านขวา), W (กลางด้านซ้าย) - NE (มุมบนขวา), NW(มุมบนซ้าย) - SE (มุมล่างขวา), SW (มุมล่างซ้าย) - ค่าเริ่มต้น (default) จะอยู่ตรงกลาง (Center) ของเซลล์ 	<pre>w.pack(anchor="w") w.pack(anchor=se)</pre>
fill	ขยายขนาด widget ตามแนวแกน x และ y หรือกำหนดให้เป็น both เพื่อขยายทั้ง 2 ด้าน	<pre>w.pack(fill="x") w.pack(fill=y)</pre>
padx	กำหนดระยะห่างจากขอบภายนอกของ widget แนวอนแกน x	<pre>w.pack(padx=5)</pre>
pady	กำหนดระยะห่างจากขอบภายนอกของ widget แนวตั้งแกน y	<pre>w.pack(pady=5)</pre>
ipadx	กำหนดระยะห่างจากขอบภายในของ widget แนวอนแกน x	<pre>w.pack(ipadx=5)</pre>
ipady	กำหนดระยะห่างจากขอบภายในของ widget แนวตั้งแกน y	<pre>w.pack(ipady=5)</pre>
side	กำหนดตำแหน่งการวาง widget เป็น left, right หรือ bottom ซึ่ง default คือ top	<pre>w.grid(side=left)</pre>



Geometry management using grid method

GEOMETRY MANAGEMENT: **GRID()**

- THIS GEOMETRY MANAGER ORGANIZES WIDGETS IN A TABLE-LIKE STRUCTURE IN THE PARENT WIDGET.
- **SYNTAX:**

```
widgetVar.grid(option=value, ...)
```

widgetVar	คือ ตัวแปร widget ที่ต้องการจัดตำแหน่ง
grid()	คือ method สำหรับจัดตำแหน่ง
option=value	คือ คุณสมบัติการจัดตำแหน่ง widget

GEOMETRY MANAGEMENT: GRID()

row=0 column=0	row=0 column=1	row=0 column=2
row=1 column=0	row=1 column=1	row=1 column=2
row=2 column=0	row=2 column=1	row=2 column=2

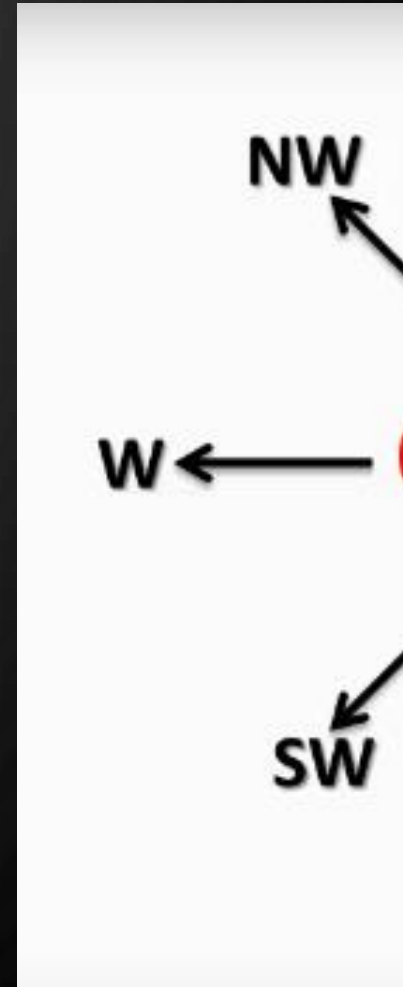
GEOMETRY MANAGEMENT: GRID()

```
from tkinter import *
top = Tk()
top.title("Using grid() by A.Ning")
lb1 = Label(top,width=20,height=3,bg="#b39bc8")
lb2 = Label(top,width=20,height=3,bg="#7fe6e8")
lb3 = Label(top,width=20,height=3,bg="#a4d3ff")
lb4 = Label(top,width=20,height=3,bg="#60f4b4")
lb5 = Label(top,width=20,height=3,bg="#fff68f")
lb6 = Label(top,width=20,height=3,bg="#ff7518")
lb7 = Label(top,width=20,height=3,bg="#a52008")
lb8 = Label(top,width=20,height=3,bg="#CCFF44")
lb9 = Label(top,width=20,height=3,bg="#ffd1dc")
lb1.grid(row=0,column=0)
lb2.grid(row=0,column=1)
lb3.grid(row=0,column=2)
lb4.grid(row=1,column=0)
lb5.grid(row=1,column=1)
lb6.grid(row=1,column=2)
lb7.grid(row=2,column=0)
lb8.grid(row=2,column=1)
lb9.grid(row=2,column=2)
top.mainloop()
```

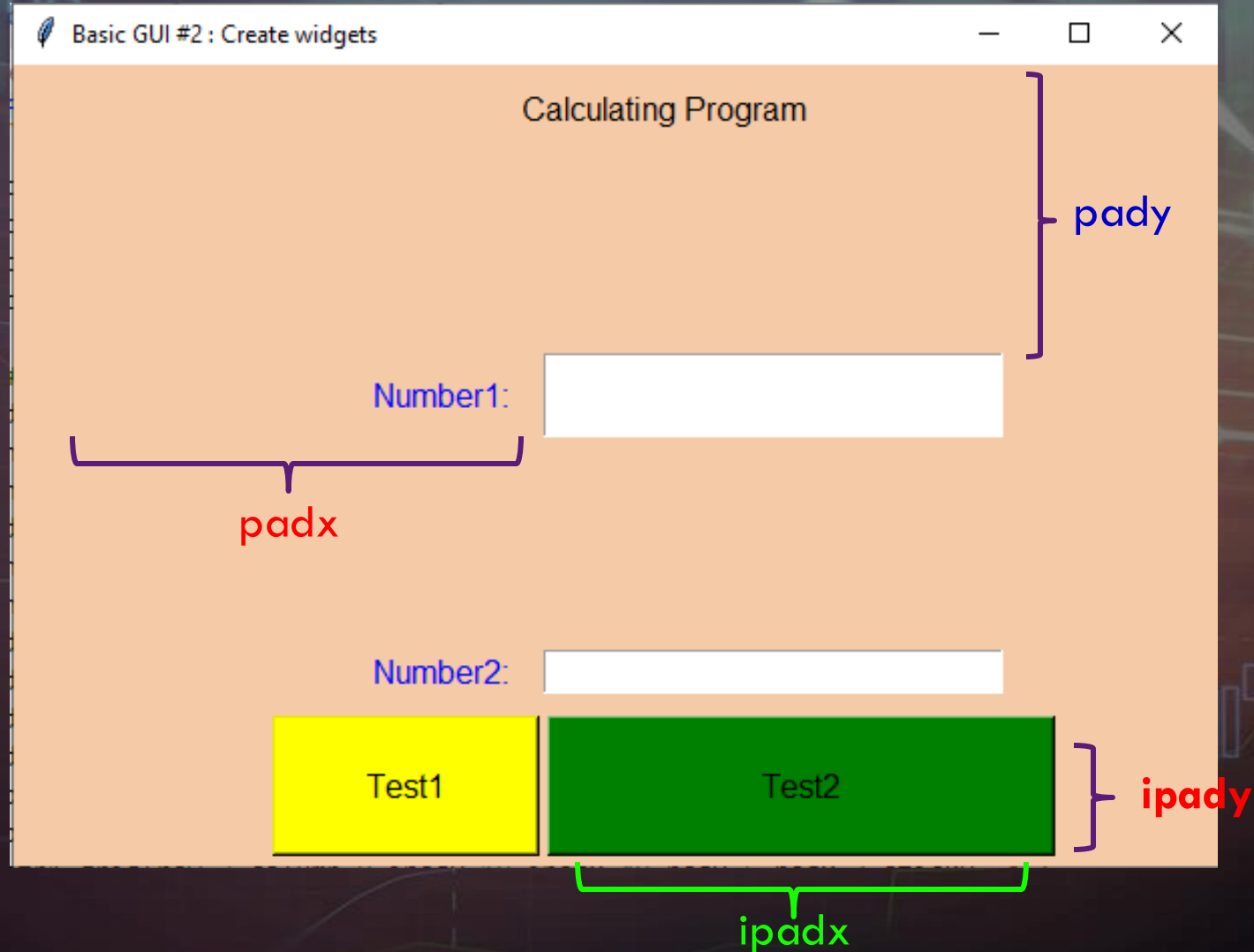


LIST OF OPTIONS/ATTRIBUTES : **GRID()**

Option name	Meaning	Using
column	กำหนด column ที่ต้องการวาง widget ค่าเริ่มต้นคือ 0	w.grid(column=1)
columnspan	กำหนดจำนวน column ที่ต้องการรวมเข้าด้วยกัน	w.grid(columnspan=2)
row	กำหนด row ที่ต้องการวาง widget ค่าเริ่มต้นคือ 0	w.grid(row=1)
rowspan	กำหนดจำนวน row ที่ต้องการรวมเข้าด้วยกัน	w.grid(rowspan=2)
padx	กำหนดระยะห่างจากขอบภายนอกของ widget แนวนอน	w.grid(padx=5)
pady	กำหนดระยะห่างจากขอบภายนอกของ widget แนวตั้ง	w.grid(pady=5)
ipadx	กำหนดระยะห่างจากขอบภายในของ widget แนวนอน	w.grid(ipadx=5)
ipady	กำหนดระยะห่างจากขอบภายในของ widget แนวตั้ง	w.grid(ipady=5)
sticky	ค่าที่ระบุการวางตำแหน่ง widget ภายใน cell ได้แก่ <ul style="list-style-type: none"> - N (กลางด้านบน), S (กลางด้านล่าง) - E (กลางด้านขวา), W (กลางด้านซ้าย) - NE (มุมบนขวา), NW (มุมบนซ้าย) - SE (มุมล่างขวา), SW (มุมล่างซ้าย) - ค่าเริ่มต้น (default) จะอยู่ตรงกลาง (Center) ของเซลล์ 	w.grid(sticky="w") w.grid(sticky="se")



GEOMETRY MANAGEMENT EXAMPLE



COLUMN, ROW AND COLUMNSPAN

The diagram illustrates a table with 3 rows and 6 columns. The first row is the header, and the subsequent two rows contain data. The first column is labeled 'Life Expectancy' and the second column is labeled 'Current Age'. The first row has a 'Colspan=2' label above the first two columns and another 'Colspan=2' label above the last two columns. The second row has a 'Colspan=2' label above the first two columns and another 'Colspan=2' label above the last two columns. The third row has a 'Colspan=2' label above the first two columns and another 'Colspan=2' label above the last two columns. The table is annotated with green arrows on the left pointing to 'row0', 'row1', and 'row2', and blue arrows at the bottom pointing to 'Col 0' through 'Col 5'. The data values are: Row 0: 65, 40, 20; Row 1: Men, Women, Men, Women, Men, Women; Row 2: 82, 85, 78, 82, 77, 81.

Life Expectancy		Current Age			
65	40	20			
Men	Women	Men	Women	Men	Women
82	85	78	82	77	81

COLUMNSPAN EXAMPLE

Invoice			
Item / Desc.	Qty.	@	Price
Paperclips (Box)	100	1.15	115.00
Paper (Case)	10	45.99	459.90
Wastepaper Baskets	2	17.99	35.98
Subtotal			610.88
Tax		7%	42.76
Total			653.64

Colspan=3

Colspan=3

ROWSPAN EXAMPLE

Favorite and Least Favorite Things			
row0		Bob	Alice
row1	Favorite	Blue	Purple
		Banana	Chocolate
row2	Least Favorite	Yellow	Pink
		Mint	Walnut

Col 0 Col 1 Col 2 Col 3

Rowspan=2

Rowspan=2



Geometry management using place method

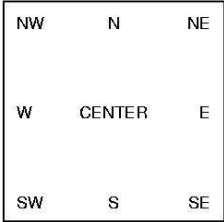
GEOMETRY MANAGEMENT: **PLACE()**

- THIS GEOMETRY MANAGER ORGANIZES WIDGETS BY PLACING THEM IN A SPECIFIC POSITION IN THE PARENT WIDGET.
- **SYNTAX:**

```
widgetVar.place(option=value, ...)
```

widgetVar	คือ ตัวแปร widget ที่ต้องการจัดตำแหน่ง
place()	คือ method สำหรับจัดตำแหน่ง
option=value	คือ คุณสมบัติการจัดตำแหน่ง widget

LIST OF OPTIONS/ATTRIBUTES: **PLACE()**

Option name	Meaning	Using
anchor	<p>กำหนดตำแหน่ง widget ภายใน frame ได้แก่</p> <ul style="list-style-type: none"> - N (กลางด้านบน), S (กลางด้านล่าง) - E (กลางด้านขวา), W (กลางด้านซ้าย) - NE (มุมบนขวา), NW(มุมบนซ้าย) - SE (มุมล่างขวา), SW (มุมล่างซ้าย) - ค่าเริ่มต้น (default) จะอยู่ตรงกลาง (Center) ของเซลล์ 	<p><code>w.place(anchor="w")</code> <code>w.place(anchor=se)</code></p>
height	กำหนดความสูงของ widget ที่จะวางลงบน parent	<code>w.place(height=50)</code>
width	กำหนดความกว้างของ widget ที่จะวางลงบน parent	<code>w.place(width=150)</code>
x	กำหนดตำแหน่งแกน x ของ parent ที่จะวาง widget	<code>w.place(x=150)</code>
y	กำหนดตำแหน่งแกน y ของ parent ที่จะวาง widget	<code>w.place(x=150, y=150)</code>
relheight	กำหนดความสูงของ widget ที่จะวางลงบน parent	<code>w.pack(ipady=5)</code>
relwidth	กำหนดความกว้างของ widget ที่จะวางลงบน parent	<code>w.grid(side=left)</code>



Class Activity of Week2



Assignment of Week2



See you next week