Python GUI Programming (Tkinter)

- Python provides various options for developing **graphical user interfaces (GUIs)**.
- Most important are listed below:
  1. **Tkinter:** Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
  2. **wxPython:** This is an open-source Python interface for wxWindows http://wxpython.org.
  3. **JPython:** JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

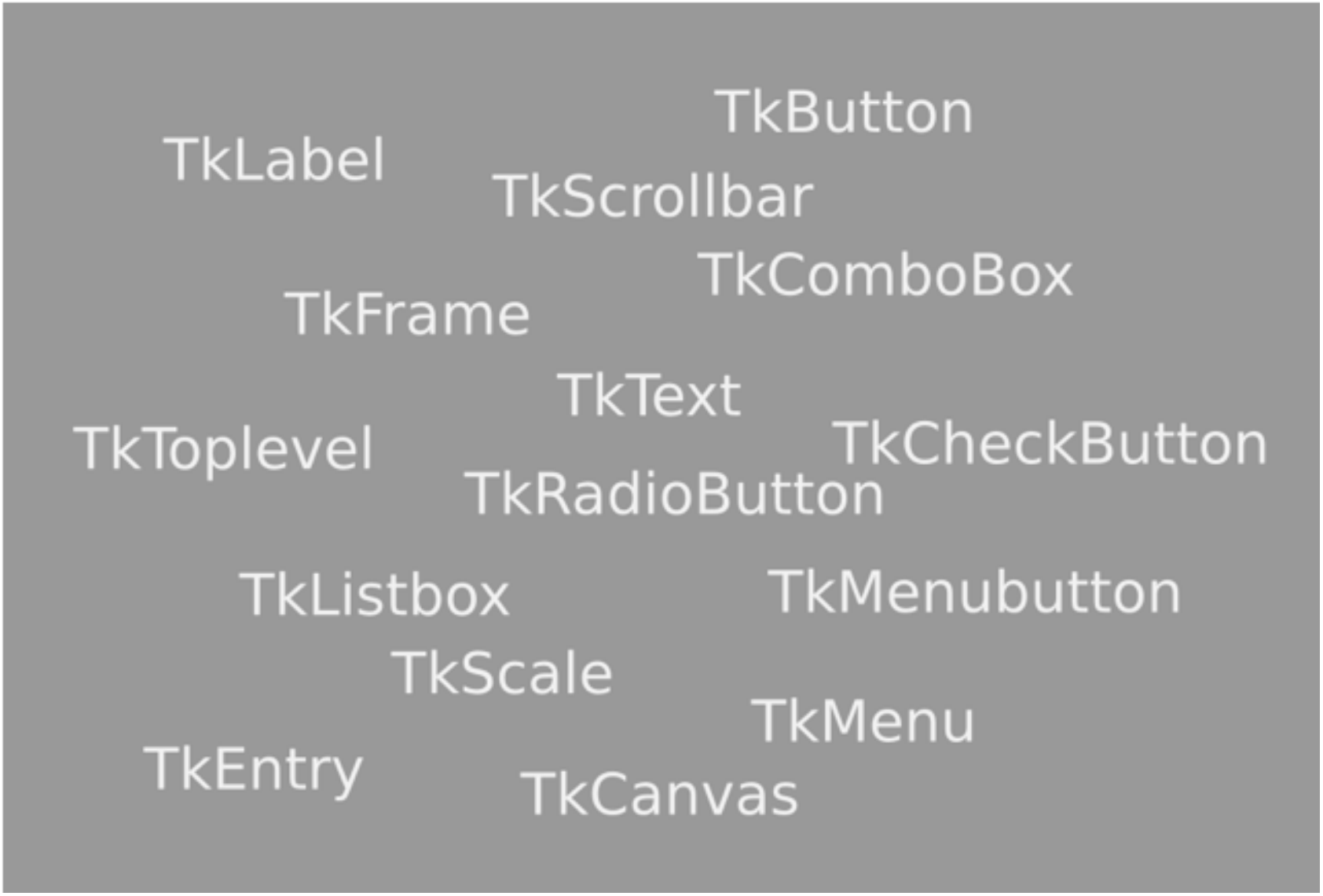
Tkinter Programming

- 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.
- 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.

Example

```
#!/usr/bin/python

import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

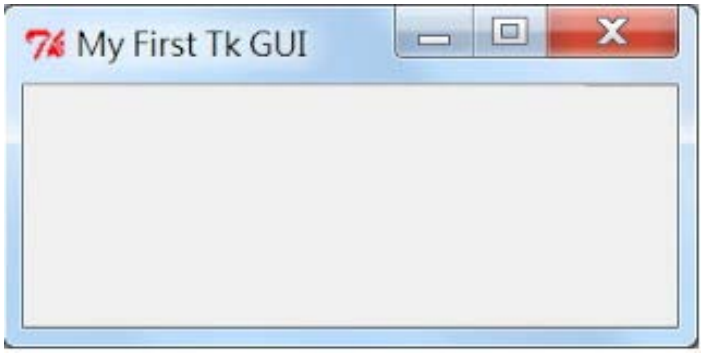
Tkinter Components



Operator	Description
Button	The Button widget is used to display buttons in your application.
Canvas	The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
Checkbutton	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
Entry	The Entry widget is used to display a single-line text field for accepting values from a user.
Frame	The Frame widget is used as a container widget to organize other widgets.
Label	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
Listbox	The Listbox widget is used to provide a list of options to a user.
Menubutton	The Menubutton widget is used to display menus in your application.
Menu	The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
Message	The Message widget is used to display multiline text fields for accepting values from a user.
Radiobutton	The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
Scale	The Scale widget is used to provide a slider widget.
Scrollbar	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
Text	The Text widget is used to display text in multiple lines.
Toplevel	The Toplevel widget is used to provide a separate window container.
Spinbox	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
PanedWindow	A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
LabelFrame	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
tkMessageBox	This module is used to display message boxes in your applications.

Example

```
import Tkinter as tk
win=tk.Tk()
win.title("My First Tk GUI")
win.mainloop()
```

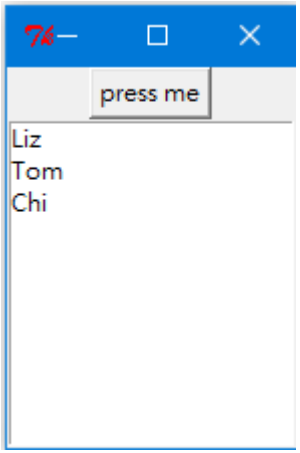



```
import Tkinter as tk
win=tk.Tk()
win.title("My First Tk GUI")
win.resizable(0,0)
win.mainloop()
```

Example

```
import Tkinter as tk
win=tk.Tk()     #建立視窗容器物件
win.title("Tk GUI")
label=tk.Label(win, text="Hello World!")   #
建立標籤物件
label.pack()       #顯示元件
button=tk.Button(win, text="OK")
button.pack()     #顯示元件
win.mainloop()
```

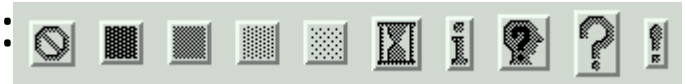

```
from Tkinter import *          #This interface allow us to draw windows
def DrawList():
        plist = ['Liz','Tom','Chi']
        for item in plist:
                listbox.insert(END,item);

win = Tk()                     #This creates a window, but it won't show up
listbox = Listbox(win)
button = Button(win, text = "press me", command = DrawList)
button.pack()
listbox.pack()                 #this tells the listbox to come out
win.mainloop()                  #This command will tell the window come out
```



Standard Attributes

- Let's take a look at how some of their common attributes, such as sizes, colors and fonts are specified.
  - Dimensions
  - Colors
  - Fonts
  - Anchors : The Tkinter module defines a number of *anchor* constants that you can use to control where items are positioned relative to their context.
  - Relief styles : The *relief style* of a widget refers to certain simulated 3-D effects around the outside of the widget.

  - Bitmaps :



  - Cursors

Geometry Management

Pack方法提供了選項來布局元件在介面中的位置，選項有：side、expand、fill、等。
Grid方法是採用行列來確定元件在介面中的位置，row是行號，column是列號。
Place方法是通過元件在介面中的橫縱坐標來固定位置。

- All Tkinter widgets have access to 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.

Example

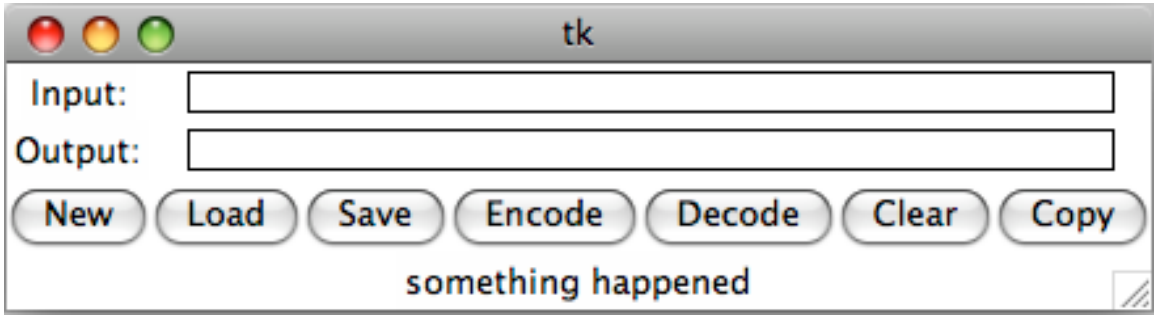
```
from Tkinter import *
win = Tk()
frame = Frame(win)
frame.pack()
bottomframe = Frame(win)
bottomframe.pack( side = BOTTOM )
redbutton = Button(frame, text="Red", fg="red")
redbutton.pack( side = LEFT)
brownbutton = Button(frame, text="Brown", fg="brown")
brownbutton.pack( side = LEFT )
```

Example

```
bluebutton = Button(frame, text="Blue", fg="blue")
bluebutton.pack( side = LEFT )
blackbutton = Button(bottomframe, text="Black", fg="black")
blackbutton.pack( side = BOTTOM)
win.mainloop()
```

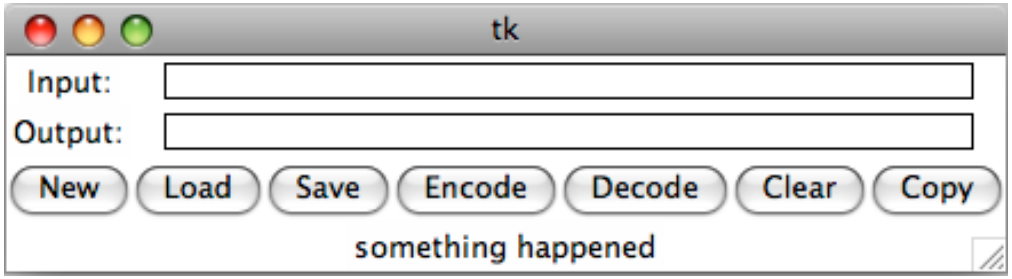
Example

- Import Tkinter
- class GUIDemo(Frame): # (inherit) Tkinter Frame
- def __init__(self, master=None):
- Frame.__init__(self, master)
- self.grid()
- self.createWidgets()



- def createWidgets(self):
- # input
- self.inputText = Label(self)
- self.inputText["text"] = "Input:"
- self.inputText.grid(row=0, column=0)
- self.inputField = Entry(self)
- self.inputField["width"] = 50
- self.inputField.grid(row=0, column=1, columnspan=6)
- #output
- self.outputText = Label(self)
- self.outputText["text"] = "Output:"
- self.outputText.grid(row=1, column=0)
- self.outputField = Entry(self)
- self.outputField["width"] = 50
- self.outputField.grid(row=1, column=1, columnspan=6)

```
        self.new = Button(self)
        self.new["text"] = "New"
        self.new.grid(row=2, column=0)

        self.load = Button(self)
        self.load["text"] = "Load"
        self.load.grid(row=2, column=1)

        self.save = Button(self)
        self.save["text"] = "Save"
        self.save.grid(row=2, column=2)

        self.encode = Button(self)
        self.encode["text"] = "Encode"
        self.encode.grid(row=2, column=3)

        self.decode = Button(self)
        self.decode["text"] = "Decode"
        self.decode.grid(row=2, column=4)
```



- self.clear = Button(self)
- self.clear["text"] = "Clear"
- self.clear.grid(row=2, column=5)

- self.copy = Button(self)
- self.copy["text"] = "Copy"
- self.copy.grid(row=2, column=6)

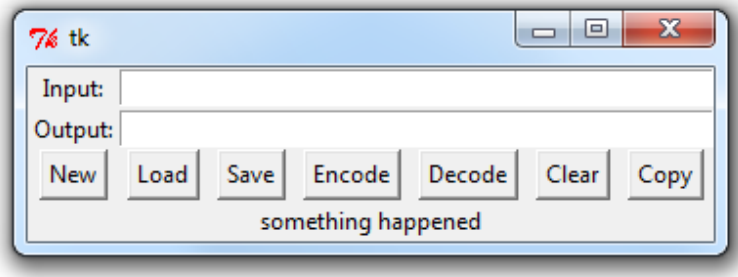
- self.displayText = Label(self)
- self.displayText["text"] = "something happened"
- self.displayText.grid(row=3, column=0, columnspan=7)

- if __name__ == '__main__':
- root = Tk()
- app = GUIDemo(master=root)
- app.mainloop()

GUI1.py - C:/Python27/GUI1.py

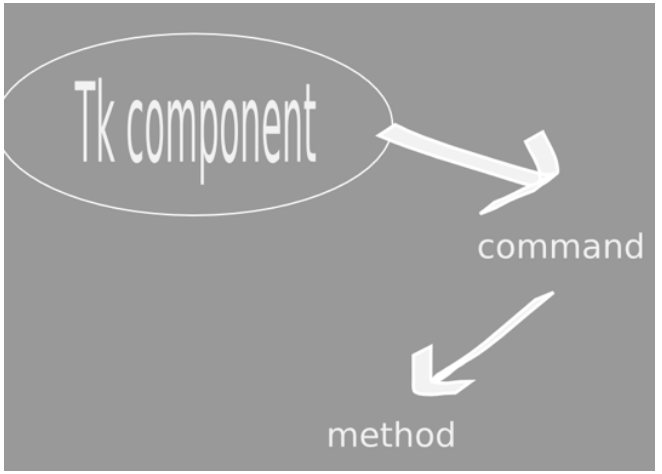
File Edit Format Run Options Windows Help

```
from Tkinter import *

class GUIDemo(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.grid()
        self.createWidgets()

    def createWidgets(self):
        self.inputText = Label(self)
        self.inputText["text"] = "Input:"
        self.inputText.grid(row=0, column=0)
        self.inputField = Entry(self)
        self.inputField["width"] = 50
        self.inputField.grid(row=0, column=1, columnspan=6)

        self.outputText = Label(self)
        self.outputText["text"] = "Output:"
        self.outputText.grid(row=1, column=0)
        self.outputField = Entry(self)
        self.outputField["width"] = 50
        self.outputField.grid(row=1, column=1, columnspan=6)

        self.new = Button(self)
        self.new["text"] = "New"
        self.new.grid(row=2, column=0)
        self.load = Button(self)
        self.load["text"] = "Load"
        self.load.grid(row=2, column=1)
        self.save = Button(self)
        self.save["text"] = "Save"
        self.save.grid(row=2, column=2)
        self.encode = Button(self)
        self.encode["text"] = "Encode"
        self.encode.grid(row=2, column=3)
        self.decode = Button(self)
        self.decode["text"] = "Decode"
        self.decode.grid(row=2, column=4)
        self.clear = Button(self)
        self.clear["text"] = "Clear"
        self.clear.grid(row=2, column=5)
```



Ln: 1 Col: 6

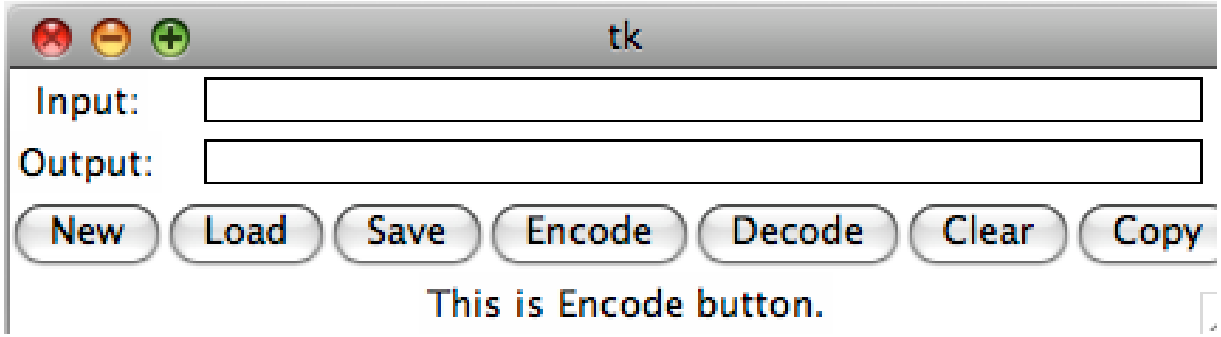
Command



- self.new["command"] = self.newMethod

- def newMethod(self):

  self.displayText["text"] = "This is New button.”



- Add commands to New, Load, Save, Encode, Decode, Clear, and Copy

Command

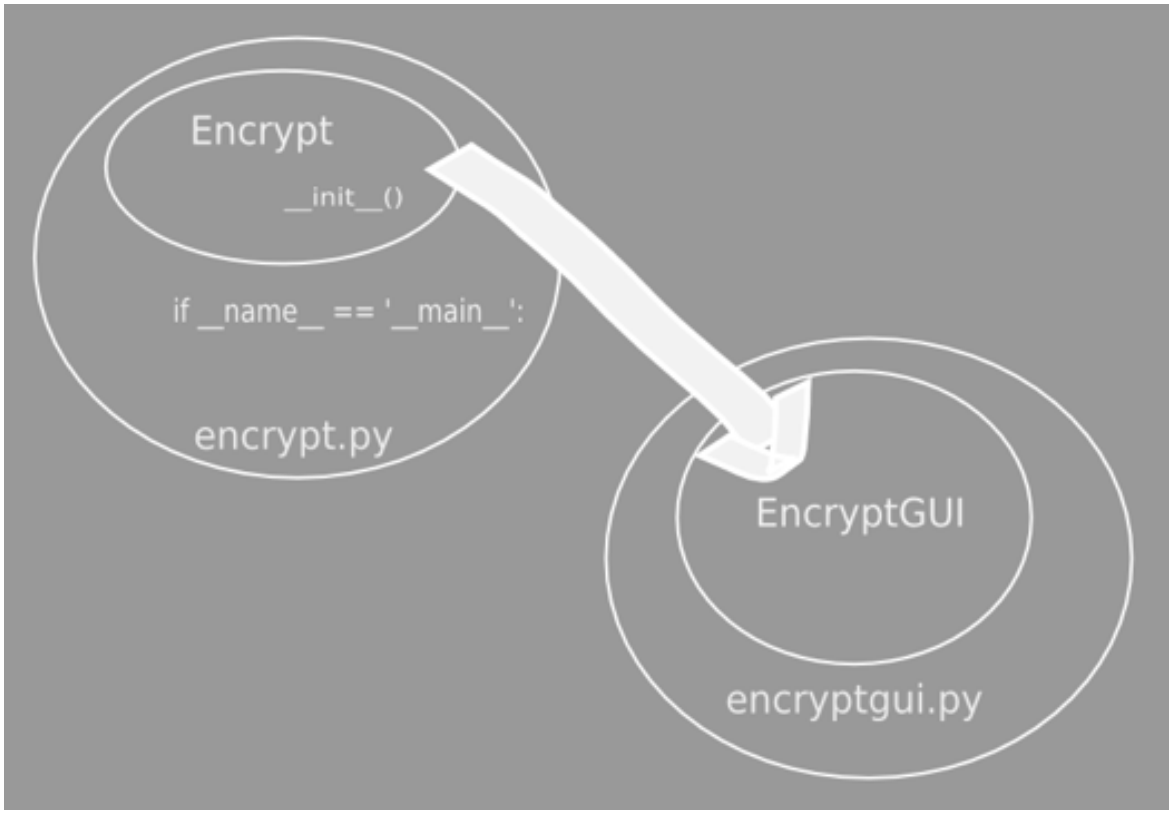
- self.new = Button(self)
- self.new["text"] = "New"
- self.new.grid(row=2, column=0)
- self.new["command"] = self.newMethod
- self.load = Button(self)
- self.load["text"] = "Load"
- self.load.grid(row=2, column=1)
- self.load["command"] = self.loadMethod
- self.save = Button(self)
- self.save["text"] = "Save"
- self.save.grid(row=2, column=2)
- self.save["command"] = self.saveMethod

```
self.encode = Button(self)
self.encode["text"] = "Encode"
self.encode.grid(row=2, column=3)
self.encode["command"] =  self.encodeMethod
self.decode = Button(self)
self.decode["text"] = "Decode"
self.decode.grid(row=2, column=4)
self.decode["command"] =  self.decodeMethod
self.clear = Button(self)
self.clear["text"] = "Clear"
self.clear.grid(row=2, column=5)
self.clear["command"] =  self.clearMethod
self.copy = Button(self)
self.copy["text"] = "Copy"
self.copy.grid(row=2, column=6)
self.copy["command"] =  self.copyMethod
self.displayText = Label(self)
self.displayText["text"] = "something happened"
self.displayText.grid(row=3, column=0, columnspan=7)
```

```
def newMethod(self):
    self.displayText["text"] = "This is New button."
def loadMethod(self):
    self.displayText["text"] = "This is Load button."
def saveMethod(self):
    self.displayText["text"] = "This is Save button."
def encodeMethod(self):
    self.displayText["text"] = "This is Encode button."
def decodeMethod(self):
    self.displayText["text"] = "This is Decode button."
def clearMethod(self):
    self.displayText["text"] = "This is Clear button."
def copyMethod(self):
    self.displayText["text"] = "This is Copy button."
```

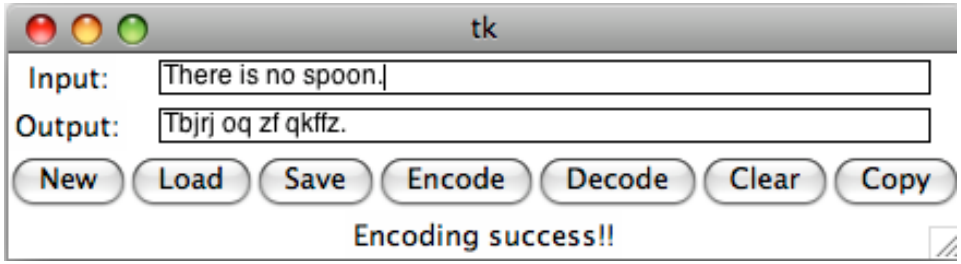
Encrypt

- Import Tkinter
- import Encrypt

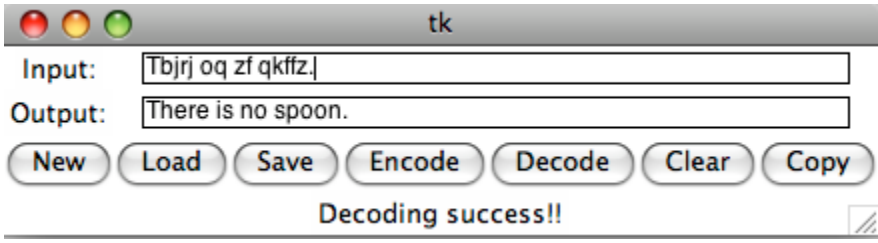


encodeMethod

```
def encodeMethod(self):
    self.userinput = self.inputField.get()

    if self.userinput == "":
        self.displayText["text"] = "No input string!!"
    else:
        if self.e == None:
            self.displayText["text"] = "No encrypt object!!"
        else:
            self.result = self.e.toEncode(self.userinput)
            self.outputField.delete(0, 200)
            self.outputField.insert(0, self.result)
            self.displayText["text"] = "Encoding success!!"
```



decodeMethod




```
def decodeMethod(self):
    self.userinput = self.inputField.get()

    if self.userinput == "":
        self.displayText["text"] = "No input string!!"
    else:
        if self.e == None:
            self.displayText["text"] = "No encrypt object!!"
        else:
            self.result = self.e.toDecode(self.userinput)
            self.outputField.delete(0, 200)
            self.outputField.insert(0, self.result)
            self.displayText["text"] = "Decoding success!!"
```

Save

```
def saveMethod(self):
    if self.e == None:
        self.displayText["text"] = "No Encrypt object can save!!"
    else:
        f = open('./code.txt', 'w')
        f.write(self.e.getCode())
        f.closed
        self.displayText["text"] = "The code is saved."
```

Load

```
def loadMethod(self):
    if os.path.exists("./code.txt"):
        f = open('./code.txt', 'r')
        code = f.readline()
        self.e = Encrypt()
        self.e.setCode(code)
        self.displayText["text"] = "code: " + self.e.getCode()
    else:
        self.displayText["text"] = "Load denied!!"
```

Clear

```
def clearMethod(self):
    self.e = None
    self.userinput = ""
    self.result = ""
    self.inputField.delete(0, 200)
    self.outputField.delete(0, 200)
    self.displayText["text"] = "It's done."
```

Copy

```
def copyMethod(self):
    if self.result == "":
        self.displayText["text"] = "Copy denied!!"
    else:
        self.clipboard_clear()
        self.clipboard_append(self.result)
        self.displayText["text"] = "It is already copied to the
clipboard."
```

New

```
def newMethod(self):
        self.e = Encrypt()
        self.displayText["text"] = self.e
```

encrypt.py

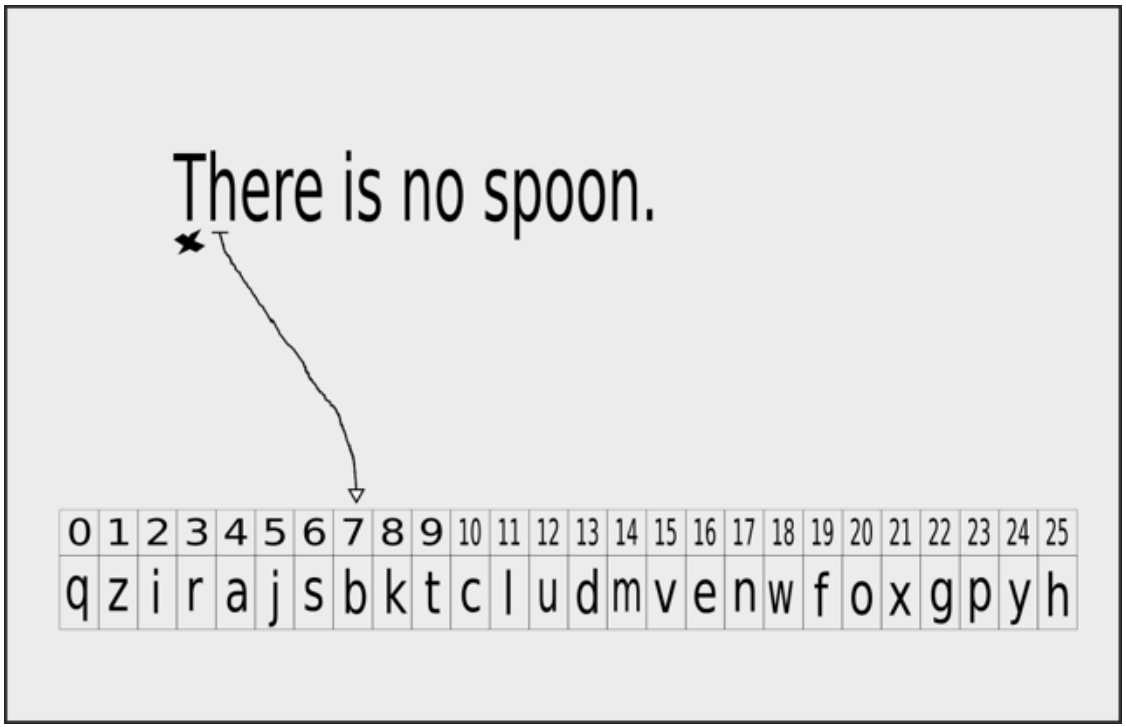
```
import random
class Encrypt:
    def __init__(self):
        self.code = [chr(i) for i in range(97, 123)]
        random.shuffle(self.code)
        self.alph = [chr(i) for i in range(97, 123)]

    def __str__(self):
        return "code: " + "".join(self.code)

    def setCode(self, data):
        self.code = list(data)

    def getCode(self):
        return "".join(self.code)
```

```
def toEncode(self, s):
        result = ""
        for i in s:
            if i in self.code: //如果是小寫a~z則進行編碼
                j = self.alph.index(i)
                result += self.code[j]
            else:
                result += i

        return result

    def toDecode(self, s):
        result = ""
        for i in s:
            if i in self.code:
                j = self.code.index(i)
                result += self.alph[j]
            else:
                result += i

        return result
```

ASCII 97 is a and 122 is (z)

encoding

- "qzirajsbktcludmvenwfoxgpyh"

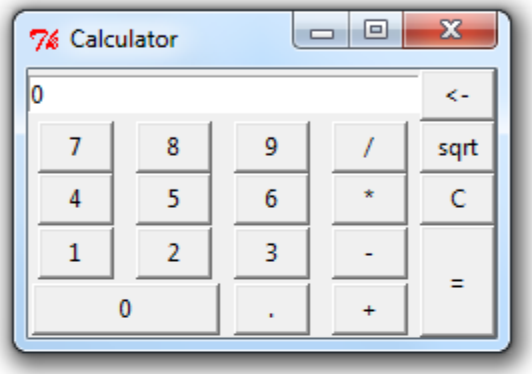


encrypt.py

```
if __name__ == '__main__':
    e = Encrypt()
    print()
    print(e)
    s1 = "There is no spoon."
    print("input: " + s1)
    s2 = e.toEncode(s1)
    print("encode: " + s2)
    s3 = e.toDecode(s2)
    print("decode: " + s3)
    print()
```

File Edit Format Run Options Windows Help

```
from Tkinter import *
import Tkinter

class GUIDemo(Tkinter.Frame):
    def __init__(self, master=None):
        Tkinter.Frame.__init__(self, master)
        self.grid()
        self.createWidgets()
        self.num = 0
        self.inputNumber = 0
        self.answer = 0
        self.inputFlag = 0
        self.op = 0
        self.opFlag = 0
        self.dotFlag = 0
        self.content = 0
        self.idx = -1

    def createWidgets(self):
        self.outputField = Entry(self)
        self.outputField["width"] = 32
        self.outputField.insert(0,"0")
        self.outputField.grid(row=0, column=0, columnspan=4)

        self.back = Button(self)
        self.back["width"] = 4
        self.back["text"] = "<-"
        self.back.grid(row=0, column=4)
        self.back["command"]= self.backMethod

        self.zero = Button(self)
        self.zero["width"] = 12
        self.zero["text"] = "0"
        self.zero.grid(row=4, column=0, columnspan=2)
        self.zero["command"]= self.zeroMethod

        self.one = Button(self)
        self.one["width"] = 4
        self.one["text"] = "1"
        self.one.grid(row=3, column=0)
```



Ln: 1 Col: 0

createWidgets

- self.outputField = Entry(self)
- self.outputField["width"] = 32
- self.outputField.insert(0,"0")
- self.outputField.grid(row=0, column=0, columnspan=4)

- self.back = Button(self)
- self.back["width"] = 4
- self.back["text"] = "<-"
- self.back.grid(row=0, column=4)
- self.back["command"]= self.backMethod

- self.zero = Button(self)
- self.zero["width"] = 12
- self.zero["text"] = "0"
- self.zero.grid(row=4, column=0, columnspan=2)
- self.zero["command"]= self.zeroMethod

createWidgets

- self.one = Button(self)
- self.one["width"] = 4
- self.one["text"] = "1"
- self.one.grid(row=3, column=0)
- self.one["command"]= self.oneMethod

- self.two = Button(self)
- self.two["width"] = 4
- self.two["text"] = "2"
- self.two.grid(row=3, column=1)
- self.two["command"]= self.twoMethod

command

```
def numberMethod(self):
    if self.dotFlag == 1:  # float situation
        self.content = self.outputField.get() + str(self.inputNumber)
        self.num = float(self.content)
    elif self.inputFlag == 0:
        self.num = self.inputNumber
    else: # non-float
        self.num = 10 * self.num + self.inputNumber
    self.inputFlag = 1
    self.outputField.delete(0, 40)
    self.outputField.insert(0, self.num)
```

command

```
def zeroMethod(self):
        self.inputNumber = 0
        self.numberMethod()

    def oneMethod(self):
        self.inputNumber = 1
        self.numberMethod()

    def twoMethod(self):
        self.inputNumber = 2
        self.numberMethod()

    def threeMethod(self):
        self.inputNumber = 3
        self.numberMethod()
```

equalMethod

```
def equalMethod(self):
    if self.op == '+':
        self.answer = self.answer + self.num
    elif self.op == '-':
        self.answer = self.answer - self.num
    elif self.op == '*':
        self.answer = self.answer * self.num
    elif self.op == '/':
        if self.num == 0:
            self.answer = 'NAN'
        else:
            self.answer = self.answer / self.num
    else:
        self.answer = self.num
        self.dotFlag = 0
    if self.answer == 0.0:
        self.answer = 0
        self.dotFlag = 0
    self.outputField.delete(0, 40)
    self.outputField.insert(0, self.answer)
    self.inputFlag = 0
    self.opFlag = 0
```

sqrtMethod

```
def sqrtMethod(self):
        if self.opFlag == 0:
            if self.answer < 0:
                self.outputField.delete(0, 40)
                self.outputField.insert(0, "invalid")
                return
            self.answer = self.answer ** 0.5
            self.outputField.delete(0, 40)
            self.outputField.insert(0, self.answer)
        else:
            if self.num < 0:
                self.outputField.delete(0, 40)
                self.outputField.insert(0, "invalid")
                return
            self.num = self.num ** 0.5
            self.c()
        self.inputFlag = 1
        self.op = 'sqrt'
```

dotMethod

```
def dotMethod(self):
        if self.opFlag == 0 and self.dotFlag == 0 and self.inputFlag == 0:
            self.answer = 0
            self.outputField.delete(0, 40)
            self.outputField.insert(0, str(self.answer) + ".")

        elif self.inputFlag == 0:
            self.num = 0.0
        else:
            self.outputField.delete(0, 40)
            self.outputField.insert(0, str(self.num) + ".")
        self.dotFlag = 1
```