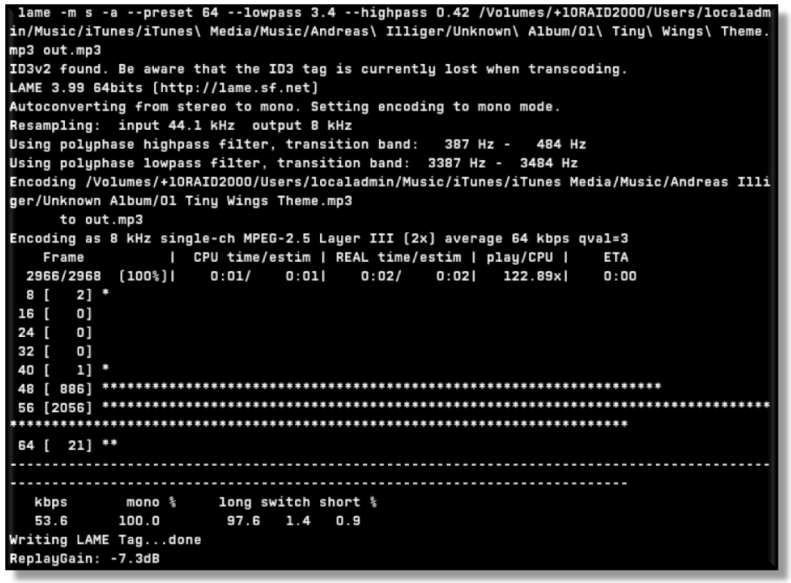
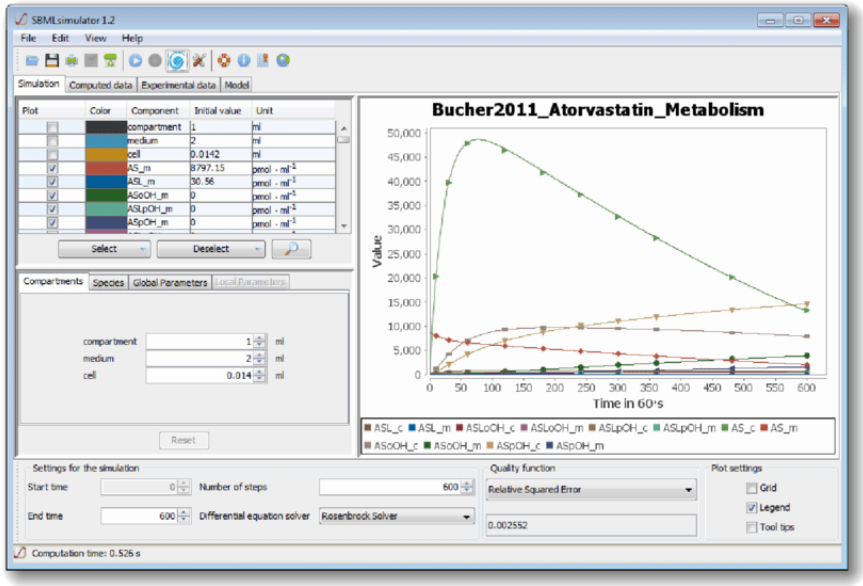
ENGR 102 PROGRAMMING PRACTICE

WEEK 3

Graphical User Interface (GUI) Programming

GUI





- Programs can interact with users in two ways:
  - ✓ Textual output, i.e., print …
  - ✓ Graphical user interface, i.e., buttons, menus, etc.
- GUI is short for "graphical user interface"
- Python has several GUI modules including Tkinter, PyQt, wxPython, etc.
- We are going to work with Tkinter.

Tkinter

- The standard Python interface to the Tk GUI toolkit.
  - Tk itself is not part of Python; it is maintained at ActiveState.
- Both Tk and tkinter are available on most Unix and Windows platforms

tkinter classes

- class Tk
  - instantiated without arguments.
    - this creates a toplevel widget of Tk which usually is the main window of an application.
  - is meant to be instantiated only once in an application.
- class Widget
  - not meant to be instantiated
  - widget classes (e.g., Button, Label, ...) inherit Widget

Terms and Concepts

- **Widget**: Elements that make up GUI.
  - Visible interactive widgets: e.g., Button
  - Invisible container widgets: e.g., Frame
- **Layout:** Organizing widgets on GUI to specific positions
- **Event:** Objects that are generated for user or system actions, such as mouse click.

Widgets

- Tkinter provides 18 built-in widget implementations.
- **Button**: A widget, containing text or an image, that performs an action when pressed.
- **Canvas**: A region that can display lines, rectangles, circles, and other shapes.
- **Entry**: A region where users can type text.
- **Scrollbar**: A widget that controls the visible part of another widget.
- **Frame**: A container, often invisible, that contains other widgets.

Hello Tkinter

```
from Tkinter import *

root = Tk()
root.title('Hello World!')

#geometry: width x height + x + y
root.geometry("250x350+300+300")

root.mainloop()
```

- Initialization → create *Tk root* which is a window.
  - One root per application
  - It should be created before anything else

Button Widget

- used to add buttons in a Python application.
- can display text or image.
- you can attach a function or a method to a button which is called automatically when the button is clicked.

Hello, Again! - ButtonWidget

```
from Tkinter import *

class HelloApp(Frame):

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

    def initUI(self):
       self.hi_there = Button(self, text="Hello", command=self.say_hi)

       self.hi_there.pack()
       self.pack()

    def say_hi(self):
        print "hi there, everyone!"

def main():
    root = Tk()
    root.geometry("250x150+300+300")
    root.title('Hello World!')
    app = HelloApp(root)
    root.mainloop()

main()
```

Checkbutton Widget

- provides a check box with a text label.
- has two states: on and off.
- the on state is visualized by a check mark.
- used to denote some boolean property (i.e., True, False).

Checkbutton Widget

```
from Tkinter import *
class HelloApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        self.var = BooleanVar()
        cb = Checkbutton(self, text="Show title", variable=self.var,
                         command=self.onClick)
        self.pack()
        cb.pack()

    def onClick(self):
        if self.var.get() == True:
            self.parent.title("Hello CheckButton!")
        else:
            self.parent.title("")

def main():
    root = Tk()
    root.geometry("300x150+300+300")
    root.title("Hello CheckButton!")
    app = HelloApp(root)
    root.mainloop()

main()
```

Label Widget

- used to display text or images.

```
from Tkinter import *
class HelloApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        var = StringVar()
        label = Label(self, textvariable=var)
        var.set("Hey!? How are you doing?")

        label.pack()
        self.pack()

def main():
    root = Tk()
    root.geometry("250x150+300+300")
    app = HelloApp(root)
    root.mainloop()

main()
```

Example

- Write an app with a button, a label and a checkbox
  - app maintains a state for a shop: is opened or not
  - if shop is opened,
    - label shows “opened”
    - checkbox is checked
    - button text shows “close”
  - else
    - label shows “closed”
    - checkbox is unchecked
    - button text shows “open”
  - Button and checkbox clicks swaps state

ListBox Widget

- Displays a list of entries.
- Allows selecting one or multiple items.

```
from Tkinter import *
class CitySelector(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        self.pack()
        cities = ['Uskudar', 'Kadikoy',
                  'Maltepe', 'Kartal']

        lb = Listbox(self, selectmode='multiple')
        for i in cities:
            lb.insert(END, i)

        lb.pack(pady=15)

def main():
    root = Tk()
    root.geometry("250x150+300+300")
    app = CitySelector(root)
    root.mainloop()

main()
```

Entry Widget

- used to accept single-line text from a user.
- use the *Label* widget if
  - you want to display one or more lines of text that cannot be modified by the user.
- Use *Text* widget if
  - you want to display multiple lines of text that can be edited

```
from Tkinter import *
class UserInfo
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):

        la = Label(self, text="User Name")
        en = Entry(self)

        self.pack()
        la.pack()
        en.pack()

def main():
    root = Tk()
    root.geometry("250x150+300+300")
    app = UserInfo(root)
    root.mainloop()

main()
```

Example

- Write an app with an entry, button and a label
  - When the user clicks on the button
    - label shows n+1 where n is the numeric input in the entry
    - if the input is not numeric, label shows an error

Text Widget

- provides formatted text display.
- allows you to display and edit text with various styles and attributes.
- supports embedded images and windows.

```
from Tkinter import *
class TextEditor(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        text = Text(self)
        text.insert(INSERT, "Hello.....")
        text.insert(END, "Bye Bye.....")

        text.tag_add("here", "1.0", "1.4")
        text.tag_add("start", "1.8", "1.13")
        text.tag_config("here", background="yellow", foreground="blue")
        text.tag_config("start", background="black", foreground="green")

        self.pack()
        text.pack()

def main():
    root = Tk()
    root.geometry("250x150+300+300")
    app = TextEditor(root)
    root.mainloop()

main()
```

Canvas Widget – Drawing Lines

- a general purpose widget
- display and edit graphs and other drawings
- implement various kinds of custom widgets, e.g., a progress bar

```
from Tkinter import *
class DrawerApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        canvas = Canvas(self)
        canvas.create_line(15, 25, 200, 25)
        canvas.create_line(300, 35, 300, 200, dash=(4, 2))
        canvas.create_line(55, 85, 155, 85, 105, 180, 55, 85)

        self.pack()
        canvas.pack()

def main():
    root = Tk()
    root.geometry("350x250+300+300")
    app = DrawerApp(root)
    root.mainloop()

main()
```

Canvas Widget – Drawing Shapes

```
from Tkinter import *

class DrawerApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        canvas = Canvas(self)
        canvas.create_oval(10, 10, 80, 80, outline="gray", fill="blue", width=2)
        canvas.create_oval(110, 10, 210, 80, outline="gray", fill="red", width=2)
        canvas.create_rectangle(230, 10, 290, 60, outline="gray", fill="green", width=2)

        points = [150, 100, 200, 120, 240, 180, 210, 200, 150, 150, 100, 200]
        canvas.create_polygon(points, outline='gray', fill='yellow', width=2)

        self.pack()
        canvas.pack()

def main():
    root = Tk()
    root.geometry("350x250+300+300")
    app = DrawerApp(root)
    root.mainloop()

main()
```

Example

- Write an app that
  - draws a small red balloon and large blue balloon in a canvas