A brief demonstration of some Python GUI libraries

Primož Podržaj
University of Ljubljana, Faculty of Mechannical Engineering
Aškerčeva 6, 1000 Ljubljana, Slovenia
primoz.podrzaj@fs.uni-lj.si

ABSTRACT

In this paper some possible solutions of graphical user interfaces (GUIs) within Python programming language are presented. GUIs are becoming increasingly important, because human beings are more and more involved in dealing with computers for every day life. GUIs are however part of computer systems that presents the border between computer and human being. It is therefore very important for any serious programming language to offer easy to use solutions in the area of GUI development. In this paper a few possible GUI libraries for Python are presented. A very basic examples of their application are also given in order to give a sense of their complexity. Two examples which can be useful in educational environment are also presented at the end of the paper.

KEYWORDS

Python, GUI, Graphical user interface, Jupyter Notebook, Tkinter, PySimpleGUI, Flexx, Ipywidgets.

1 INTRODUCTION

Humans are relying on computers and digital data more and more. Graphical user interface (GUI) is a part of computer program designed specifically for interaction with humans. It is therefore extremely important for every serious programming language to offer solutions in this regard, especially if it is regarded to be a high level programming language.
Python is a general-purpose programming language created by Guido vanRossum in 1991. The first release was followed by Python 2.0 in 2000 and Python 3.0 in 2008. At this moment the latest version is Python 3.7. Compared to other programming languages, Python is a high level programming language and consequently a little slower than C/C++ for example, but due to its emphasis on readability, it is becoming more and more popular [1]. Python is a good choice for all the researchers in the scientific community because it is [2]:

- free and open source
- a scripting language, meaning that it is interpreted
- a modern language (object oriented, exception handling, dynamic typing etc.)
- concise, easy to read and quick to learn
- full of freely available libraries, in particular scientific ones (linear algebra, visualization tools, plotting, image analysis, differential equations solving, symbolic computations, statistics etc.)
- useful in a wider setting: scientific computing, scripting, web sites, text parsing, etc.
- widely used in industrial applications

Especially the availability of a vast array of free libraries makes it very popular for research. In this paper some of the libraries used for the design of GUI will be presented and very basic examples of their application will be given.

2 PYTHON GUI TOOLKITS

There are many GUI libraries available for Python. The most common ones are: Tkinter, Kivy, PyQt, WxPython, Libavg, PyGUI, PySide, Pyforms, Wax Python GUI, PySimpleGUI, Flexx, IPyWidgets, etc. [3],[4],[5]. It is of course impossible to present all of them in a short conference paper, so four of them (Tkinter, PySimpleGUI, Flexx and Tkinter) were selected for a brief presentation.

ISBN: 978-1-941968-59-8 ©2019 SDIWC

2.1 Tkinter

Tkinter is a Python interface to the Tk GUI library and has been a part of the Python standard library since 1994 with the release of Python version 1.1, making it the de facto GUI library for Python [6]. The Tk widget library originates from the Tool Command Language (Tcl) programming language. Tcl and Tk were created by John Ousterman in the late 1980s as an easier way to program engineering tools while being used at the university. Tkinter is nowadays arguably the most common GUI framework in Python GUI library. The features that make it a great choice for GUI programming include the following [7],[8]:

- It is quite simple to learn (simpler than any other GUI package for Python).
- Relatively little code can produce powerful GUI applications.
- Layered design ensures that it is easy to grasp.
- It is portable across all operating systems.
- It is easily accessible, as it comes preinstalled with the standard Python distribution.
- Code can be written both procedurally or using object-oriented practices (which is the preferred style for anything nonexperimental).

It might seem at first sight that Tkinter is not going to do well, because the Python interpreter is using the Tkinter module which, in turn, uses the tkinter interface, which then calls Tcl and Tk libraries and sometimes also the Tcl interpreter in order to bind properties to widgets [9]. Despite this it is performing very well on modern computers. It should be noted, that starting from Python 3, the Tkinter module was renamed to tkinter [10]. An outline for its application is shown in the following code [12]:

```
import tkinter
root = tkinter.Tk()
# Code to add widgets ...
root.mainloop()
```

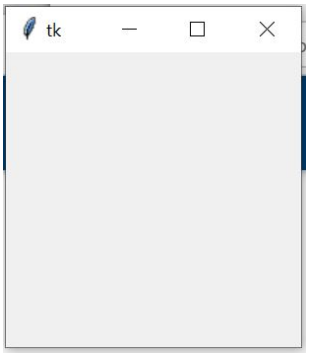

Figure 1. The root window

These three lines (excluding commented line) result in the root window (shown in Fig. 1) appearing on the screen. In the first line of code we just import the tkinter module into the namespace. After that, all its definitions of the classes, attributes and methods can be accessed. The second line created the root window, which is an instance of tkinter.Tk class. The third line is just a comment, where we will be later adding other code. The forth line is an event loop method of the root object and keeps the root window visible. Without it, the window created in the second line would disappear too fast. In order to make some useful application using Tkinter, we need widgets. Tkinter includes 21 core widgets, as shown in Tab. 1. The code for an example using Label and Button widgets of Tkinter is stated below:

Table 1. The Tkinter's 21 core widgets [7]

Top-level	Label	Button
Frame	LabelFrame	Listbox
Menu	Menubutton	Message
OptionMenu	PanedWindow	Radiobutton
Scale	Scrollbar	Spinbox
Text	Bitmap	Image

```
import tkinter as tk
class Window(tk.Tk):
  def __init__(self):
    super().__init__()
    self.title("Hello Tkinter")
    self.label = tk.Label(self,
     text="Choose One")
    self.label.pack(fill=tk.BOTH,
     expand=1, padx=100, pady=10)
    hello_btn = tk.Button(self,
     text="Hello", command=self.hello)
    hello_btn.pack(side=tk.LEFT,
     padx=(20, 0), pady=(0, 20))
    goodbye_btn = tk.Button(self,
     text="Goodbye", command=self.goodbye)
```

ISBN: 978-1-941968-59-8 ©2019 SDIWC

```
    goodbye_btn.pack(side=tk.RIGHT,
     padx=(0, 20), pady=(0, 20))
  def hello(self):
    self.label.configure(text="Hello!")
  def goodbye(self):
    self.label.configure(text="Goodbye!
     \n (Closing in 3 seconds)")
    self.after(3000, self.destroy)
if __name__ == "__main__":
  window = Window()
  window.mainloop()
```

When we execute the program the window titled ”Hello Tkinter” appears as shown in Fig. 2. Its label says ”Choose one” in order

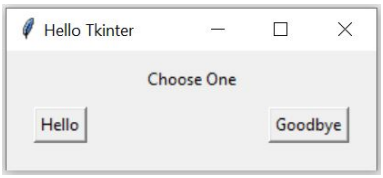

Figure 2. An example of a window for UI

to press one of the buttons ”Hello” or ”Goodbye”. If we press ”Hello” button, the label will change to ”Hello”. If we however press ”Goodbye” button, we get the window shown in Fig. 3, which will close after three seconds. The placing of the label or buttons is controlled

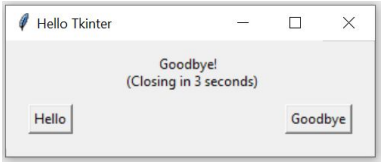

Figure 3. An example of a window for UI

using ”pack” and its attributes. The ”side” argument is used for placing of widgets. So ”side.tk.left” says that a certain button should be at the left side of the window. The ”padx” and ”pady” arguments are used to determine the amount of space left/right and above/below a certain button.

2.2 PySimpleGUI

PySimpleGUI [13] (Python 3 version) or PySimpleGUI27 (Python 2.7 version), created in 2018, is sort of a wrapper for tkinter, offering the creation of GUI elements of the same look and functionality as tkinter does. Besides the basic version, there are also versions that provide ports to the Qt GUI Framework (PySimpleGUIQt), WxPython (PySimpleGUIWx) and Remi (PySimpleGUIWeb). These versions cover fresher GUI features, SystemTray Icon features and web functionality, respectively. PySimpleGUIs main attribute is its simplicity, since it requires up to one half less amount of code in order to produce the same result in comparison to the underlying frameworks and removes the requirement for object-oriented programming. PySimpleGUI was actually built with the purpose to offer the developers to get the required output in as small amount of code as possible. For example, upon installing and importing the library, a single line of code (PySimpleGUI.Popup(’Popup window in one line of code’)) is required to display a popup window with a message (see Figure XX). Additional features of PySimpleGUI are the possibility of the simultaneous creation of desktop- and web-based GUIs, availability of more than 100 demo programs (including Machine Learning examples) together with a cookbook [14] for getting familiar with the library, integration with Matplotlib and Pyplot etc. Last but not least, the author offers active support and updates are being published regularly. The goal of the author is to help Python become the programming language of choice when it comes to the creation of GUI applications running on Windows, Mac and Linux operating systems.

2.3 Flexx

Flexx is a pure Python toolkit for creating graphical user interfaces (GUIs), that uses web technology for its rendering. Apps are written purely in Python; The PScript transpiler generates the necessary JavaScript on the fly [15]. Flexx is very easy to use and quite fast to execute. An example code needed to form a slider and a button is given below:

```
from flexx import flx
class Example(flx.Widget):
  def init(self):
    with flx.VBox():
      self.slider = flx.Slider(min=1,
       max=10, value=5, flex=1)
      self.button = flx.Button(
```

```
        text='Press')
      self.buttonlabel= flx.Label(
        text='...')
  @flx.reaction('button.pointer_click')
  def _button_clicked(self, *events):
    ev = events[-1]
    self.buttonlabel.set_text(
     'Clicked on the ' + ev.source.text)
if __name__ == '__main__':
  m = flx.launch(Example)
  flx.run()
```

When the button is pressed, we get the result shown in Fig. 4.

Figure 4. A simple example of UI made in Flexx

2.4 Ipywidgets

Another very efficient way of getting interactivity into Python is the so called ipywidgets library. It is for example very easy to interact with numerical values. Widgets exist for displaying integers and floats, both bounded and unbounded. The naming scheme for both is very similar. The only difference is whether the name begins Float or Int. An example is a slider shown in Fig. 5. In order to get it the

Figure 5. An example of IntSlider using ipywidgets library

following code must be entered:

```
import ipywidgets as widgets
widgets.IntSlider(
    value=7,
    min=0,
    max=10,
    step=1,
    description='Value:',
    disabled=False,
    continuous_update=False,
    orientation='horizontal',
    readout=True,
    readout_format='d'
)
```

In the first line we just import the ipywidgets library and give it an abbreviation widgets. In the second line the IntSlider is created with the default value of 1, minimal value 0 and maximal value 10. When the slider is being moved, the values will change in steps equal to 1. With the description attribute it is as the name suggests possible to describe the slider, i.e. state what it is used for. With orientation it is for example also possible to have it in horizontal or vertical direction.
When dealing with scientific applications there is an interesting and a very useful widget called HTMLMath widget. An example of an application of such a widget is shown in Fig. 6. It

Some HTML: Some math and *HTML*:

$$\int_0^\infty \sin^2(x)dx$$

and

$$\frac{x^3+7}{x^2+3}$$

Figure 6. An example of HTMLMath widget

makes it possible to enter math text in Latex style commands. The result shown in Fig. 6 was obtained using the following code:

```
import ipywidgets as widgets
widgets.HTMLMath(
  value=r"Some math and <i>HTML</i>:
   $$\int_0^\infty \sin^2(x) dx$$
   and $$\frac{x^3+7}{x^2+3}$$",
  placeholder='Some HTML',
  description='Some HTML:',
)
```

3 GUI RELATED APPLICATIONS

GUI plays an important part in various applications. The first example in this paper is the visualization of the function:

$$x(t) = e^{at}\sin(\omega t + \varphi) \qquad (1)$$

where parameters a, ω and φ can be set in the GUI. The code (using Flexx as GUI) is given below:

```
from flexx import flx
class Example(flx.Widget):
  def init(self):
    time = [i/100 for i in range(100)]
    with flx.VBox():
      with flx.HBox():
```

```
        flx.Label(text='Frequency:')
        self.sl1 = flx.Slider(min=1,
         max=10, value=5, flex=1)
        flx.Label(text='Phase:')
        self.sl2 = flx.Slider(min=0,
         max=6, value=0, flex=1)
        flx.Label(text='Exponent:')
        self.sl3 = flx.Slider(min=-2,
         max=2, value=0, flex=1)
        self.plot = flx.PlotWidget(
         flex=1, xdata=time, xlabel='time',
         ylabel='value', title='Graph')
  @flx.reaction
  def __update_amplitude(self, *events):
    global Math
    freq, phase, exponent = self.sl1.value,
     self.sl2.value, self.sl3.value
    ydata = []
    for x in self.plot.xdata:
      ydata.append(Math.sin(freq*x*2*Math.PI+
       phase)*Math.exp(exponent*x))
    self.plot.set_data(self.plot.xdata, ydata)
if __name__ == '__main__':
  m = flx.launch(Example)
  flx.run()
```

The result is shown in Fig. 7.

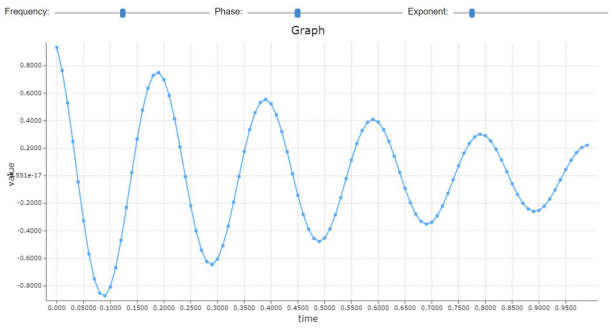

Figure 7. A damped sinusoid example in Flexx

As the second example a partial fraction expansion will be demonstrated. Let's say that we want to determine a step response of a system with the following transfer function:

$$P(s) = \frac{a}{(s^2 + 2s + 1)(s + 3)} \qquad (2)$$

We want to analyze the system when parameter a is varied. The output $Y(s) = P(s) \cdot X(s)$, where $X(s) = \frac{1}{s}$, is therefore equal to:

$$Y(s) = \frac{a}{(s^2 + 2s + 1)(s + 3)} \cdot \frac{1}{s} \qquad (3)$$

The partial fraction expansion, the inverse Laplace transform $y(t)$ and its graph can be obtained by the following code:

```
from ipywidgets import interact
import scipy
from sympy import *
import ipywidgets as widgets
from ipywidgets import interactive
from IPython.display import display
import matplotlib.pyplot as plt
import numpy as np
init_printing(use_unicode=True)
s=Symbol('s')
ts=Symbol('ts')
def func(a):
    Ys = a/(s*(s**2 +2*s+ 1)*(s+3))
    r=apart(Ys)
    ys = inverse_laplace_transform(Ys,s,ts)
    print('Y')
    print(Ys)
    print('y')
    print(ys)
    t=np.arange(0.0, 10.0, 0.01)
    y = np.zeros(len(t))
    for i in range(len(t)):
        y[i] = ys.subs(ts,t[i])
    plt.figure()
    plt.plot(t,y,label='y(t)')
    plt.legend()
    plt.xlabel('Time')
    plt.show()
    return (r)
interact(func, a=(0.1,10));
```

The result of the code for $a = 2$ is shown in Fig. 8. For $a = 6$ the same code gives the

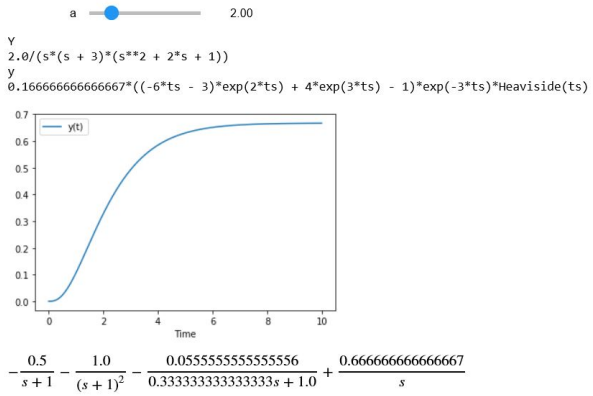

Figure 8. An example of partial fraction example for $a = 2$

result shown in Fig. 9. On this example we can clearly see the advantages of the application of GUI (in this case made with Ipywidgets library) for system response analysis purposes.

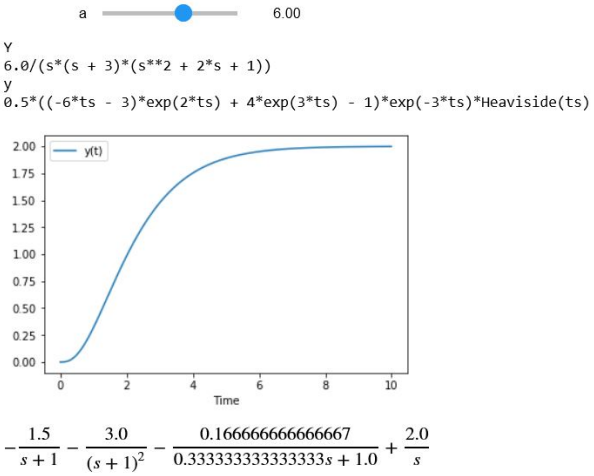

Figure 9. An example of partial fraction example for $a = 2$

4 CONCLUSION

In the paper four popular Python GUI libraries are briefly presented. Their way of application is similar in some respects but different in others. Mainly the personal preference determines which library is preferred by each individual. In this paper only the most basic examples are presented for each GUI library. Two educational examples are also presented.
In future it would be interesting to make a research of more GUIs and not only compare them based on the (visual) preference, but also based on the performance and versatility.

REFERENCES

[1] https://www.tiobe.com/tiobe-index/

[2] C. Fuhrer, J. E. Solem, and O. Verdier, Scientific Computing with Python 3, Packt Publishing Ltd, 2016.

[3] https://techsore.com/best-python-gui/

[4] https://insights.dice.com/2017/08/07/7-top-python-gui-frameworks/

[5] https://unwiredlearning.com/blog/top-python-gui-frameworks/

[6] A. D. Moore, Python GUI Programming with Tkinter. Packt Publishing Ltd., 2018.

[7] B. Chaudhary, Tkinter GUI application development blueprints: build nine projects by working with widgets, geometry management, event handling, and more. Packt Publishing Ltd., 2018.

[8] D. Love, Tkinter GUI Programming by Example. Packt Publishing Ltd., 2018.

[9] J. E. Grayson, Python and Tkinter programming. Greenwich: Manning,2000.

[10] A. Rodas de Paz, Tkinter GUI Application Development Cookbook. Packt Publishing Ltd., 2018.

[11] M. Roseman, Modern Tkinter for Busy Python Developers: Quickly learn to create great looking user interfaces for Windows, Mac and Linux using Python's standard GUI toolkit. Late Afternoon Press, 2012.

[12] https://www.tutorialspoint.com/python3/index.htm

[13] https://pysimplegui.readthedocs.io/en/latest/

[14] https://pysimplegui.readthedocs.io/en/latest/cookbook/

[15] https://flexx.readthedocs.io/en/stable/

[16] https://ipywidgets.readthedocs.io/en/stable/