FRAMEWORKS E BIBLIOTECAS PARA INTERFACE GRÁFICA

Frameworks e bibliotecas para GUI

Conceitos

A linguagem python possui muitos frameworks para desenvolvimento de aplicações de interface gráfica para interação com o usuário, chamadas, comumente, de GUI (Graphical User Interface).

O framework mais comum é o Tkinter (python interface to Tcl/Tk-2020) que já faz parte da instalação python, mas existem outros frameworks com características específicas que podem torná-los a escolha adequada para um projeto.

Entre os frameworks para aplicações GUI mais comuns estão:

Explorando Frameworks e Bibliotecas para GUI em Python: Uma Visão Abrangente

Neste vídeo, vamos explorar uma variedade de frameworks e bibliotecas populares para criação de interfaces gráficas em Python, incluindo Tkinter, Flexx, CEF Python, Kivy, Pyforms, PyQt, wxPython, PyAutoGUI e PySimpleGUI. Discutiremos as características distintas de cada um, suas aplicações e vantagens, oferecendo uma visão abrangente das opções disponíveis para desenvolvimento de GUIs em Python.

Tkinter

É o framework GUI padrão do python. Sua sintaxe é simples, possui muitos componentes para interação com o usuário. Além disso, seu código é aberto e é disponível sob a licença python. Caso ela não esteja instalada na sua versão do python, basta digitar o comando:

```
pip install tkinter
```

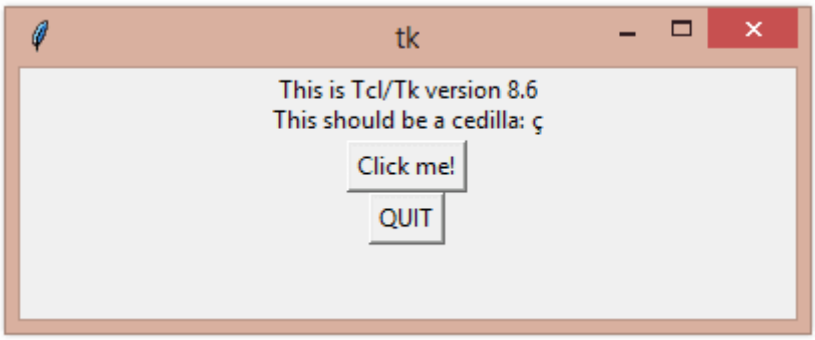
Para quem usa a IDE Spyder (SPYDER, 2020), é necessário colocar uma exclamação antes do comando “pip”, ou seja:

```
!pip install tkinter
```

Para testar a instalação, basta escrever o seguinte código na linha de comando:

```
import tkinter
tkinter._test()
```

Se a instalação ocorreu normalmente, aparecerá uma tela com alguns componentes para que você possa interagir e ter uma impressão inicial do potencial desse framework:



Exemplo de aplicação do Tkinter.

Atenção!

Devido à importância do Tkinter, vamos explorá-lo com bastantes detalhes mais à frente.

Flexx

É um kit de ferramentas para o desenvolvimento de interfaces gráficas com o usuário implementado em python que faz uso de tecnologia web para sua renderização. O Flexx pode ser usado para criar tanto aplicações de desktop como para web e até mesmo exportar uma aplicação para um documento HTML independente. Para instalar o Flexx, basta digitar o comando:

```
pip install flexx
```

Para quem usa a IDE Spyder, é necessário colocar uma exclamação antes do comando “pip”, isso vale para qualquer instalação de biblioteca/framework/pacote.

Comentário

Para evitar repetições sobre isso, apresentaremos aqui como instalar o pacote Flexx na IDE Spyder, mas, para os próximos casos, mostraremos apenas a instalação tradicional, ou seja, sem o símbolo de “exclamação”.

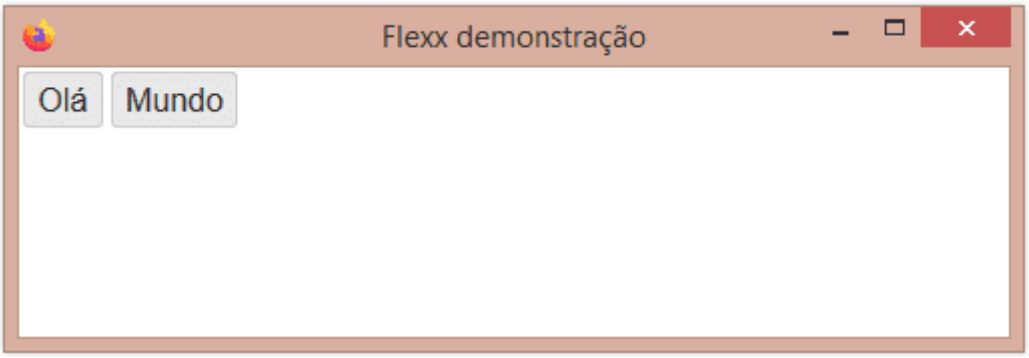
No caso do Spyder, o comando deve ser:

```
!pip install flexx
```

Uma forma prática de testar a instalação e aprender um pouco mais sobre esse framework é escrever o código abaixo na linha de comando, ou em um arquivo, e executar:

```
from flexx import flx
class Exemplo(flx.Widget):
    def init(self):
        flx.Button(text='Olá')
        flx.Button(text='Mundo')
if __name__ == '__main__':
```

Se a instalação estiver correta, a seguinte janela vai se abrir:



Exemplo de aplicação com Flexx.

CEF python

É um projeto de código aberto voltado para o desenvolvimento de aplicações com integração ao Google Chrome. Existem muitos casos de uso para CEF. Por exemplo, ele pode ser usado para criar uma GUI baseada em HTML 5, pode usá-lo para testes automatizados, como também pode ser usado para web scraping, entre outras aplicações.

Para instalá-lo, basta digitar na linha de comando:

```
pip install cefpython3
```

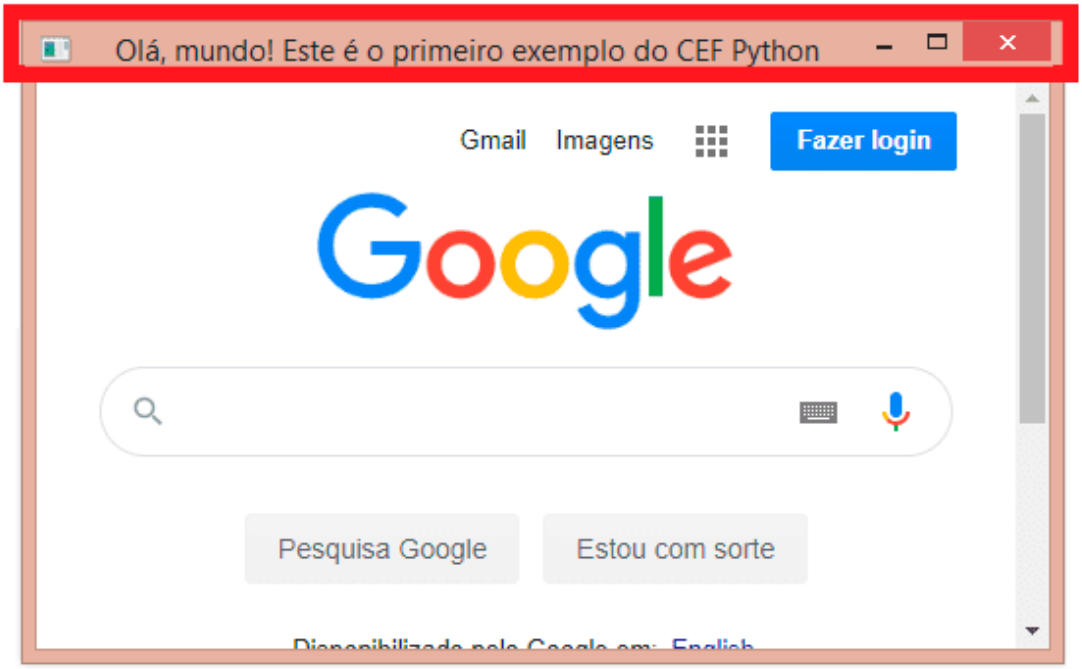
Para testar a instalação do CEF python, basta escrever o código abaixo na linha de comando, ou em um arquivo, e executar:

```
from cefpython3 import cefpython as cef
import platform
import sys
def main():
    check_versions()
    sys.excepthook = cef.ExceptHook # To shutdown all CEF processes on error
    cef.Initialize()
```

Atenção!

Como esse exemplo, apesar de simples, é um pouco maior, cabe lembrar que a indentação faz parte da sintaxe do python (python 3.8.5 documentation, 2020), por isso, é necessário ter bastante cuidado, caso contrário o programa apresentará erros.

Se tudo funcionou corretamente, quando o programa executar, abrirá a seguinte janela:



Exemplo de aplicação com CEF python.

Observe o título da janela: “Olá, mundo! Este é o primeiro exemplo do CEF python”.

Kivy

É um framework python de código aberto para o desenvolvimento de aplicações com interfaces de usuário e multitoque. Ele é escrito em python e Cython, baseado em OpenGL ES 2, suporta vários dispositivos de entrada e possui uma extensa biblioteca de componentes (widgets).

Com o mesmo código, a aplicação funciona para Windows, macOS, Linux, Android e iOS. Todos os widgets Kivy são construídos com suporte multitoque.

Para instalá-lo, é necessário escrever na linha de comando:

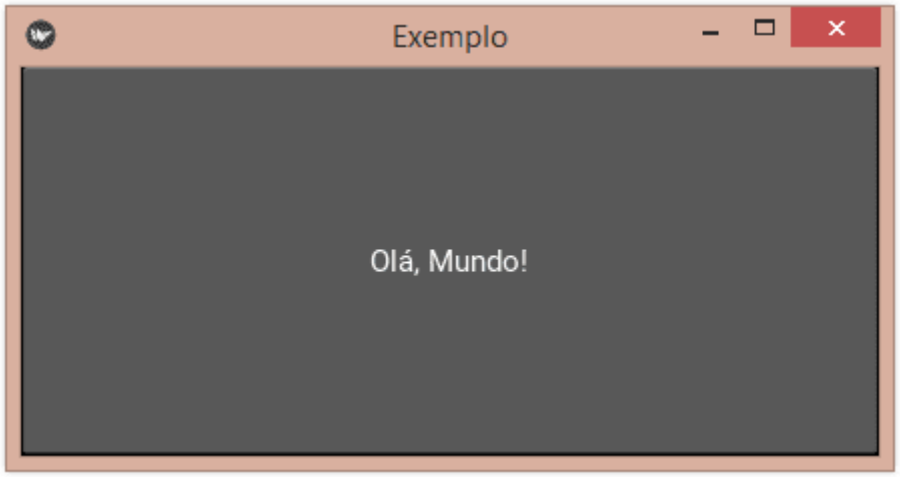
```
pip install Kivy
```

Uma forma de testar a instalação é escrever e executar o programa:

```
from kivy.app import App
from kivy.uix.button import Button
```

```
class ExemploApp(App):
    def build(self):
        return Button(text='Olá, Mundo!')
```

Se tudo funcionar corretamente, aparecerá a aplicação, conforme a próxima imagem.



Exemplo de aplicação Kivy.

Na imagem anterior, é possível identificar o título da janela “Exemplo” e um componente botão no centro da tela com a mensagem “Olá, Mundo!”. Um ponto interessante pode ser observado no código, que é a utilização da programação orientada a objetos.

Pyforms

É um framework python 3 para desenvolver aplicações que podem operar nos ambientes Desktop GUI, Terminal e Web. A biblioteca é composta por três sub-bibliotecas, cada uma implementando a camada responsável por interpretar a aplicação Pyforms em cada ambiente diferente:

1. Pyforms-gui.
2. Pyforms-web.
3. Pyforms-terminal.

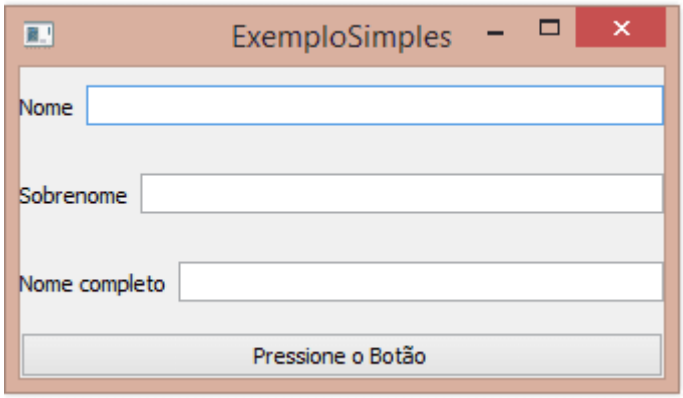
Essas camadas podem ser usadas individualmente ou em conjunto, dependendo da instalação do Pyforms. Para fazer a instalação básica, é necessário escrever na linha de comando:

```
pip install pyforms
```

Uma forma de testar a instalação é escrever e executar o programa:

```
import pyforms
from pyforms.basewidget import BaseWidget
from pyforms.controls import ControlText
from pyforms.controls import ControlButton
class ExemploSimples(BaseWidget):
    def __init__(self):
```

Se tudo funcionar corretamente, aparecerá a aplicação conforme a imagem seguinte.



Exemplo de aplicação Pyform.

Comentário

No exemplo da imagem anterior, é possível ver três caixas de texto, chamadas de "controle de texto", e um componente botão, chamado de "controle de botão" pelo Pyforms. Ele foi projetado a fim de desenvolver aplicações para executar no modo Windows GUI.

PyQt

Uma aplicação desenvolvida no framework PyQt e executada nas plataformas Windows, macOS, Linux, iOS e Android.

Trata-se de um framework que aborda, além de desenvolvimento GUI, abstrações de sockets de rede, threads, Unicode, expressões regulares, bancos de dados SQL, OpenGL, XML, entre outras aplicações.

Suas classes empregam um mecanismo de comunicação segura entre objetos que é fracamente acoplada, tornando mais fácil criar componentes de software reutilizáveis.

Para fazer a instalação básica, é necessário escrever na linha de comando:

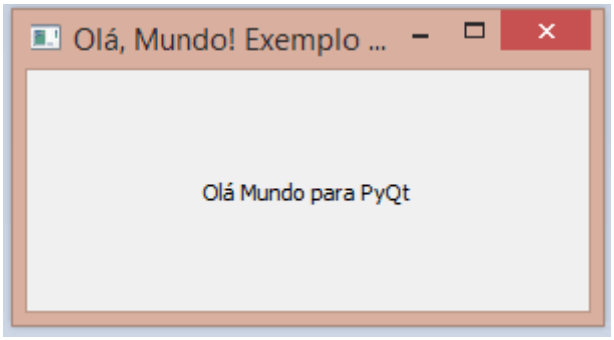
```
pip install PyQt5
```

Uma forma de testar a instalação é escrever e executar o programa:

```
import sys
from PyQt5 import QtCore, QtWidgets
from PyQt5.QtWidgets import QMainWindow, QLabel, QGridLayout,
QWidget
from PyQt5.QtCore import QSize
class HelloWindow(QMainWindow):
  def __init__(self):
    QMainWindow.__init__(self)
    self.setMinimumSize(QSize(280, 120))
    self.setWindowTitle("Olá, Mundo! Exemplo PyQT5")
    centralWidget = QWidget(self)
    self.setCentralWidget(centralWidget)
    gridLayout = QGridLayout(self)
    centralWidget.setLayout(gridLayout)
```

```
    title = QLabel("Olá Mundo para PyQt", self)
    title.setAlignment(QtCore.Qt.AlignCenter)
    gridLayout.addWidget(title, 0, 0)
```

Se tudo funcionar corretamente, aparecerá a aplicação conforme a imagem a seguir.



Exemplo de aplicação PyQt.

No exemplo da imagem acima, é possível identificar uma janela, o título da janela e uma mensagem ao centro.

wxPython

É um kit de ferramentas GUI baseadas em uma biblioteca C++ chamada wxWidgets que foi lançada em 1998. O wxpython usa os componentes (widgets) reais na plataforma nativa sempre que possível. Essa, inclusive, é a principal diferença entre o wxpython e outros kits de ferramentas, como PyQt ou Tkinter.

Atenção!

As aplicações desenvolvidas em wxpython se assemelham a aplicações nativas do sistema operacional em que estão sendo executadas.

As bibliotecas PyQt e Tkinter têm componentes personalizados. Por isso é bastante comum que as aplicações fiquem com um aspecto diferente das nativas do sistema operacional. Apesar disso, o wxpython também oferece suporte a componentes personalizados.

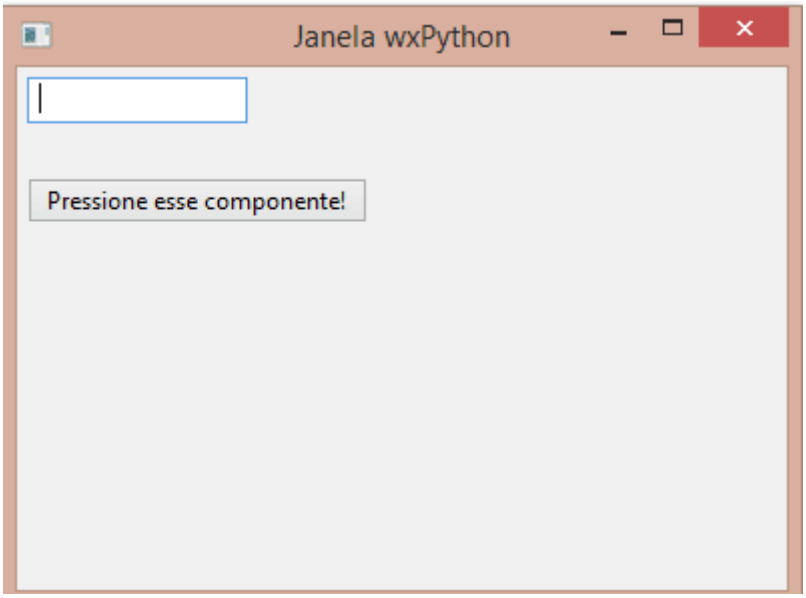
Para fazer a instalação básica, é necessário escrever na linha de comando:

```
pip install wxpython
```

Uma forma de testar a instalação é escrever e executar o programa:

```
import wx
class Janela(wx.Frame):
  def __init__(self, parent, title):
    super(Janela, self).__init__(parent, title=title, size = (400,300))
    self.panel = ExemploPainel(self)
    self.text_ctrl = wx.TextCtrl(self.panel, pos=(5, 5))
    self.btn_test = wx.Button(self.panel, label='Pressione esse componente!', pos=(5, 55))
class ExemploPainel(wx.Panel):
  def __init__(self, parent):
    super(ExemploPainel, self).__init__(parent)
class ExemploApp(wx.App):
  def OnInit(self):
    self.frame = Janela(parent=None, title="Janela wxPython")
    self.frame.Show()
    return True
```

Se tudo funcionar corretamente, aparecerá a aplicação, conforme a imagem seguinte.



Exemplo de aplicação wxpython.

No exemplo da imagem anterior, é possível identificar uma janela, o seu respectivo título, uma caixa de texto e um botão com a mensagem “Pressione esse componente!”.

PyAutoGUI

Permite desenvolver aplicações python que controlem o mouse e o teclado para automatizar as interações com outros aplicativos.

Comentário

Uma das situações em que essa característica pode ser muito interessante é na implementação de testes que simulem a interação do usuário com o sistema.

O PyAutoGUI funciona no Windows, macOS e Linux e é executado no python.

Para fazer a instalação básica, é necessário escrever na linha de comando:

```
!pip install PyAutoGUI
```

Uma forma de testar a instalação é escrever e executar o programa:

```
import pyautogui
screenWidth, screenHeight = pyautogui.size()
```

```
currentMouseX, currentMouseY = pyautogui.position()
pyautogui.moveTo(100, 150)
pyautogui.click()
pyautogui.click(100, 200)
pyautogui.move(0, 10)
pyautogui.doubleClick()
```

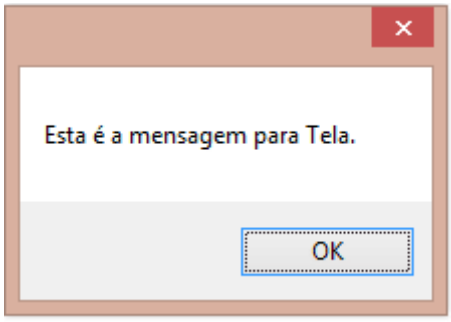
Nesse código, basicamente tem-se esta sequência de instruções:

1. Obter o tamanho do monitor principal.
2. Obter a posição XY do mouse.
3. Mover o mouse para as coordenadas XY.
4. Clicar com o mouse.
5. Mover o mouse para as coordenadas XY e clicar nelas.
6. Mover o mouse 10 pixels para baixo de sua posição atual.
7. Clicar duas vezes com o mouse.
8. Usar a função de interpolação/atenuação para mover o mouse por 2 segundos com pausa de um quarto de segundo entre cada tecla.
9. Pressionar a tecla Esc.
10. Pressionar a tecla Shift e segurá-la.
11. Pressionar a tecla de seta para a esquerda 4 vezes.
12. Soltar a tecla Shift.
13. Pressionar a combinação de teclas de atalho Ctrl-C.
14. Mostrar uma caixa de alerta aparecer na tela e pausar o programa até clicar em OK.

Atenção!

Antes de executar o código, cabe um alerta: Essa aplicação, apesar de ser muito simples, vai interagir com o seu sistema.

Se tudo funcionar corretamente, aparecerá a aplicação conforme a próxima imagem.



Exemplo de aplicação PyAutoGUI.

A imagem vista anteriormente aparecerá depois do “cursor do mouse” se movimentar na tela e o “teclado” escrever a mensagem “Esta é a mensagem para tela”.

As possibilidades de aplicações são muitas para o PyAutoGUI.

PySimpleGUI

Esse pacote foi lançado em 2018 e possui portabilidade com os pacotes: Tkinter, PyQt, wxpython e Remi, portanto aumenta as possibilidades de uso de componentes na programação.

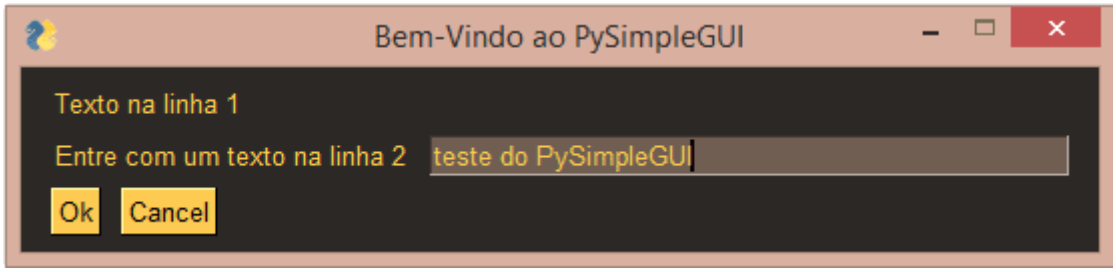
Para fazer a instalação básica, é necessário escrever na linha de comando:

```
pip install pysimplegui
```

Uma forma de testar a instalação é escrever e executar o programa:

```
import PySimpleGUI as sg
sg.theme('DarkAmber')
layout = [ [sg.Text('Texto na linha 1')],
 [sg.Text('Entre com um texto na linha 2'), sg.InputText()],
 [sg.Button('Ok'), sg.Button('Cancel')] ]
 window = sg.Window('Bem-Vindo ao PySimpleGUI', layout)
```

Se tudo funcionar corretamente, aparecerá a aplicação conforme a imagem a seguir.



Exemplo PySimpleGUI.

No exemplo da imagem anterior, é possível identificar uma janela, o seu respectivo título, dois componentes “label”, uma caixa de texto e dois botões.

Atenção!

A lista de frameworks e bibliotecas disponíveis para python ainda tem muitos outros exemplos, como o PySide e o PyObject. A escolha deve levar em consideração a maturidade da biblioteca/framework e a necessidade de o projeto incorporar ou não aspectos mais elaborados de uma GUI.