Python para Aplicações em Eletrônica

Aula 06: Interface Gráfica

Interface Gráfica (GUI)

- Permite criar formas mais amigáveis de entrada e exibição de informações.
- Diferentes bibliotecas podem ser utilizadas.
- Uma das mais comuns é a **Tkinter**.

Principais Elementos

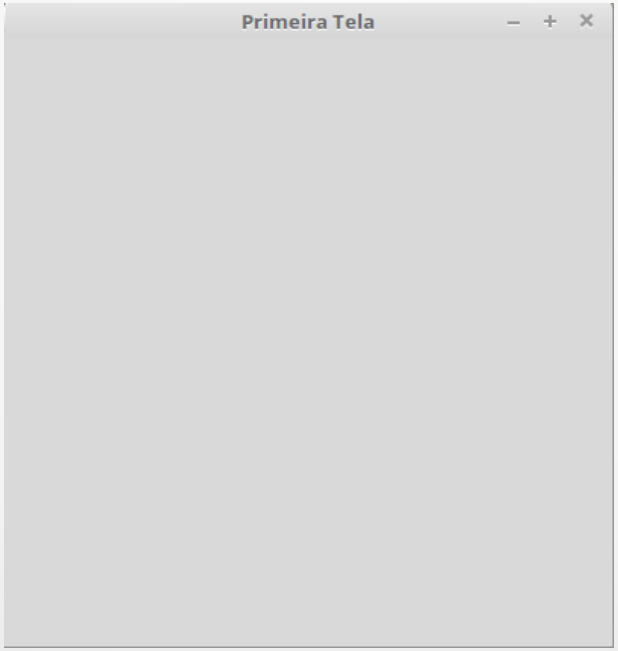
- Janelas: Base da interface gráfica. É o local onde os elementos serão exibidos.
- Containers: Facilitam a organização e agrupamento dos widgets.
- Widgets: São os elementos gerais da interface: botões, labels, etc.
- Geometria: **`pack()`**, **`grid()`** e **`place()`**.

Primeira Tela

```
1 from Tkinter import *
2
3 window = Tk()
4 window.title("Primeira Tela")
5 window.geometry("400x400")
6 window.mainloop()
```

X

```
1 from Tkinter import *
2 import time
3
4 window = Tk()
5 window.title("Primeira Tela")
6 window.geometry("400x400")
7
8 while True:
9     window.update()
10    time.sleep(1)
```



Principais *Widgets*: Uso Geral

- **`Button`**: botão
- **`Entry`**: caixa de texto editável.
- **`Label`**: caixa de texto estático.
- **`tkMessageBox`**: tela que pode ser utilizada para pedir confirmação ao usuário.

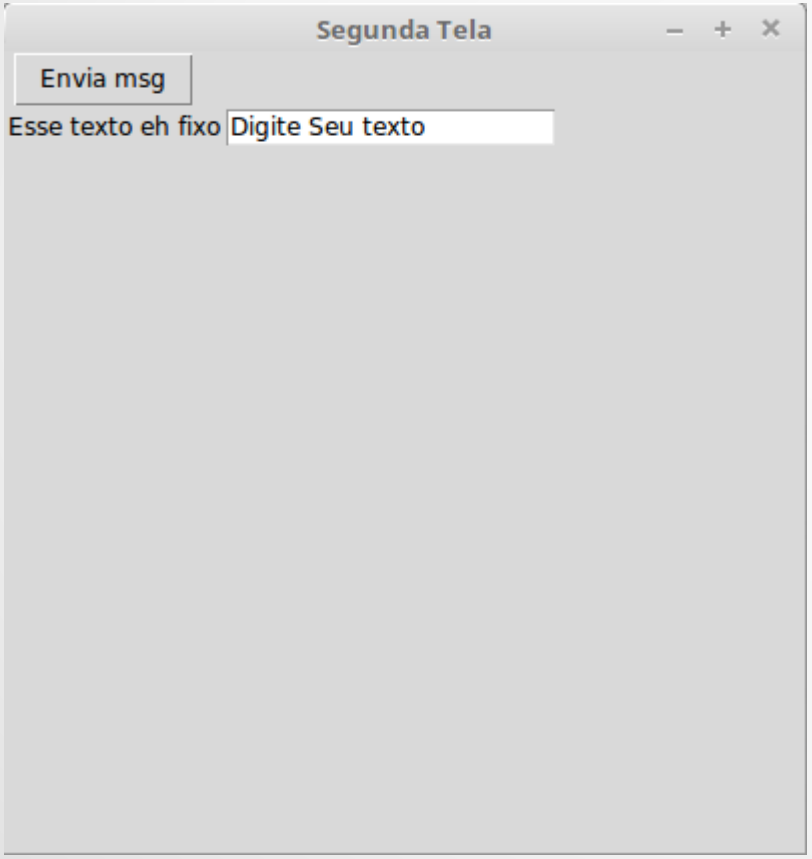
Usando a Geometria **`grid()`**

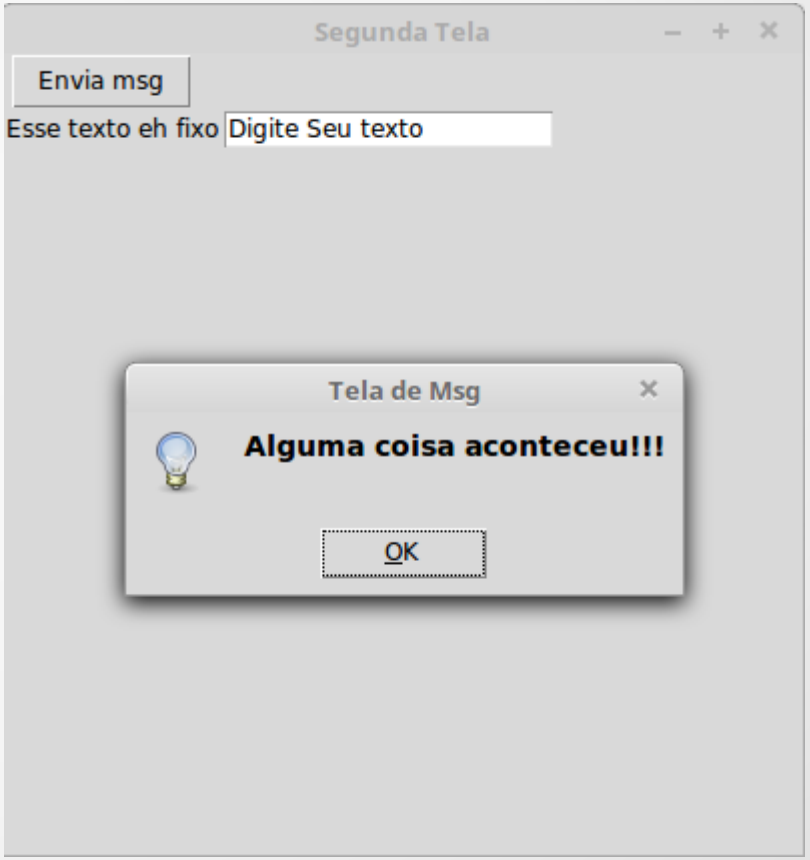
- Utilizaremos apenas a geometria do tipo *grid*.
- Define a área do *container* como uma matriz.
- Os *widgets* são adicionados em posições da matriz
- É possível configurar o *grid* para que o objeto ocupe mais de uma linha ou coluna.
- Principais atributos:
  - **`column`** e **`columnspan`**: posição (coluna) e ocupação
  - **`row`** e **`rowspan`**: posição (linha) e ocupação
  - **`sticky`**: alinhamento – N, E, S, W, NE, NW, SE, SW. O padrão é centralizado.

Segunda Tela

```
 1 from Tkinter import *
 2 import tkMessageBox
 3
 4 window = Tk()
 5 window.title("Segunda Tela")
 6 window.geometry("400x400")
 7
 8 def msg():
 9     tkMessageBox.showinfo( "Tela de Msg", "Alguma coisa aconteceu!!!")
10
11 bt1 = Button(window, text = "Envia msg", command = msg)
12 bt1.grid(row=1)
13 txt = Entry(window)
14 txt.insert(0,'Digite Seu texto')
15 txt.grid(row=2, column=3)
16
17 lb1 = Label(window, text="Esse texto eh fixo")
18 lb1.grid(row=2,columnspan=2)
19
20 window.mainloop()
21 #Nada depois desse codigo eh executado
```

Segunda Tela





Principais *Widgets*: **`Button`**

- Principais atributos:
  - **`command:`** `função ou método executado quando o botão é pressionado`
  - **`text:`** `texto do botão`
  - **`height`** `e` **`width:`** `altura e largura`
  - **`bg:`** `cor de fundo`
  - **`image:`** `imagem de fundo`
  - **`bd:`** `tamanho da borda`
- No atributo **`command`**, não pode ser passado nenhum parâmetro
  - É possível usar a “definição” **`lambda`** para lidar com isso.

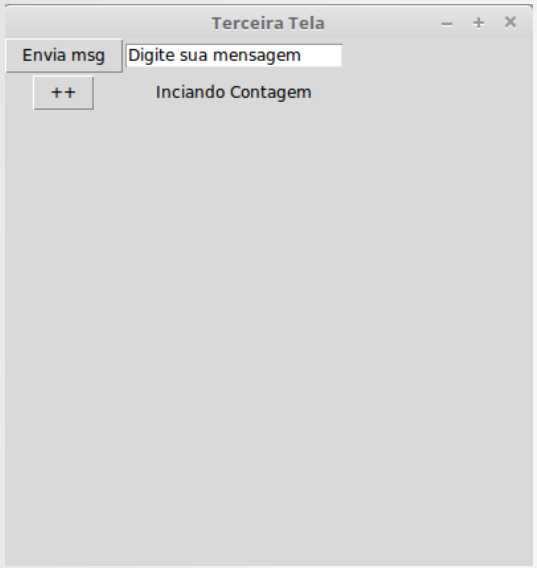
Principais *Widgets*: **Entry**

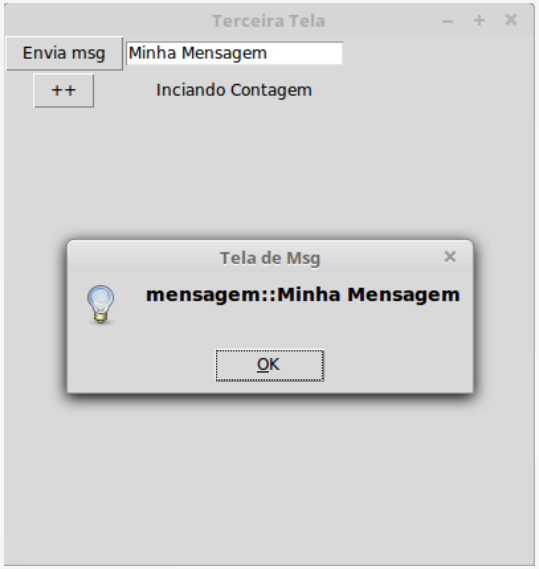
- Principais atributos:
  - **text:** texto inicial [só serve no construtor]
  - **height** e **width:** altura e largura
  - **bg:** cor de fundo
  - **bd:** tamanho da borda
- Principais métodos:
  - **get():** retorna o texto que está escrito na caixa de texto.
  - **insert(i,str):** insere a string **str** a partir do índice **i.** Serve para mudar o texto a qualquer momento.

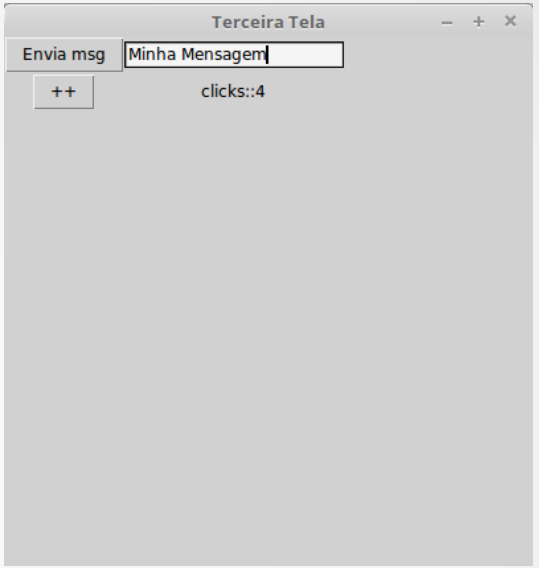
Principais *Widgets*: **Label**

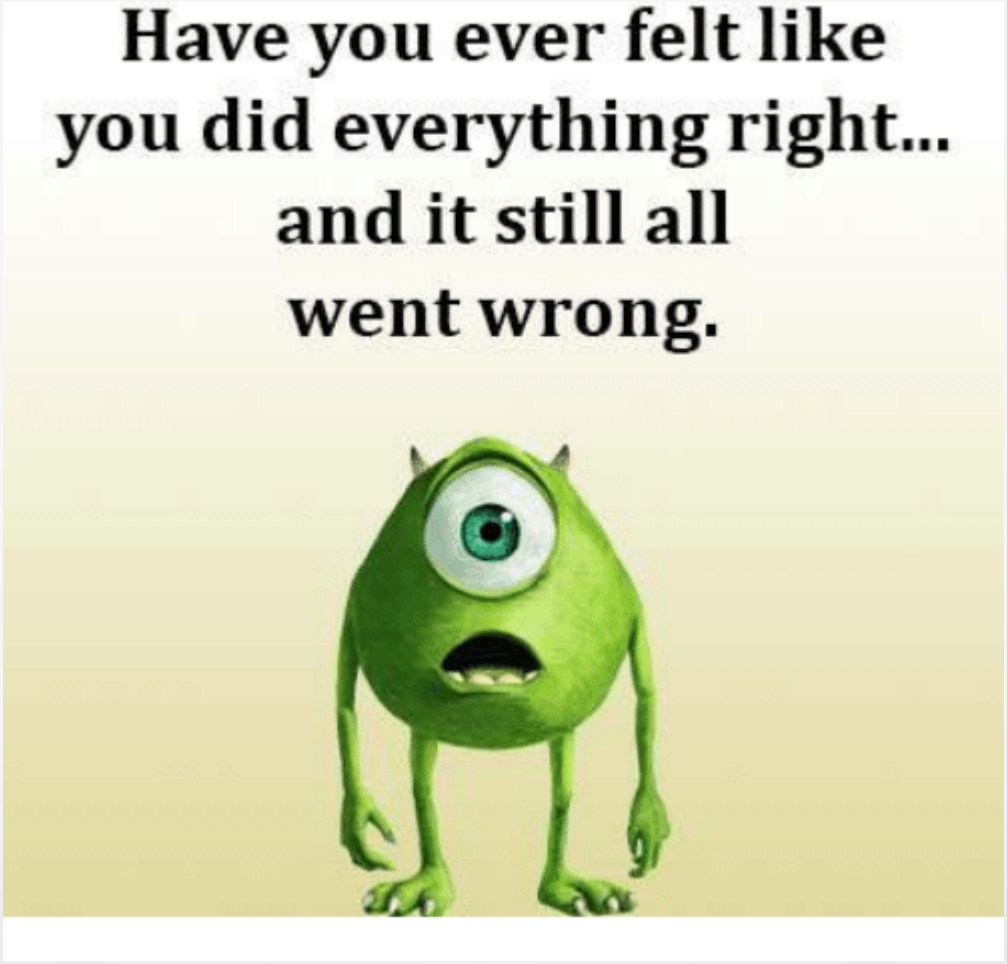
- Principais atributos:
  - **textvariable:** permite alterar o texto do botão.
  - **text:** texto do botão[pode ser usado no construtor]. Não pode ser usado se for definida uma **txtvariable.**
  - **height** e **width:** altura e largura
  - **bg:** cor de fundo
  - **bd:** tamanho da borda

Principais elementos: Uso básico








Have you ever felt like
you did everything right...
and it still all
went wrong.

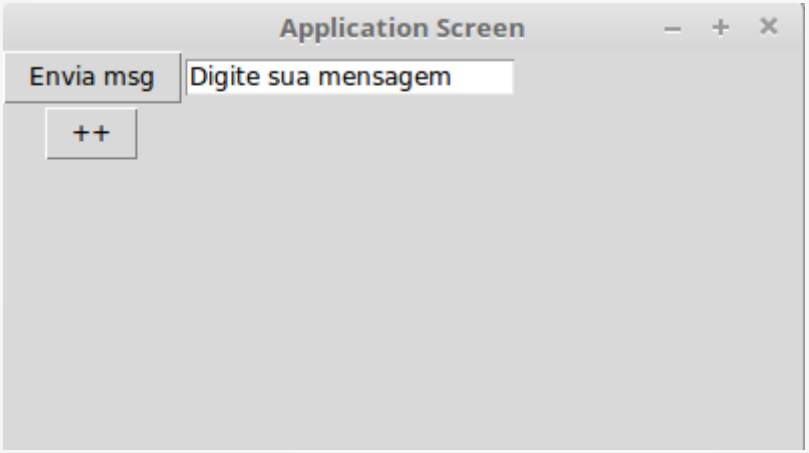
Usando Orientação a Objetos

- Sem a definição de classes, se torna mais difícil gerenciar a execução.
  - Variáveis globais precisam ser utilizadas.
  - Mais difícil gerenciar as ações de botões e outros widgets.
  - Mais difícil de gerenciar a mudança de valores.
  - Código menos estruturado e passível de erros.

Usando Orientação a Objetos

```
1 from Tkinter import *
2 import tkMessageBox
3
4 class Application:
5     def __init__(self, master):
6         self.master=master
7         self.master.title('Application Screen')
8         self.master.geometry("400x200")
9
10        # Label StringVar
11        self.txt_lbA = StringVar()
12        # Accumulator
13        self.count=0
14
15        ## Create the main frame
16        self.mainFrame = Frame(master)
17        self.mainFrame.grid() # Creates the main frame
18
19        ## Create widgets
20        self.txt = Entry(self.mainFrame)
21        self.txt.grid(row=1, column=3,columnspan=2)
22        self.txt.insert(0,'Digite sua mensagem')
23
24        # Notice that I did not create an attribute
25        Button(self.mainFrame, text = "Envia msg", command = self.msg).grid(row=1, column=0)
26        Button(self.mainFrame, text = "++", command = self.conta_click).grid(row=3,column=0)
27        Label(self.mainFrame, textvariable=self.txt_lbA).grid(row=3,column=2,columnspan=2)
28
29    def msg(self):
30        tkMessageBox.showinfo( "Tela de Msg", "mensagem::"+self.txt.get())
31
32    def conta_click(self):
33        self.count+=1
34        self.txt_lbA.set("clicks::"+str(self.count))
35
36 window = Tk()
37 myApp = Application(window)
38 window.mainloop()
39 #Nada depois desse codigo eh executado
```

Usando Orientação a Objetos



Tópicos Especiais

Separador e Gráficos

```
1 from Tkinter import *
2 import ttk ## Used to create separators
3 ## Libraries used to plot graphs
4 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
5 from matplotlib.figure import Figure
6 import random, math
```

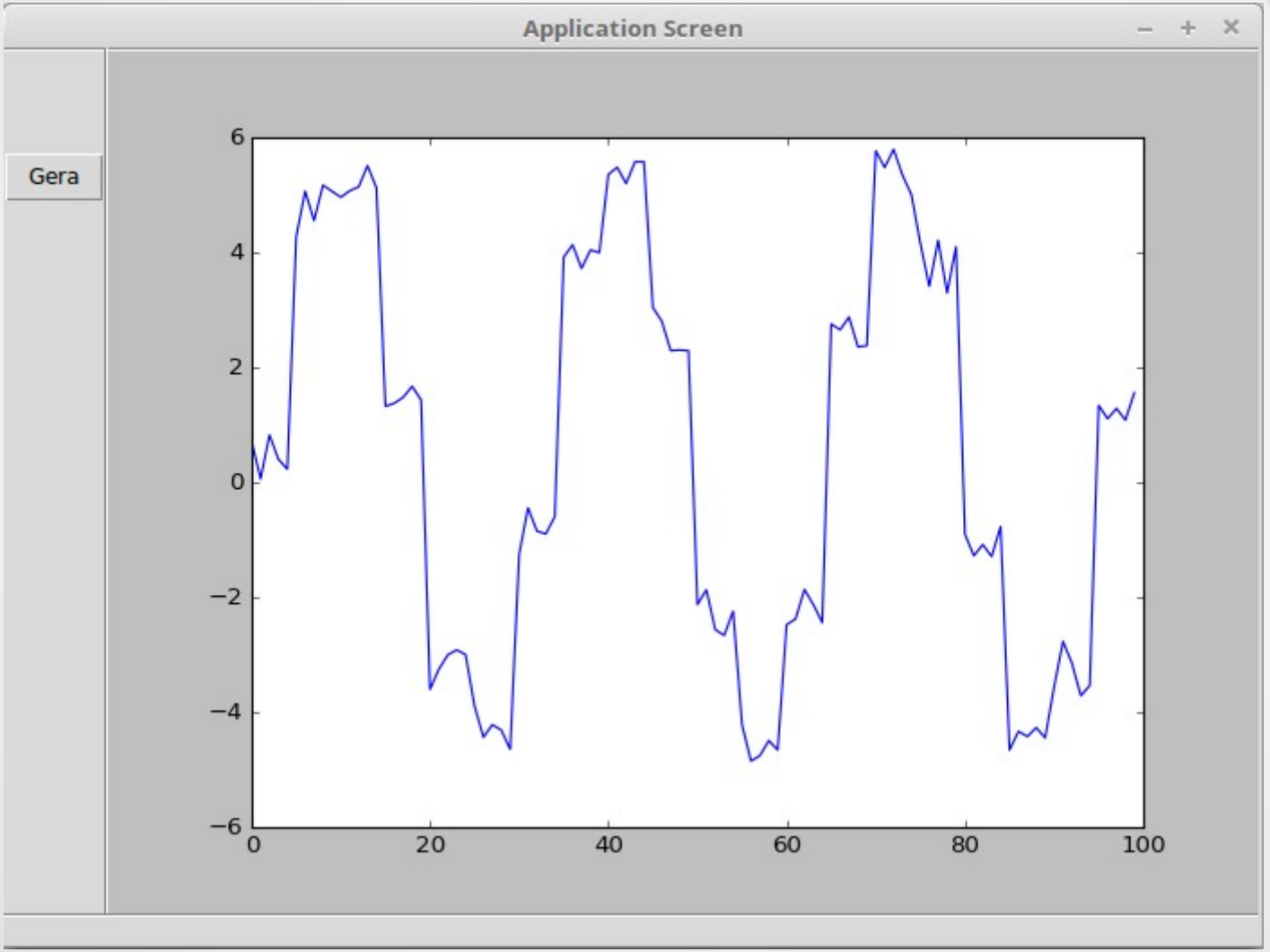
Separador e Gráficos

```
 8 class Application:
 9     def __init__(self, master):
10         self.master=master
11         self.master.title('Application Screen')
12         self.master.geometry("700x500")
13 
14         ## Create the main frame
15         self.mainFrame = Frame(master)
16         self.mainFrame.grid() # Creates the main frame
17 
18         ## Create widgets
19         Button(self.mainFrame, text = "Gera", command = self.gera_grafico).grid(row=1, column=0)
20         ttk.Separator(self.mainFrame,orient=HORIZONTAL).grid(row=0, columnspan=6,sticky="ew")
21         ttk.Separator(self.mainFrame,orient=VERTICAL).grid(row=0, column=1,rowspan=5,sticky="ns")
22         ttk.Separator(self.mainFrame,orient=HORIZONTAL).grid(row=5, columnspan=6,sticky="ew")
23 
24         ## First plot
25         self.fig = Figure()
26         self.ax = self.fig.add_subplot(111)
27         self.fig.set_size_inches(8, 6)
28         self.canvas = FigureCanvasTkAgg(self.fig,master=self.mainFrame)
29         self.canvas.get_tk_widget().grid(row=1, column=2, rowspan=4, columnspan=4,sticky=W)
30 
31 
32     def gera_grafico(self):
33         dados=[]
34         tempo=[]
35         for i in range(100):
36             tempo.append(i)
37             dados.append(5*math.sin(i/5)+random.random())
38         self.ax.cla()
39         self.ax.plot(tempo,dados)
40         self.canvas.show()
```

Separador e Gráficos

```
 1 from Tkinter import *
 2 import ttk ## Used to create separators
 3 ## Libraries used to plot graphs
 4 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
 5 from matplotlib.figure import Figure
 6 import random, math
 7
 8 class Application:
 9     def __init__(self, master):
10         self.master=master
11         self.master.title('Application Screen')
12         self.master.geometry("700x500")
13
14         ## Create the main frame
15         self.mainFrame = Frame(master)
16         self.mainFrame.grid() # Creates the main frame
17
18         ## Create widgets
19         Button(self.mainFrame, text = "Gera", command = self.gera_grafico).grid(row=1, column=0)
20         ttk.Separator(self.mainFrame,orient=HORIZONTAL).grid(row=0, columnspan=6,sticky="ew")
21         ttk.Separator(self.mainFrame,orient=VERTICAL).grid(row=0, column=1,rowspan=5,sticky="ns")
22         ttk.Separator(self.mainFrame,orient=HORIZONTAL).grid(row=5, columnspan=6,sticky="ew")
23
24         ## First plot
25         self.fig = Figure()
26         self.ax = self.fig.add_subplot(111)
27         self.fig.set_size_inches(8, 6)
28         self.canvas = FigureCanvasTkAgg(self.fig,master=self.mainFrame)
29         self.canvas.get_tk_widget().grid(row=1, column=2, rowspan=4, columnspan=4,sticky=W)
30
31
32     def gera_grafico(self):
33         dados=[]
34         tempo=[]
35         for i in range(100):
36             tempo.append(i)
37             dados.append(5*math.sin(i/5)+random.random())
38         self.ax.cla()
39         self.ax.plot(tempo,dados)
40         self.canvas.show()
41
42 window = Tk()
43 myApp = Application(window)
44 window.mainloop()
45 #Nada depois desse codigo eh executado
```

Separador e Gráficos



Exercícios