Computação II

Departamento de Ciência da Computação – UFRJ

Interface Gráfica – Parte 2

Professora: *Fernanda Duarte Vilela Reis de Oliveira*

2019/01

1. Interfaces gráficas - Tkinter

- Em uma aplicação com interface gráfica, a interação com o usuário é feita através de eventos:
  - Ao clicar com o mouse ou apertar uma tecla do teclado é gerado um evento.
- Métodos e funções (callbacks) podem ser associados aos eventos através da função bind (binding):
  - widget.bind(evento, função) , onde widget deve ser substituído pelo nome do objeto onde o evento deve ser detectado (Tk, Frame, Canvas...), evento é o nome do tipo de evento detectado (“<Right>”, “a”, “<Button-1>”), e função o nome da função que será executata quando o evento ocorrer.

1. Interfaces gráficas - Tkinter

- Exemplos:

```
root.bind('<Key>',tecla)
root.bind('<Left>',tecla)
root.bind('<Button-3>',mouse)
root.bind('<Enter>',mouse)
frame.bind('<Return>',tecla)
frame.bind('a',tecla)
frame.bind('<Button-1>',mouse)
frame.bind('<ButtonRelease-1>',mouse)
```

- A função “focus_set” diz ao Tkinter qual é a widget que está associada ao teclado (uma janela Tk, um frame, um canvas)... Sem a focus_set o teclato fica associado à janela principal.

```
frame.focus_set()
```

1. Interfaces gráficas - Tkinter

- Exemplos:

```
root.bind('<Key>',tecla)
root.bind('<Left>',tecla)
root.bind('<Button-3>',mouse)
root.bind('<Enter>',mouse)
```

Associando os eventos à instância root de Tk().

```
frame.bind('<Return>',tecla)
frame.bind('a',tecla)
frame.bind('<Button-1>',mouse)
frame.bind('<ButtonRelease-1>',mouse)
```

Associando os eventos à instância frame de Frame().

A função bind envia à função chamada um objeto com informações do evento gerado (slide 24).

- A função "focus_set" diz ao Tkinter qual é a widget que está associada ao teclado (uma janela Tk, um frame, um canvas)... Sem a focus_set o teclato fica associado à janela principal.

```
frame.focus_set()
```

1. Interfaces gráficas - Tkinter

- Exemplos:

Um evento é gerado quando:

```
root.bind('<Key>',tecla)
```
Qualquer tecla do teclado for pressionada.

```
root.bind('<Left>',tecla)
```
A seta para a esquerda for pressionada.

```
root.bind('<Button-3>',mouse)
```
O botão direito no mouse for clicado.

```
root.bind('<Enter>',mouse)
```
O cursor do mouse entrar na widget associada.

```
frame.bind('<Return>',tecla)
```
Quando o Enter do teclado for pressionado.

```
frame.bind('a',tecla)
```
A tecla “a” (minúsculo) for pressionada.

```
frame.bind('<Button-1>',mouse)
```
O botão esquerdo do mouse for clicado.

```
frame.bind('<ButtonRelease-1>',mouse)
```
O botão esquerdo do mouse for liberado.

- A função “focus_set” diz ao Tkinter qual é a widget que está associada ao teclado (uma janela Tk, um frame, um canvas)... Sem a focus_set o teclato fica associado à janela principal.

```
frame.focus_set()
```

1. Interfaces gráficas - Tkinter

- Exemplos:

```
root.bind('<Key>',tecla)
root.bind('<Left>',tecla)
root.bind('<Button-3>',mouse)
root.bind('<Enter>',mouse)
frame.bind('<Return>',tecla)
frame.bind('a',tecla)
frame.bind('<Button-1>',mouse)
frame.bind('<ButtonRelease-1>',mouse)
```

Nome da função chamada quando o evento é gerado – essa função recebe como argumento um objeto com informações do evento gerado. Para eventos do teclado: char e keysym (nome simbólico da tecla). Para eventos do mouse: num (número do botão clicado do mouse). Qualquer tipo de evento: x e y (posição do mouse na widget no instante do evento), x_root e y_root (), widget.

- A função “focus_set” diz ao Tkinter qual é a widget que está associada ao teclado (uma janela Tk, um frame, um canvas)... Sem a focus_set o teclato fica associado à janela principal.

```
frame.focus_set()
```

1. Interfaces gráficas - Tkinter

- Exemplos:

```
root.bind('<Key>',tecla)
root.bind('<Left>',tecla)
root.bind('<Button-3>',mouse)
root.bind('<Enter>',mouse)
frame.bind('<Return>',tecla)
frame.bind('a',tecla)
frame.bind('<Button-1>',mouse)
frame.bind('<ButtonRelease-1>',mouse)
```

- A função “focus_set” diz ao Tkinter qual é a widget que está associada ao teclado (uma janela Tk, um frame, um canvas)... Sem a focus_set o teclato fica associado à janela principal.

```
frame.focus_set()
```

Indicando que o teclado deve ser capturado na instância frame: frame.bind só passa a funcionar após o frame.focus_set() .

1. Interfaces gráficas - Tkinter

- Pegando a posição do mouse e a tecla que foi digitada na janela root:

```
from tkinter import *

def mouse(evento):
    print("Posicao: ", evento.x, evento.y)

def tecla(evento):
    print("Tecla", repr(evento.char))

root = Tk()

root.bind("<Key>", tecla)
root.bind("<Button-1>", mouse)

root.mainloop()
```

1. Interfaces gráficas - Tkinter

- Pegando a posição do mouse e a tecla que foi digitada na janela root:

```
from tkinter import *

def mouse(evento):
    print("Posicao: ", evento.x, evento.y)

def tecla(evento):
    print("Tecla", repr(evento.char))
```

```
root = Tk()
```
Cria uma janela principal.

```
root.bind("<Key>", tecla)
```
Para qualquer tecla, chame a função “tecla”.

```
root.bind("<Button-1>", mouse)
```
Para um cliquer do botão esquerdo do mouse, chame a função “mouse”.

```
root.mainloop()
```
Loop da interface.

Como o bind está sendo feito somente na janela principal o focus_set não é necessário.

1. Interfaces gráficas - Tkinter

- Pegando a posição do mouse e a tecla que foi digitada na janela root:

```
from tkinter import *

def mouse(evento):
    print("Posicao: ", evento.x, evento.y)

def tecla(evento):
    print("Tecla", repr(evento.char))

root = Tk()

root.bind("<Key>", tecla)
root.bind("<Button-1>", mouse)

root.mainloop()
```

Funções chamadas recebem um objeto com informações do evento.

1. Interfaces gráficas - Tkinter

- Pegando a posição do mouse e a tecla que foi digitada na janela root:

```
from tkinter import *

def mouse(evento):
    print("Posicao: ", evento.x, evento.y)

def tecla(evento):
    print("Tecla", repr(evento.char))

root = Tk()

root.bind("<Key>", tecla)
root.bind("<Button-1>", mouse)

root.mainloop()
```

Posição do mouse no instante do clique.

Caractere que foi teclado.

1. Interfaces gráficas - Tkinter

- Pegando a posição do mouse e a tecla que foi digitada:

```
from tkinter import *

def mouse(event):
    frame.focus_set()
    print("Posicao: ", event.x, event.y)

def tecla(event):
    print("Tecla", repr(event.char))

root = Tk()

frame = Frame(root, width=400, height=400)
frame.bind("<Key>", tecla)
frame.bind("<Button-1>", mouse)
frame.pack()

root.mainloop()
```

1. Interfaces gráficas - Tkinter

- Pegando a posição do mouse e a tecla que foi digitada:

```
from tkinter import *

def mouse(event):
    frame.focus_set()
    print("Posicao: ", event.x, event.y)

def tecla(event):
    print("Tecla", repr(event.char))

root = Tk()

frame = Frame(root, width=400, height=400)
frame.bind("<Key>", tecla)
frame.bind("<Button-1>", mouse)
frame.pack()

root.mainloop()
```

Mesmo objetivo do código anterior, mas agora é criado um frame dentro da janela.
Uma vez que o bind é feito no frame, o focus_set se torna necessário.

1. Interfaces gráficas - Tkinter

- Pegando a posição do mouse e a tecla que foi digitada:

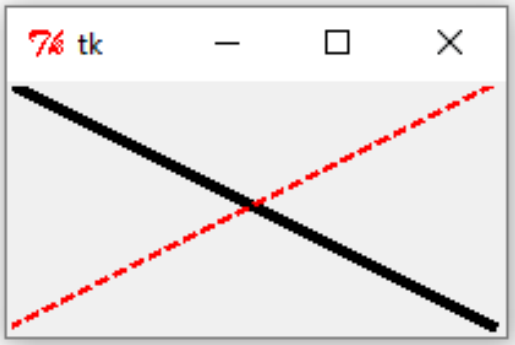
```
from tkinter import *

def mouse(event):
    frame.focus_set()
    print("Posicao: ", event.x, event.y)

def tecla(event):
    print("Tecla", repr(event.char))

root = Tk()

frame = Frame(root, width=400, height=400)
frame.bind("<Key>", tecla)
frame.bind("<Button-1>", mouse)
frame.pack()

root.mainloop()
```

Como o focus_set está dentro da função chamada pelo clique do mouse, os eventos do teclado só são detectados após o primeiro clique do mouse.

1. Interfaces gráficas - Tkinter

- Canvas: utilizado para desenhos, gráficos, criar editores de gráficos e, por causa da versatilidade, permite implementar outras widgets.
- Métodos create adicionam novos itens:

```
from tkinter import *

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

janela.create_line(0, 0, 200, 100, width=4.0)
janela.create_line(0, 100, 200, 0, fill="red",\
                   dash=(15, 15), width=2.0)

root.mainloop()
```

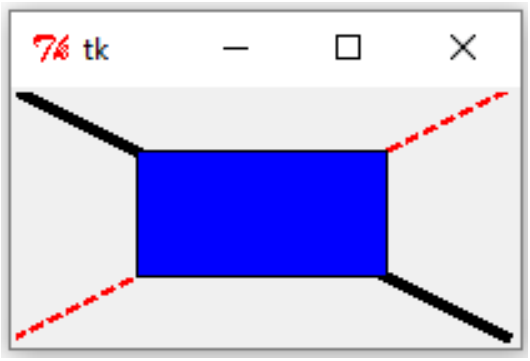


\Exemplos\ex12CanvasCreateLine_Python3.py

1. Interfaces gráficas - Tkinter

- Canvas: utilizado para desenhos, gráficos, criar editores de gráficos e, por causa da versatilidade, permite implementar outras widgets.
- Métodos create adicionam novos itens:

```
from tkinter import *

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

janela.create_line(0, 0, 200, 100, width=4.0)
janela.create_line(0, 100, 200, 0, fill="red",\
                   dash=(15, 15), width=2.0)
retang = janela.create_rectangle(50, 25, 150, 75, fill="blue")
root.mainloop()
```



\Exemplos\ex13CanvasCreateRectangle_Python3.py

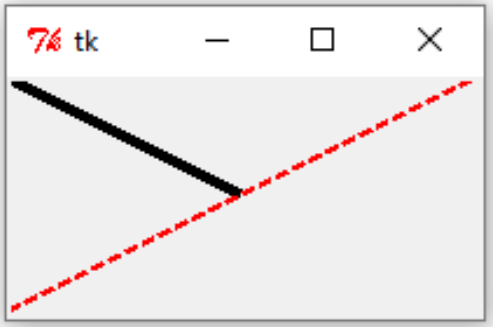
1. Interfaces gráficas - Tkinter

- Método coords - recebe um identificador do objeto que será modificado e, opcionalmente, um conjunto de coordenadas:
  - Se as coordenadas não forem dadas, coords retorna as posições que compõe o objeto.
  - Se forem dados pares de coordenadas proporcionais à posição original do objeto, o objeto é movido.
  - Se forem dados os pares originais e mais pares, o objeto do tipo linha é estendido.

1. Interfaces gráficas - Tkinter

- Coordenadas de duas linhas criadas:

```
from tkinter import *

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)
print(janela.coords(linhaPreta))
print(janela.coords(linhaVerm))

root.mainloop()
```



Interface criada ao rodar o programa.

Criando uma linha vermelha e outra preta na interface.

\Exemplos\ex14CanvasLinesCoords_Python3.py

1. Interfaces gráficas - Tkinter

- Coordenadas de duas linhas criadas:

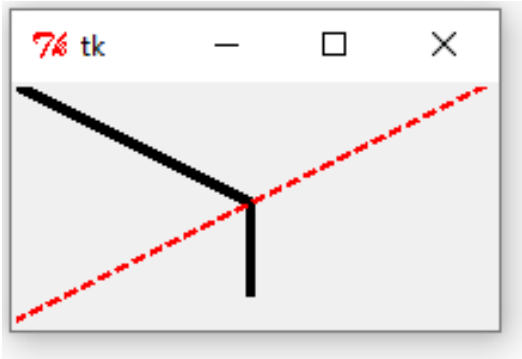
```
from tkinter import *

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)
print(janela.coords(linhaPreta))
print(janela.coords(linhaVerm))

root.mainloop()
```

```
[0.0, 0.0, 100.0, 50.0]
[0.0, 100.0, 200.0, 0.0]
```

\Exemplos\ex14CanvasLinesCoords_Python3.py

1. Interfaces gráficas - Tkinter

- Coordenadas de duas linhas criadas:




```
from tkinter import *

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, 100, 90, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)
print(janela.coords(linhaPreta))
print(janela.coords(linhaVerm))

root.mainloop()
```

Três coordenadas definem dois segmentos de reta.

\Exemplos\ex15CanvasLinesCoordsSegs_Python3.py

1. Interfaces gráficas - Tkinter

- Coordenadas de duas linhas criadas:

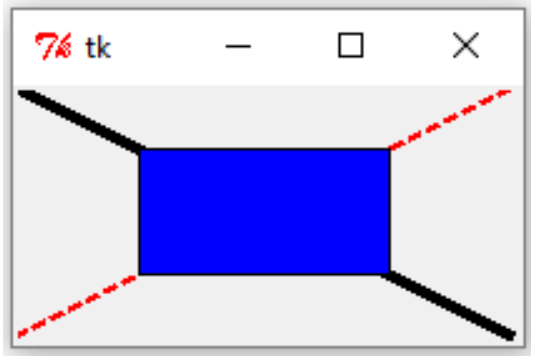
```
from tkinter import *

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, 100, 90, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)
print(janela.coords(linhaPreta))
print(janela.coords(linhaVerm))

root.mainloop()
```

```
[0.0, 0.0, 100.0, 50.0, 100.0, 90.0]
```

\Exemplos\ex15CanvasLinesCoordsSegs_Python3.py

1. Interfaces gráficas - Tkinter

- Coordenadas do retângulo:




```
from tkinter import *

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

janela.create_line(0, 0, 200, 100, width=4.0)
janela.create_line(0, 100, 200, 0, fill="red",\
                   dash=(15, 15), width=2.0)
retang = janela.create_rectangle(50, 25, 150, 75, fill="blue")

print(janela.coords(retang))

root.mainloop()
```

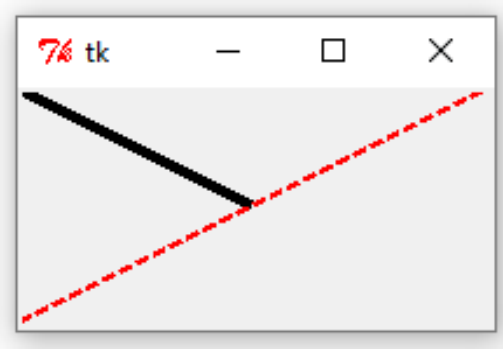
```
[50.0, 25.0, 150.0, 75.0]
```

\Exemplos\ex16CanvasRectangleCoords_Python3.py

1. Interfaces gráficas - Tkinter

- Movendo uma linha com coords:

```
from tkinter import *

def moverLinha(evento):
    janela.coords(linhaPreta,100,50,200,100)


root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                  dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

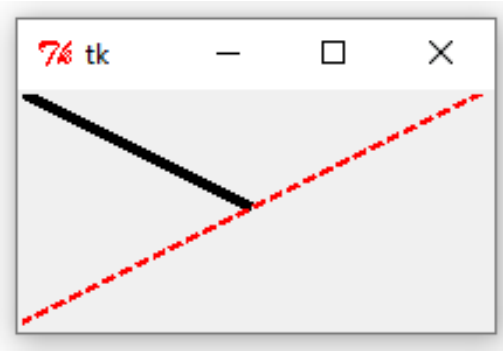


\Exemplos\ex17CanvasMovendoLinhaComCoords_Python3.py

1. Interfaces gráficas - Tkinter

- Movendo uma linha com coords:

```
from tkinter import *

def moverLinha(evento):
    janela.coords(linhaPreta,100,50,200,100)


root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```



Interface criada ao rodar o programa.

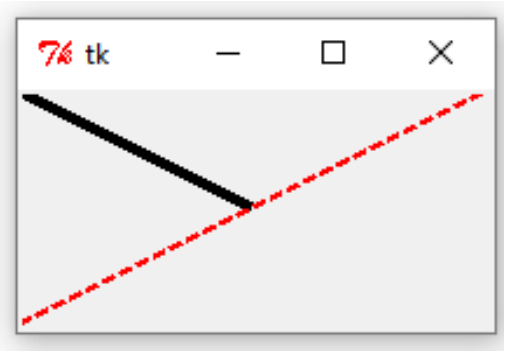
Criando uma linha vermelha e outra preta na interface.

\Exemplos\ex17CanvasMovendoLinhaComCoords_Python3.py

1. Interfaces gráficas - Tkinter

- Movendo uma linha com coords:

```
from tkinter import *

def moverLinha(evento):
    janela.coords(linhaPreta,100,50,200,100)


root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                   dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```



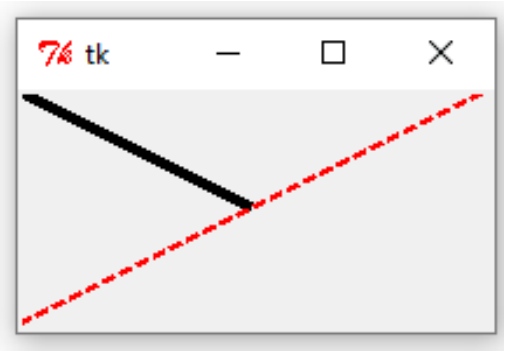
Interface criada ao rodar o programa.

Métodos create retornam um número inteiro que identifica o item criado (id). Esse número pode ser usado para modificar o objeto. A cada item criado um identificador diferente é gerado.

1. Interfaces gráficas - Tkinter

- Movendo uma linha com coords:

```
from tkinter import *

def moverLinha(evento):
    janela.coords(linhaPreta,100,50,200,100)


root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```



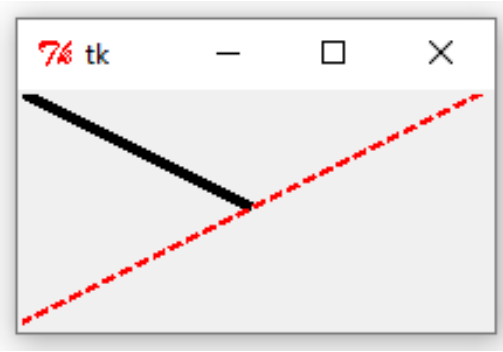
Interface criada ao rodar o programa.

A variável linhaPreta assume o valor do identificador da primeira linha (1) e a variável linhaVerm assume o valor do identificador da segunda linha (2).

1. Interfaces gráficas - Tkinter

- Movendo uma linha com coords:

```
from tkinter import *

def moverLinha(evento):
    janela.coords(linhaPreta,100,50,200,100)


root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                  dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```



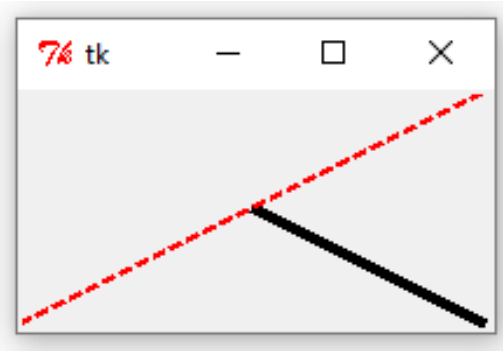
Associando o evento do clique do botão esquerdo do mouse à função moverLinha

1. Interfaces gráficas - Tkinter

- Movendo uma linha com coords:

```
from tkinter import *

def moverLinha(evento):
    janela.coords(linhaPreta,100,50,200,100)

root = Tk()
janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

Ao clicar com o botão esquerdo, a linha preta é movida utilizando o método coords, pois a variável linhaPreta, que contém o identificador do item é utilizada.



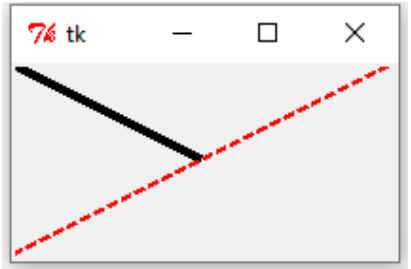
\Exemplos\ex17CanvasMovendoLinhaComCoords_Python3.py

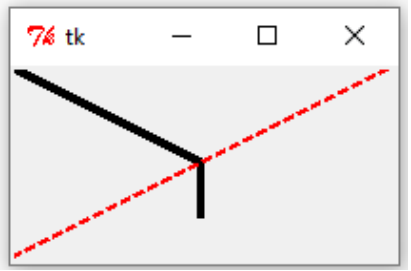
1. Interfaces gráficas - Tkinter

- Modificando a linha:

```
from tkinter import *

def moverLinha(evento):
    janela.coords(linhaPreta,0, 0, 100, 50,100,80)

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

Ao clicar com o botão esquerdo, um par de coordenadas é acrescentado aos que já existiam (0, 0, 100, 50), criando um novo segmento de reta.





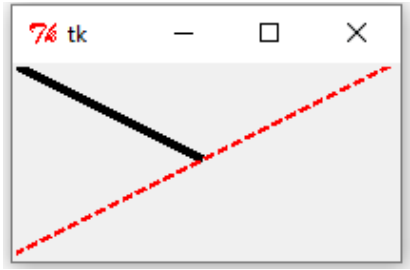
ex18CanvasModificandoLinhaComCoords_Python3

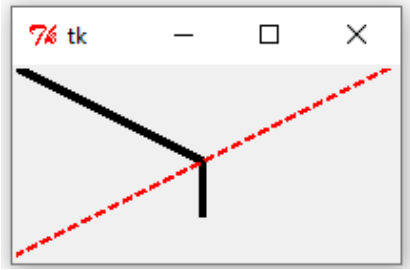
1. Interfaces gráficas - Tkinter

- Modificando a linha:

```
from tkinter import *

def moverLinha(evento):
    coordenadas = janela.coords(linhaPreta)
    coordenadas = coordenadas + [100,80]
    janela.coords(linhaPreta,*coordenadas)

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

Podemos usar coords para pegar as coordenadas que já estavam associadas à linha.





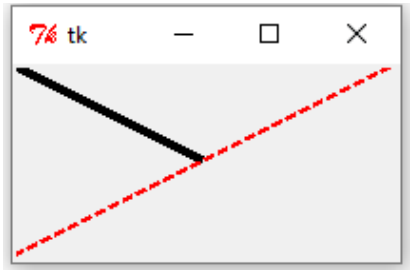
ex19CanvasModificandoLinhaComCoords2_Python3

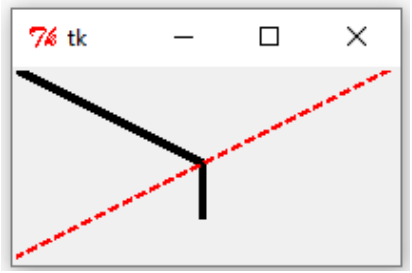
1. Interfaces gráficas - Tkinter

- Modificando a linha:

```
from tkinter import *

def moverLinha(evento):
    coordenadas = janela.coords(linhaPreta)
    coordenadas = coordenadas + [100,80]
    janela.coords(linhaPreta,*coordenadas)

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

O resultado é um vetor de coordenadas que pode ser concatenado com outro vetor de coordenadas.





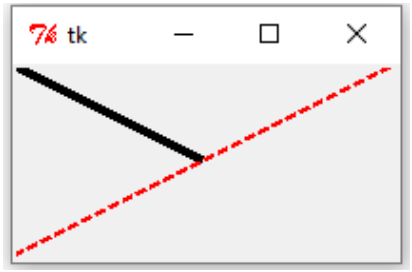
ex19CanvasModificandoLinhaComCoords2_Python3

1. Interfaces gráficas - Tkinter

- Modificando a linha:

```
from tkinter import *

def moverLinha(evento):
    coordenadas = janela.coords(linhaPreta)
    coordenadas = coordenadas + [100,80]
    janela.coords(linhaPreta,*coordenadas)

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

linhaPreta = janela.create_line(0, 0, 100, 50, width=4.0)
linhaVerm = janela.create_line(0, 100, 200, 0, fill="red",\
                    dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

O vetor deve ser transformado em um conjunto de pontos e isso é feito com o asterisco.



ex19CanvasModificandoLinhaComCoords2_Python3

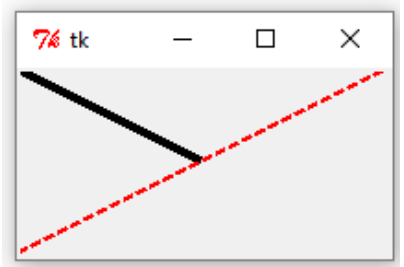
1. Interfaces gráficas - Tkinter

- Outra possibilidade, associando a linha a um tag:

```
from tkinter import *

def moverLinha(evento):
    janela.coords('linhaPreta',100,50,200,100)


root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

janela.create_line(0, 0, 100, 50, width=4.0,tags='linhaPreta')
janela.create_line(0, 100, 200, 0, fill="red",\
                   dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

1. Interfaces gráficas - Tkinter

- Outra possibilidade, associando a linha a um tag:

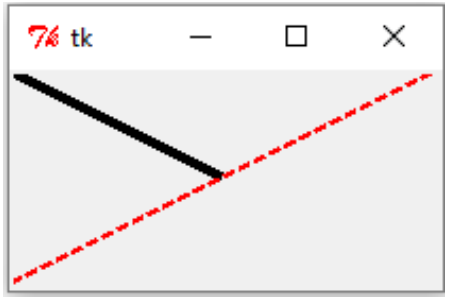



```
from tkinter import *

def moverLinha(evento):
    janela.coords('linhaPreta',100,50,200,100)


root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()

janela.create_line(0, 0, 100, 50, width=4.0,tags='linhaPreta')
janela.create_line(0, 100, 200, 0, fill="red",\
                   dash=(15, 15), width=2.0)

janela.bind("<Button-1>", moverLinha)

root.mainloop()
```

1. Interfaces gráficas - Tkinter

- Mais de um item pode estar associado a um mesmo tag:

```
from tkinter import *

def moverLinha(evento):
    janela.coords('linha',0, 0, 100, 50,100,80)
    print(janela.find_withtag('linha'))

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()
janela.create_line(0, 0, 100, 50, width=4.0, tags='linha')
print(janela.find_withtag('linha'))
janela.create_line(0, 100, 200, 0, fill='red',\
                   dash=(15, 15), width=2.0,\
                   tags='linha')
print(janela.find_withtag('linha'))

janela.bind('<Button-1>', moverLinha)

root.mainloop()
```

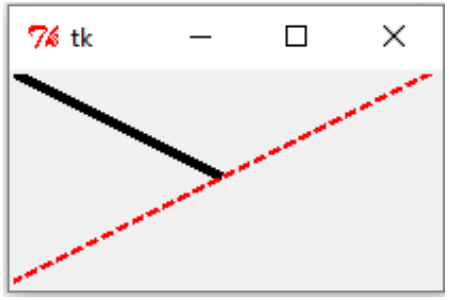


As duas linhas criadas possuem o mesmo tag, mas cada linha possui um id diferente.

ex21CanvasCoordsDuasLinhasMesmoTag_Python3.py

1. Interfaces gráficas - Tkinter

- Mais de um item pode estar associado a um mesmo tag:




```
from tkinter import *

def moverLinha(evento):
    janela.coords('linha',0, 0, 100, 50,100,80)
    print(janela.find_withtag('linha'))

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()
janela.create_line(0, 0, 100, 50, width=4.0,tags='linha')
print(janela.find_withtag('linha'))
janela.create_line(0, 100, 200, 0, fill='red',\
                   dash=(15, 15), width=2.0,\
                   tags='linha')
print(janela.find_withtag('linha'))

janela.bind('<Button-1>', moverLinha)

root.mainloop()
```

Retorna o id dos items associados ao tag 'linha'.

Da primeira vez, só um id é associado ao tag.

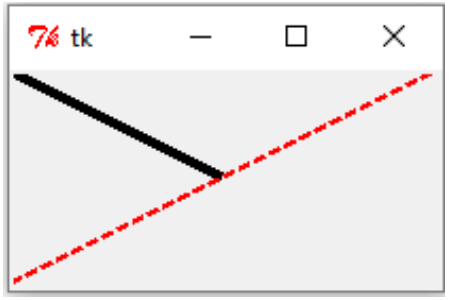
Da segunda vez que a função é chamada, dois ids estão associados ao tag.

ex21CanvasCoordsDuasLinhasMesmoTag_Python3.py

1. Interfaces gráficas - Tkinter

- Mais de um item pode estar associado a um mesmo tag:

```
from tkinter import *

def moverLinha(evento):
    janela.coords('linha',0, 0, 100, 50,100,80)
    print(janela.find_withtag('linha'))

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()
janela.create_line(0, 0, 100, 50, width=4.0,tags='linha')
print(janela.find_withtag('linha'))
janela.create_line(0, 100, 200, 0, fill='red',\
                   dash=(15, 15), width=2.0,\
                   tags='linha')
print(janela.find_withtag('linha'))

janela.bind('<Button-1>', moverLinha)

root.mainloop()
```



(1,) Id do primeiro item criado e associado ao tag.

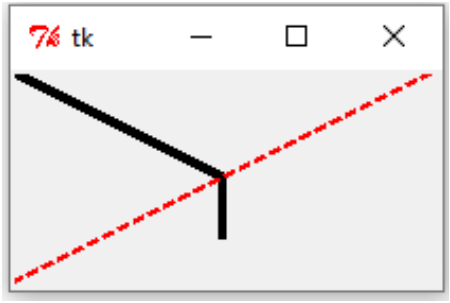
(1, 2) Id dos dois itens criados e associados ao mesmo tag.

ex21CanvasCoordsDuasLinhasMesmoTag_Python3.py

1. Interfaces gráficas - Tkinter

- Mais de um item pode estar associado a um mesmo tag:

As coordenadas do item de menor id são modificadas.




```
from tkinter import *

def moverLinha(evento):
    janela.coords('linha',0, 0, 100, 50,100,80)
    print(janela.find_withtag('linha'))

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()
janela.create_line(0, 0, 100, 50, width=4.0,tags='linha')
print(janela.find_withtag('linha'))
janela.create_line(0, 100, 200, 0, fill='red',\
                   dash=(15, 15), width=2.0,\
                   tags='linha')
print(janela.find_withtag('linha'))

janela.bind('<Button-1>', moverLinha)

root.mainloop()
```

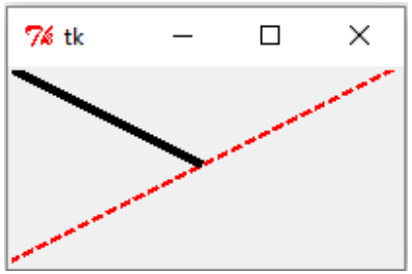
ex21CanvasCoordsDuasLinhasMesmoTag_Python3.py

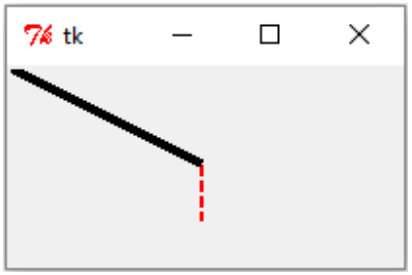
1. Interfaces gráficas - Tkinter

- O primeiro item criado possui menor id:

Mudando a ordem de criação das linhas, o id da linha vermelha é menor que o da linha preta, de forma que o item modificado é a linha vermelha.

```
from tkinter import *

def moverLinha(evento):
    janela.coords('linha',0, 0, 100, 50,100,80)

root = Tk()

janela = Canvas(root, width=200, height=100)
janela.pack()
janela.create_line(0, 100, 200, 0, fill='red',\
                   dash=(15, 15), width=2.0,\
                   tags='linha')
janela.create_line(0, 0, 100, 50, width=4.0,tags='linha')

janela.bind('<Button-1>', moverLinha)

root.mainloop()
```





1. Interfaces gráficas - Tkinter

- Utilizando o canvas para desenho:

```
from tkinter import *
def novalinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    janela.create_line(x,y,x,y,tags="linha")
def estenderlinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    coords = janela.coords("linha") + [x,y]
    janela.coords("linha",*coords)
root = Tk()
janela = Canvas(root,width=512, height=512)
janela.pack()
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
root.mainloop()
```

\Exemplos\ex23CanvasDesenharNovaEstender_Python3.py

1. Interfaces gráficas - Tkinter

- Utilizando o canvas para desenho:

```
from tkinter import *
def novalinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    janela.create_line(x,y,x,y,tags="linha")
def estenderlinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    coords = janela.coords("linha") + [x,y]
    janela.coords("linha",*coords)
root = Tk()
janela = Canvas(root,width=512, height=512)
janela.pack()
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
root.mainloop()
```

Associando o evento de clique do botão esquerdo e o evento de movimento do mouse com o botão clicado a duas funções diferentes.

1. Interfaces gráficas - Tkinter

- Utilizando o canvas para desenho:

```
from tkinter import *
def novalinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    janela.create_line(x,y,x,y,tags="linha")
def estenderlinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    coords = janela.coords("linha") + [x,y]
    janela.coords("linha",*coords)
root = Tk()
janela = Canvas(root,width=512, height=512)
janela.pack()
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
root.mainloop()
```

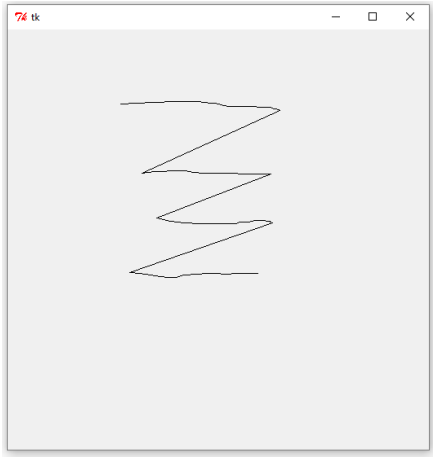
Cada vez que o botão é clicado uma nova linha é criada na posição do clique e associada a um mesmo tag. Como as posições de início e final da linha são as mesmas, cada nova linha corresponde a somente um ponto que não aparece na tela.

\Exemplos\ex23CanvasDesenharNovaEstender_Python3.py

1. Interfaces gráficas - Tkinter

- Utilizando o canvas para desenho:

```
from tkinter import *
def novalinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    janela.create_line(x,y,x,y,tags="linha")
def estenderlinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    coords = janela.coords("linha") + [x,y]
    janela.coords("linha",*coords)
root = Tk()
janela = Canvas(root,width=512, height=512)
janela.pack()
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
root.mainloop()
```

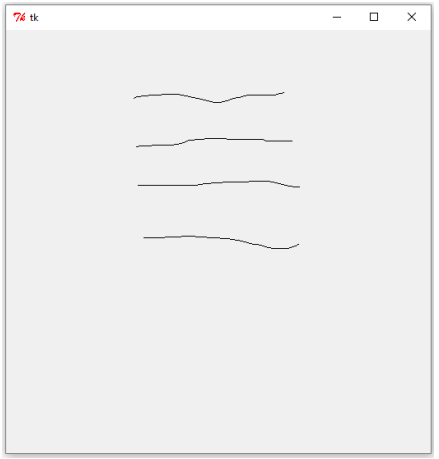


Movimentando o mouse com o botão clicado, a linha criada é modificada, pois novas coordenadas são inseridas. Somente a linha com menor id é modificada, de forma que todo o desenho estará conectado à mesma linha.

\Exemplos\ex23CanvasDesenharNovaEstender_Python3.py

1. Interfaces gráficas - Tkinter

- Para criar linhas separadas:



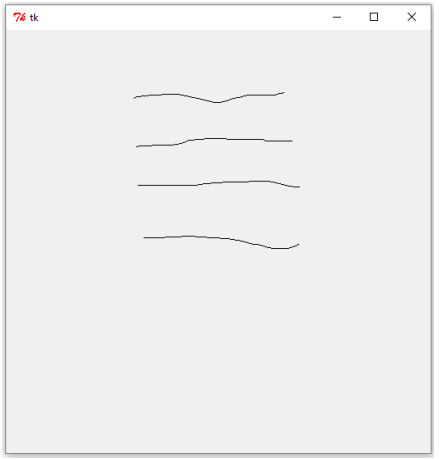

```
from tkinter import *
def novalinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    janela.create_line(x,y,x,y,tags="linha")
def estenderlinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    coords = janela.coords("linha") + [x,y]
    janela.coords("linha",*coords)
def fecharlinha(evento):
    janela.itemconfig("linha",tags=())
root = Tk()
janela = Canvas(root,width=512, height=512)
janela.pack()
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
janela.bind("<ButtonRelease-1>", fecharlinha)
root.mainloop()
```

O método itemconfig pode ser utilizado para modificar as opções de um item, ele recebe o id ou tag do(s) item(ns) que desejamos modificar e as opções que serão modificadas.

\Exemplos\ex24CanvasDesenharNovaEstenderFechar_Python3.py

1. Interfaces gráficas - Tkinter

- Para criar linhas separadas:




```
from tkinter import *
def novalinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    janela.create_line(x,y,x,y,tags="linha")
def estenderlinha(evento):
    x = janela.canvasx(evento.x)
    y = janela.canvasy(evento.y)
    coords = janela.coords("linha") + [x,y]
    janela.coords("linha",*coords)
def fecharlinha(evento):
    janela.itemconfig("linha",tags=())
root = Tk()
janela = Canvas(root,width=512, height=512)
janela.pack()
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
janela.bind("<ButtonRelease-1>", fecharlinha)
root.mainloop()
```

O evento de soltar o botão é associado a uma função que utiliza itemconfig para modificar as tags dos itens 'linha' para um tag vazio.
Dessa forma, quando um novo houver um novo clique, somente o novo item estará associado ao tag 'linha' e o desenho permite traçados separados.

\Exemplos\ex24CanvasDesenharNovaEstenderFechar_Python3.py

1. Interfaces gráficas - Tkinter

- A tag CURRENT equivale ao item sobre o ponteiro do mouse:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    coords = janela.coords("sel")
    for i in range(len(coords)):
        if i%2 == 0:
            coords[i] = coords[i] + x1-x0
        else:
            coords[i] = coords[i] + y1-y0
    janela.coords("sel",*coords)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
```

Adicionar essas funções logo após a função fecharlinha e antes de root = Tk() .

```
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
janela.bind("<ButtonRelease-3>", deselecionarlinha)
```

Essas três linhas podem ser adicionadas logo após janela.bind("<ButtonRelease-1>", fecharlinha) e antes de root.mainloop()

1. Interfaces gráficas - Tkinter

- A tag CURRENT equivale ao item sobre o ponteiro do mouse:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    coords = janela.coords("sel")
    for i in range(len(coords)):
        if i%2 == 0:
            coords[i] = coords[i] + x1-x0
        else:
            coords[i] = coords[i] + y1-y0
    janela.coords("sel",*coords)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
```

```
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
janela.bind("<ButtonRelease-3>", deselecionarlinha)
```

Vamos utilizar o clique do botão direito do mouse para movimentar os itens criados.

1. Interfaces gráficas - Tkinter

- A tag CURRENT equivale ao item sobre o ponteiro do mouse:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    coords = janela.coords("sel")
    for i in range(len(coords)):
        if i%2 == 0:
            coords[i] = coords[i] + x1-x0
        else:
            coords[i] = coords[i] + y1-y0
    janela.coords("sel",*coords)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
janela.bind("<ButtonRelease-3>", deselecionarlinha)
```

Quando clicamos com o botão direito, guardamos a posição do clique e atribuímos a tag 'sel' ao item abaixo da ponteira do mouse.

1. Interfaces gráficas - Tkinter

- A tag CURRENT equivale ao item sobre o ponteiro do mouse:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    coords = janela.coords("sel")
    for i in range(len(coords)):
        if i%2 == 0:
            coords[i] = coords[i] + x1-x0
        else:
            coords[i] = coords[i] + y1-y0
    janela.coords("sel",*coords)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
janela.bind("<ButtonRelease-3>", deselecionarlinha)
```

Ao mover o mouse com o botão direito clidado, mudamos as coordenadas o item com tag sel.

1. Interfaces gráficas - Tkinter

- A tag CURRENT equivale ao item sobre o ponteiro do mouse:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    coords = janela.coords("sel")
    for i in range(len(coords)):
        if i%2 == 0:
            coords[i] = coords[i] + x1-x0
        else:
            coords[i] = coords[i] + y1-y0
    janela.coords("sel",*coords)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
janela.bind("<ButtonRelease-3>", deselecionarlinha)
```

Cada ponto das coordenadas é somado a um mesmo valor que depende da posição inicial do clique e da posição atual do mouse.

1. Interfaces gráficas - Tkinter

- A tag CURRENT equivale ao item sobre o ponteiro do mouse:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    coords = janela.coords("sel")
    for i in range(len(coords)):
        if i%2 == 0:
            coords[i] = coords[i] + x1-x0
        else:
            coords[i] = coords[i] + y1-y0
    janela.coords("sel",*coords)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
janela.bind("<ButtonRelease-3>", deselecionarlinha)
```

As coordenadas do item selecionado são modificados.

1. Interfaces gráficas - Tkinter

- A tag CURRENT equivale ao item sobre o ponteiro do mouse:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    coords = janela.coords("sel")
    for i in range(len(coords)):
        if i%2 == 0:
            coords[i] = coords[i] + x1-x0
        else:
            coords[i] = coords[i] + y1-y0
    janela.coords("sel",*coords)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
janela.bind("<ButtonRelease-3>", deselecionarlinha)
```

A tag ‘sel’ é apagada.

1. Interfaces gráficas - Tkinter

- Uma forma mais fácil de mover a linha é utilizando o método move:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.itemconfig(CURRENT, tags="sel")
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.move("sel",x1-x0,y1-y0)
    x0,y0=x1,y1
def deselecionarlinha(evento):
    janela.itemconfig("sel", tags=())
```

Soma coordenada passada (no caso, x1-x0,y1-y0) às coordenadas originais do item com tag 'sel'. Com coords era necessário um for, com o move resolvemos o problema em só uma linha.

1. Interfaces gráficas - Tkinter

- Observação: a tag ‘sel’ não é necessária, utilizando somente

  CURRENT podemos movimentar um item:

```
def selecionarlinha(evento):
    global x0,y0
    x0,y0 = janela.canvasx(evento.x), janela.canvasy(evento.y)
def moverlinha (evento):
    global x0,y0
    x1,y1 = janela.canvasx(evento.x), janela.canvasy(evento.y)
    janela.move(CURRENT,x1-x0,y1-y0)
    x0,y0=x1,y1
root = Tk()
janela = Canvas(root,width=512, height=512)
janela.pack()
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
janela.bind("<ButtonRelease-1>", fecharlinha)
janela.bind("<Button-3>", selecionarlinha)
janela.bind("<B3-Motion>", moverlinha)
```

1. Interfaces gráficas - Tkinter

- Método after:

```
from tkinter import *

def mover_cir():
    pontos = canvas.coords(cir)
    if pontos[2] >= canvas.winfo_width() or\
       pontos[3] >= canvas.winfo_height():
        print('Fim')
        return
    canvas.move(cir, 1, 1)
    canvas.after(20, mover_cir)

root = Tk()
canvas = Canvas(root, width = 500, height = 500)
canvas.pack()
raio = 10
canvas.update()
cir = canvas.create_oval(-raio,-raio,raio,raio)
mover_cir()
root.mainloop()
```

Esse método atualiza o canvas, podendo ser usado para criar a sensação de movimento. canvas.after(tempo [ms], função).

\Exemplos\ex28CanvasAfter_Python3.py

1. Interfaces gráficas - Tkinter

- Método after:

```
from tkinter import *

def mover_cir():
    pontos = canvas.coords(cir)
    if pontos[2] >= canvas.winfo_width() or\
       pontos[3] >= canvas.winfo_height():
        print('Fim')
        return
    canvas.move(cir, 1, 1)
    canvas.after(20, mover_cir)

root = Tk()
canvas = Canvas(root, width = 500, height = 500)
canvas.pack()
raio = 10
canvas.update()
cir = canvas.create_oval(-raio,-raio,raio,raio)
mover_cir()
root.mainloop()
```

Um canvas é criado.

Criando um círculo dentro do canvas.

\Exemplos\ex28CanvasAfter_Python3.py

1. Interfaces gráficas - Tkinter

- Método after:

```
from tkinter import *

def mover_cir():
    pontos = canvas.coords(cir)
    if pontos[2] >= canvas.winfo_width() or\
       pontos[3] >= canvas.winfo_height():
        print('Fim')
        return
    canvas.move(cir, 1, 1)
    canvas.after(20, mover_cir)

root = Tk()
canvas = Canvas(root, width = 500, height = 500)
canvas.pack()
raio = 10
canvas.update()
cir = canvas.create_oval(-raio,-raio,raio,raio)
mover_cir()
root.mainloop()
```

Verifica se a posição do círculo está dentro do canvas.

Esse update é necessário para que winfo_width e winfo_height funcionem.

\Exemplos\ex28CanvasAfter_Python3.py

1. Interfaces gráficas - Tkinter

- Método after:

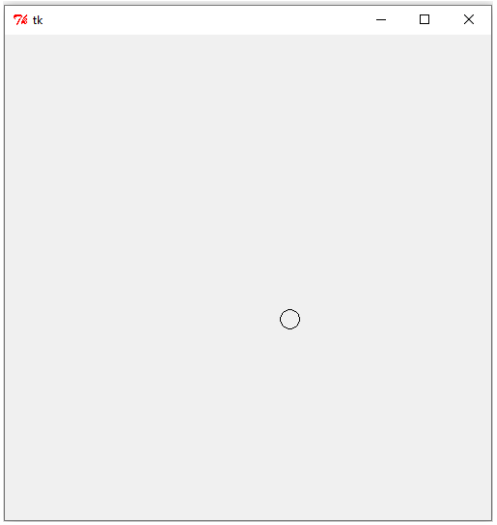
```
from tkinter import *

def mover_cir():
    pontos = canvas.coords(cir)
    if pontos[2] >= canvas.winfo_width() or\
       pontos[3] >= canvas.winfo_height():
        print('Fim')
        return
    canvas.move(cir, 1, 1)
    canvas.after(20, mover_cir)

root = Tk()
canvas = Canvas(root, width = 500, height = 500)
canvas.pack()
raio = 10
canvas.update()
cir = canvas.create_oval(-raio,-raio,raio,raio)
mover_cir()
root.mainloop()
```

Altera a posição do círculo (somando 1 à x e a y), a cada vez que move_cir é chamada.

\Exemplos\ex28CanvasAfter_Python3.py

1. Interfaces gráficas - Tkinter

- Método after:




```
from tkinter import *

def mover_cir():
    pontos = canvas.coords(cir)
    if pontos[2] >= canvas.winfo_width() or\
       pontos[3] >= canvas.winfo_height():
        print('Fim')
        return
    canvas.move(cir, 1, 1)
    canvas.after(20, mover_cir)
```

Espera 20 ms e chama move_cir de forma iterativa. Dessa forma, é criada a sensação de movimento.

```
root = Tk()
canvas = Canvas(root, width = 500, height = 500)
canvas.pack()
raio = 10
canvas.update()
cir = canvas.create_oval(-raio,-raio,raio,raio)
mover_cir()
root.mainloop()
```

\Exemplos\ex28CanvasAfter_Python3.py

1. Interfaces gráficas - Tkinter

- Método after – barra de progresso:

```
from tkinter import *
def barra_progresso():
    coords = canvas.coords(prog)
    if coords[2] == 990:
        return
    coords[2] = coords[2] + 10
    canvas.itemconfig(porcent,text='%i%%'%(100*coords[2]/990))
    canvas.coords(prog,*coords)
    canvas.after(60, barra_progresso)
root = Tk()
canvas = Canvas(root, width = 1000, height = 100)
canvas.pack()
borda = canvas.create_rectangle(10, 10, 990, 90)
prog = canvas.create_rectangle(10, 10, 10, 90, fill = 'blue')
porcent = canvas.create_text(495,45,text='%i%%'%(100*10/990),\
                             font=('Arial',12))
barra_progresso()
root.mainloop()
```

1. Interfaces gráficas - Tkinter

- Método after – barra de progresso:

```
from tkinter import *
def barra_progresso():
    coords = canvas.coords(prog)
    if coords[2] == 990:
        return
    coords[2] = coords[2] + 10
    canvas.itemconfig(porcent,text='%i%%'%(100*coords[2]/990))
    canvas.coords(prog,*coords)
    canvas.after(60, barra_progresso)
root = Tk()
canvas = Canvas(root, width = 1000, height = 100)
canvas.pack()
borda = canvas.create_rectangle(10, 10, 990, 90)
prog = canvas.create_rectangle(10, 10, 10, 90, fill = 'blue')
porcent = canvas.create_text(495,45,text='%i%%'%(100*10/990),\
                             font=('Arial',12))
barra_progresso()
root.mainloop()
```

Cria um retângulo com preenchimento azul e um texto com a porcentagem do retângulo maior que está preenchida.

1. Interfaces gráficas - Tkinter

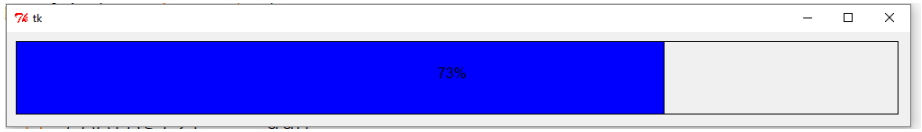
- Método after – barra de progresso:

```
from tkinter import *
def barra_progresso():
    coords = canvas.coords(prog)
    if coords[2] == 990:
        return
    coords[2] = coords[2] + 10
    canvas.itemconfig(porcent,text='%i%%'%(100*coords[2]/990))
    canvas.coords(prog,*coords)
    canvas.after(60, barra_progresso)
root = Tk()
canvas = Canvas(root, width = 1000, height = 100)
canvas.pack()
borda = canvas.create_rectangle(10, 10, 990, 90)
prog = canvas.create_rectangle(10, 10, 10, 90, fill = 'blue')
porcent = canvas.create_text(495,45,text='%i%%'%(100*10/990),\
                             font=('Arial',12))
barra_progresso()
root.mainloop()
```

Aumenta o retângulo com preenchimento azul e modifica do texto com a porcentagem cada vez que barra_progresso() é chamada.

1. Interfaces gráficas - Tkinter

- Método after – barra de progresso:




```
from tkinter import *
def barra_progresso():
    coords = canvas.coords(prog)
    if coords[2] == 990:
        return
    coords[2] = coords[2] + 10
    canvas.itemconfig(porcent,text='%i%%'%(100*coords[2]/990))
    canvas.coords(prog,*coords)
    canvas.after(60, barra_progresso)
root = Tk()
canvas = Canvas(root, width = 1000, height = 100)
canvas.pack()
borda = canvas.create_rectangle(10, 10, 990, 90)
prog = canvas.create_rectangle(10, 10, 10, 90, fill = 'blue')
porcent = canvas.create_text(495,45,text='%i%%'%(100*10/990),\
                             font=('Arial',12))
barra_progresso()
root.mainloop()
```

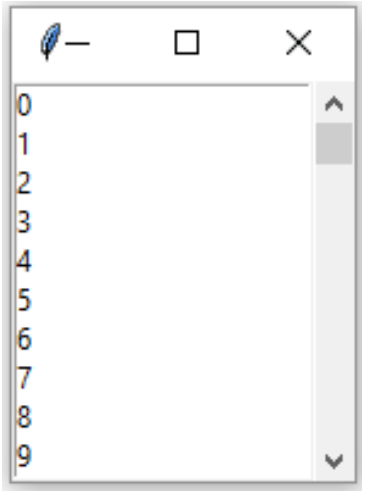
A função barra_progresso() é chamada de forma recursiva a cada 60 ms.

1. Interfaces gráficas - Tkinter

- Scrollbar – muitas vezes utilizada em conjunto com outra widget, como lista de itens, caixa de texto ou o canvas.
  - Primeiro passo: criar a widget scrollbar;
  - Segundo passo: criar a widget que desejamos associar a scrollbar, utilizando yscrollcommand=scrollbar.set como uma das opções;
  - Terceiro passo: configurar o command do scrollbar utilizando o método yview da widget que foi associada.

1. Interfaces gráficas - Tkinter

- Scrollbar para uma lista de itens:




```
from tkinter import *

root = Tk()

scroll = Scrollbar(root)
scroll.pack(side=RIGHT, fill=Y)

listaItens = Listbox(root,yscrollcommand=scroll.set)
listaItens.pack()

for i in range(100):
    listaItens.insert(END, i)

scroll.config(command=listaItens.yview)

root.mainloop()
```

\Aula9_Tkinter\ex31ScrollListaItens_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para uma lista de itens:

```
from tkinter import *

root = Tk()

scroll = Scrollbar(root)
scroll.pack(side=RIGHT, fill=Y)

listaItens = Listbox(root,yscrollcommand=scroll.set)
listaItens.pack()

for i in range(100):
    listaItens.insert(END, i)

scroll.config(command=listaItens.yview)

root.mainloop()
```

Primeiro passo: criando a widget scrollbar.

A opção fill=Y vai preencher a scrollbar ao longo do eixo Y.

\Aula9_Tkinter\ex31ScrollListaItens_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para uma lista de itens:

```
from tkinter import *

root = Tk()

scroll = Scrollbar(root)
scroll.pack(side=RIGHT, fill=Y)

listaItens = Listbox(root,yscrollcommand=scroll.set)
listaItens.pack()

for i in range(100):
    listaItens.insert(END, i)

scroll.config(command=listaItens.yview)

root.mainloop()
```

Segundo passo: criando a widget Listbox e associando ao item scroll.

\Aula9_Tkinter\ex31ScrollListaItens_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para uma lista de itens:

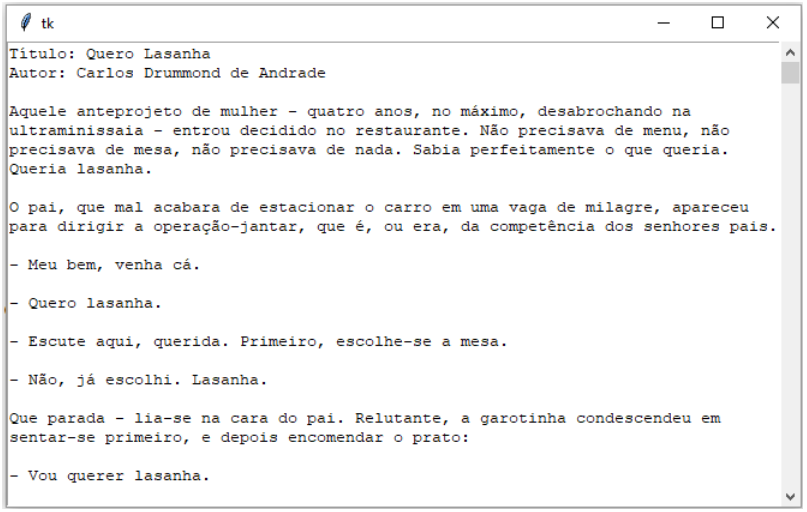
```
from tkinter import *

root = Tk()

scroll = Scrollbar(root)
scroll.pack(side=RIGHT, fill=Y)

listaItens = Listbox(root,yscrollcommand=scroll.set)
listaItens.pack()

for i in range(100):
    listaItens.insert(END, i)

scroll.config(command=listaItens.yview)

root.mainloop()
```

Terceiro passo: configurando a opção command de scroll como o método yview da Listbox.

\Aula9_Tkinter\ex31ScrollListaItens_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para uma caixa de texto:




```
from tkinter import *

root = Tk()

with open('contos.txt','r') as f:
    conteudo = f.read()

scrollbar = Scrollbar(root)
scrollbar.pack(side=RIGHT, fill=Y)

text = Text(root,wrap=WORD,yscrollcommand=scrollbar.set)
text.pack()
text.insert(INSERT,conteudo)

scrollbar.config(command=text.yview)

root.mainloop()
```

\Aula9_Tkinter\ex32ScrollTexto_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para uma caixa de texto:

```
from tkinter import *

root = Tk()

with open('contos.txt','r') as f:
    conteudo = f.read()
```

Primeiro passo.

```
scrollbar = Scrollbar(root)
scrollbar.pack(side=RIGHT, fill=Y)
```

Segundo passo.

```
text = Text(root,wrap=WORD,yscrollcommand=scrollbar.set)
text.pack()
text.insert(INSERT,conteudo)
```

```
scrollbar.config(command=text.yview)
```

Terceiro passo.

```
root.mainloop()
```

\Aula9_Tkinter\ex32ScrollTexto_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para uma caixa de texto – horizontal e vertical:

```
from tkinter import *

root = Tk()

with open('contos.txt','r') as f:
    conteudo = f.read()
root.grid_rowconfigure(0, weight=1)
root.grid_columnconfigure(0, weight=1)
scrollY = Scrollbar(root)
scrollY.grid(row=0, column=1, sticky=N+S)
scrollX = Scrollbar(root, orient=HORIZONTAL)
scrollX.grid(row=1, column=0, sticky=E+W)
text = Text(root,wrap=NONE,xscrollcommand=scrollX.set,\
            yscrollcommand=scrollY.set)
text.grid(row=0, column=0, sticky=N+S+E+W)
text.insert(INSERT,conteudo)
scrollY.config(command=text.yview)
scrollX.config(command=text.xview)

root.mainloop()
```

Necessário para que a barra de rolagem funcione corretamente quando mudamos o tamanho da janela.

Primeiro passo.

Por padrão, a barra a parece na vertical, devemos garantir que ela está na horizontal.

Segundo passo.

A opção wrap=NONE garante que não haverá quebra de linha no final da widget.

Terceiro passo.

\Aula9_Tkinter\ex33ScrollTextoHV_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para o canvas:

Os três passos são necessários, mas também é necessário dizer a região de scroll – sem a região a barra aparece no canvas, mas não funciona.

```
from tkinter import *
def novalinha(ev):
    x,y = janela.canvasx(ev.x),janela.canvasy(ev.y)
    janela.create_line(x,y,x,y,tags="linha")
def estenderlinha(ev):
    x,y = janela.canvasx(ev.x),janela.canvasy(ev.y)
    coords = janela.coords("linha") + [x,y]
    janela.coords("linha",*coords)
root = Tk()
root.grid_rowconfigure(0, weight=1)
root.grid_columnconfigure(0, weight=1)
scroll = Scrollbar(root)
scroll.grid(row=0, column=1, sticky=N+S)
janela = Canvas(root,scrollregion=(0, 0, 1000, 1000),\
                yscrollcommand=scroll.set)
janela.grid(row=0, column=0, sticky=N+S+E+W)
scroll.config(command=janela.yview)
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
root.mainloop()
```

Primeiro passo.

Segundo passo.

Terceiro passo.

Definir a região de scroll é necessária, pois o sistema de coordenadas do canvas não é limitado ao tamanho do canvas.

\Aula9_Tkinter\ex34ScrollCanvas_Python3.py

1. Interfaces gráficas - Tkinter

- Scrollbar para o canvas – horizontal e vertical:

```
root = Tk()
root.grid_rowconfigure(0, weight=1)
root.grid_columnconfigure(0, weight=1)
scrollY = Scrollbar(root)
scrollY.grid(row=0, column=1, sticky=N+S)
scrollX = Scrollbar(root, orient=HORIZONTAL)
scrollX.grid(row=1, column=0, sticky=E+W)
janela = Canvas(root,scrollregion=(0, 0, 1000, 1000),\
                yscrollcommand=scrollY.set,\
                xscrollcommand=scrollX.set)
janela.grid(row=0, column=0, sticky=N+S+E+W)
scrollY.config(command=janela.yview)
scrollX.config(command=janela.xview)
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
root.mainloop()
```

Primeiro passo.

Segundo passo.

Terceiro passo.

\Aula9_Tkinter\ex35ScrollCanvasHV_Python3.py

1. Interfaces gráficas - Tkinter

- Adicionando um frame - scrollbar para o canvas, horizontal e vertical:

```
root = Tk()
frame = Frame(root)
frame.grid_rowconfigure(0, weight=1)
frame.grid_columnconfigure(0, weight=1)
frame.grid(row=0, column=0, sticky=N+S+E+W)
scrollY = Scrollbar(frame)
scrollY.grid(row=0, column=1, sticky=N+S)
scrollX = Scrollbar(frame, orient=HORIZONTAL)
scrollX.grid(row=1, column=0, sticky=E+W)
janela = Canvas(frame,scrollregion=(0, 0, 1000, 1000),\
                yscrollcommand=scrollY.set,\
                xscrollcommand=scrollX.set)
janela.grid(row=0, column=0, sticky=N+S+E+W)
scrollY.config(command=janela.yview)
scrollX.config(command=janela.xview)
janela.bind("<Button-1>", novalinha)
janela.bind("<B1-Motion>", estenderlinha)
root.mainloop()
```

\Aula9_Tkinter\ex35ScrollCanvasHV_Python3.py

1. Interfaces gráficas - Tkinter

- Adicionando um menu e abrindo uma nova janela:

```
from tkinter import *
def imprimir():
    print("Ola!")
def funcaoSobre():
    janela = Toplevel(root)
    labelSobre = Label(janela,text = 'Exemplo para a turma de CompII.')
    labelSobre.pack()
root = Tk()
# Menu
menuObj = Menu(root)
root.config(menu=menuObj)
menuArquivo = Menu(menuObj)
menuObj.add_cascade(label="Arquivo", menu=menuArquivo)
menuArquivo.add_command(label="Novo", command=imprimir)
menuArquivo.add_command(label="Abrir...", command=imprimir)
menuArquivo.add_separator()
menuArquivo.add_command(label="Sair", command=root.destroy)
menuAjuda = Menu(menuObj)
menuObj.add_cascade(label="Ajuda", menu=menuAjuda)
menuAjuda.add_command(label="Sobre...", command=funcaoSobre)
root.mainloop()
```

\Aula9_Tkinter\ex38Menu_Python3.py

1. Interfaces gráficas - Tkinter

- Detalhes sobre o menu:

```
root = Tk()
# Menu
menuObj = Menu(root)
root.config(menu=menuObj)
menuArquivo = Menu(menuObj)
menuObj.add_cascade(label="Arquivo", menu=menuArquivo)
menuArquivo.add_command(label="Novo", command=imprimir)
menuArquivo.add_command(label="Abrir...", command=imprimir)
menuArquivo.add_separator()
menuArquivo.add_command(label="Sair", command=root.destroy)
menuAjuda = Menu(menuObj)
menuObj.add_cascade(label="Ajuda", menu=menuAjuda)
menuAjuda.add_command(label="Sobre...", command=funcaoSobre)
root.mainloop()
```

\Aula9_Tkinter\ex38Menu_Python3.py

1. Interfaces gráficas - Tkinter

- Detalhes sobre o menu:

```
root = Tk()
# Menu
menuObj = Menu(root)
root.config(menu=menuObj)
menuArquivo = Menu(menuObj)
menuObj.add_cascade(label="Arquivo", menu=menuArquivo)
menuArquivo.add_command(label="Novo", command=imprimir)
menuArquivo.add_command(label="Abrir...", command=imprimir)
menuArquivo.add_separator()
menuArquivo.add_command(label="Sair", command=root.destroy)
menuAjuda = Menu(menuObj)
menuObj.add_cascade(label="Ajuda", menu=menuAjuda)
menuAjuda.add_command(label="Sobre...", command=funcaoSobre)
root.mainloop()
```

\Aula9_Tkinter\ex38Menu_Python3.py

1. Interfaces gráficas - Tkinter

- Abrindo uma nova janela:

```
from tkinter import *
def imprimir():
    print("Ola!")
def funcaoSobre():
    janela = Toplevel(root)
    labelSobre = Label(janela,text = 'Exemplo para a turma de CompII.')
    labelSobre.pack()
```

\Aula9_Tkinter\ex38Menu_Python3.py