Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Python
Tkinter konténer widget-ek

Dr. Schuster György

2017. november 13.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

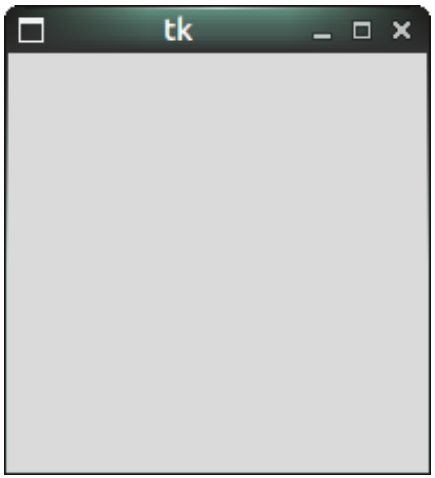
Létrehozása és konfigurálása:

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
      ⋮
mw.mainloop()
      ⋮
```



Létrehoztuk az alap ablakot.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.title("MW")
      ⋮
mw.mainloop()
      ⋮
```

Létrehoztuk az alap ablakot. Megváltoztattuk a kiírt nevét.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(width=100)
mw.config(height=100)
      ⋮
mw.mainloop()
      ⋮
```

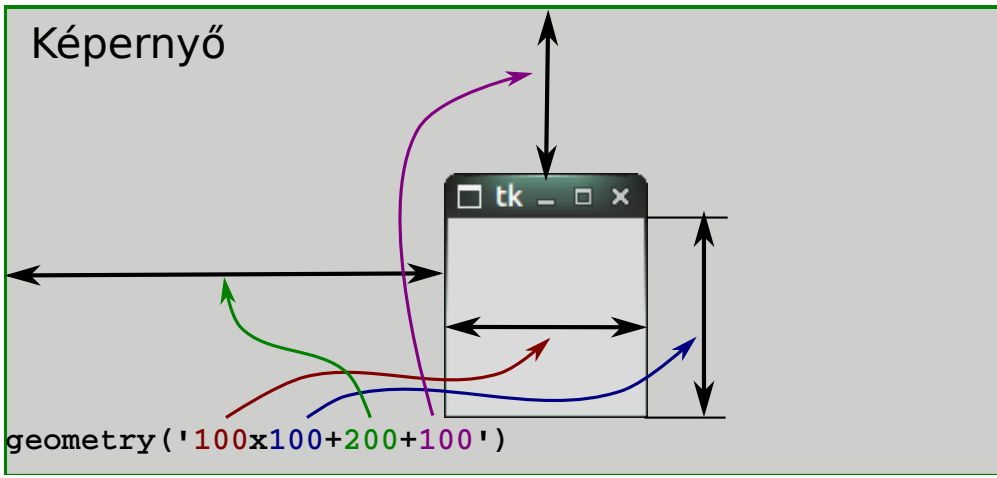
Létrehoztuk az alap ablakot.
Megváltoztattuk a kiírt nevét.
Megváltoztattuk a méreteit.

Tk widget

A **Tk** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.geometry('100x100+200+100')
      ⋮
mw.mainloop()
      ⋮
```



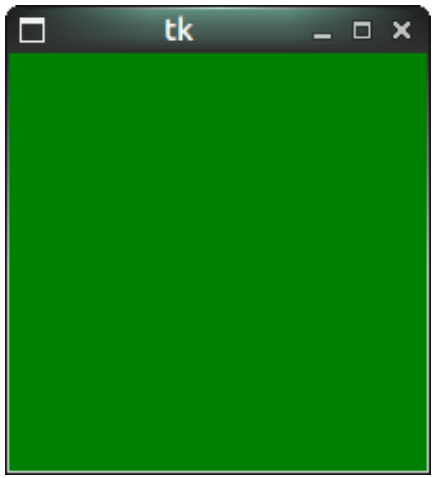
Megváltoztattuk a méreteit és elhelyeztük a képernyőn.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(background='green')
      ⋮
mw.mainloop()
      ⋮
```



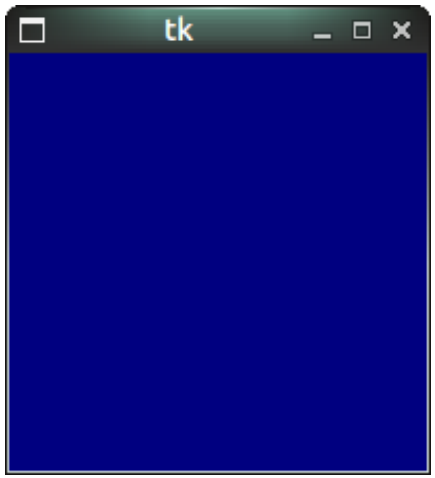
Létrehoztuk az alap ablakot. Megváltoztattuk a kiírt nevét. Megváltoztattuk a méreteit. Megváltoztattuk a háttérszínt, név szerint.

Tk widget

A **Tk** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(background='#000080')
      ⋮
mw.mainloop()
      ⋮
```



Létrehoztuk az alap ablakot.
Megváltoztattuk a kiírt nevét.
Megváltoztattuk a méreteit.
Megváltoztattuk a háttérszínt, név szerint.
Megváltoztattuk a háttérszínt, **RGB** értékek szerint.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(borderwidth=10)
      ⋮
      ⋮
mw.mainloop()
      ⋮
```

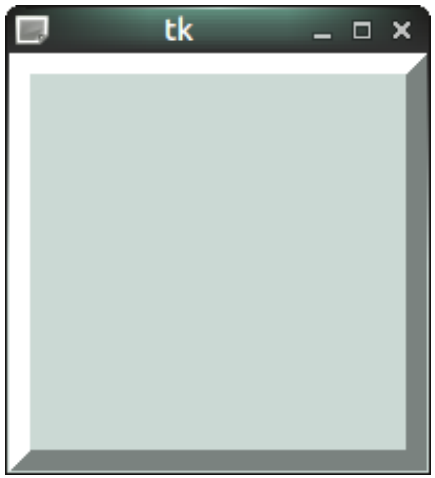
Megadjuk a keret méretét.
Az alap keret jelleg a lapos (**`flat`**).

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(borderwidth=10)
mw.config(relief='raise')
      ⋮
mw.mainloop()
      ⋮
```



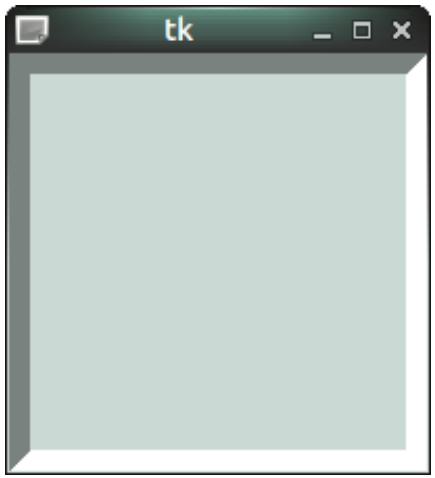
Megadjuk a keret méretét.
A keret kiemelkedő jellegű.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(borderwidth=10)
mw.config(relief='sunken')
      ⋮
mw.mainloop()
      ⋮
```



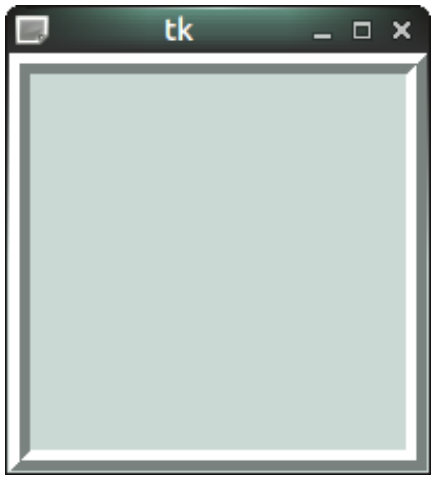
Megadjuk a keret méretét.
A keret süllyesztett jellegű.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(borderwidth=10)
mw.config(relief='ridge')
      ⋮
mw.mainloop()
      ⋮
```



Megadjuk a keret méretét.
A keret gerinc jellegű.

Tk widget

A **`Tk`** widget a grafikus alkalmazás alapvető eleme ”gyökere”. Erre építhetjük fel a további elemeket.

Létrehozása és konfigurálása:

```
      ⋮
mw=Tk()
mw.config(borderwidth=10)
mw.config(relief='groove')
      ⋮
mw.mainloop()
      ⋮
```

Megadjuk a keret méretét.
A keret árok jellegű.

Tk widget

A Tk widget függvényei:

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:

Tk widget

A Tk widget függvényei:

`title`	az ablak fejlécében a feliratot adhatjuk meg. Formátuma: **`widget_neve.title('text')`**

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:
`widget_neve.title('text')`
Példa:
`wm.title('MW')`

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:
```
widget_neve.title('text')
```
Példa:
```
wm.title('MW')
```

`config` a widget paramétereit lehet beállítani (lásd méret vagy háttérszín beállítását). A formátuma:

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:
```
widget_neve.title('text')
```
Példa:
```
wm.title('MW')
```

`config` a widget paramétereit lehet beállítani (lásd méret vagy háttérszín beállítását). A formátuma:
```
widget_neve.config(param=value)
```

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:
`widget_neve.title('text')`
Példa:
`wm.title('MW')`

`config` a widget paramétereit lehet beállítani (lásd méret vagy háttérszín beállítását). A formátuma:
`widget_neve.config(param=value)`
Példa:
`wm.config(width=100)`

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:
```
widget_neve.title('text')
```
Példa:
```
wm.title('MW')
```

`config` a widget paramétereit lehet beállítani (lásd méret vagy háttérszín beállítását). A formátuma:
```
widget_neve.config(param=value)
```
Példa:
```
wm.config(width=100)
```

`geometry` az ablak méreteit és pozícióját adja meg. Formátuma:

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:
```
widget_neve.title('text')
```
Példa:
```
wm.title('MW')
```

`config` a widget paramétereit lehet beállítani (lásd méret vagy háttérszín beállítását). A formátuma:
```
widget_neve.config(param=value)
```
Példa:
```
wm.config(width=100)
```

`geometry` az ablak méreteit és pozícióját adja meg. Formátuma:
```
widget_neve.geometry('param_string')
```

Tk widget

A Tk widget függvényei:

`title` az ablak fejlécében a feliratot adhatjuk meg. Formátuma:
```
widget_neve.title('text')
```
Példa:
```
wm.title('MW')
```

`config` a widget paramétereit lehet beállítani (lásd méret vagy háttérszín beállítását). A formátuma:
```
widget_neve.config(param=value)
```
Példa:
```
wm.config(width=100)
```

`geometry` az ablak méreteit és pozícióját adja meg. Formátuma:
```
widget_neve.geometry('param_string')
```
Példa:
```
wm.geometry('100x100+200+100')
```

Tk widget

`winfo_width` az ablak szélességének lekérdezése.

`winfo_height` az ablak magasságának lekérdezése.

`winfo_geometry` a geometriai paraméterek lekédezése.

Tk widget

winfo_width az ablak szélességének lekérdezése.

winfo_height az ablak magasságának lekérdezése.

winfo_geometry a geometriai paraméterek lekédezése.

```
        ⋮
        ⋮
w=mw.winfo_width()
h=mw.winfo_height()
w=mw.winfo_geometry()
        ⋮
```

Képernyő

```
1
1
1x1+0+0
```

Valami gond van!

Tk widget

winfo_width az ablak szélességének lekérdezése.

winfo_height az ablak magasságának lekérdezése.

winfo_geometry a geometriai paraméterek lekédezése.

```
      ⋮
      ⋮
w=mw.winfo_width()
h=mw.winfo_height()
w=mw.winfo_geometry()
      ⋮
```

Képernyő

```
1
1
1x1+0+0
```

Valami gond van!

update a paraméterek beállítása és az értékek frissítése.

Tk widget

winfo_width az ablak szélességének lekérdezése.

winfo_height az ablak magasságának lekérdezése.

winfo_geometry a geometriai paraméterek lekédezése.

```
      ⋮
mw.update()
w=mw.winfo_width()
h=mw.winfo_height()
w=mw.winfo_geometry()
      ⋮
```

Képernyő

```
100
100
100x100+202+123
```

Ok!

update a paraméterek beállítása és az értékek frissítése.

Tk widget

winfo_width az ablak szélességének lekérdezése.

winfo_height az ablak magasságának lekérdezése.

winfo_geometry a geometriai paraméterek lekédezése.

```
      ⋮
mw.update()
w=mw.winfo_width()
h=mw.winfo_height()
w=mw.winfo_geometry()
      ⋮
```

Képernyő

```
100
100
100x100+202+123
```

Ok!

update a paraméterek beállítása és az értékek frissítése.

cget tetszőleges paraméter kiolvasását teszi lehetővé.

Tk widget

winfo_width az ablak szélességének lekérdezése.

winfo_height az ablak magasságának lekérdezése.

winfo_geometry a geometriai paraméterek lekédezése.

```
      ⋮
mw.update()
w=mw.winfo_width()
h=mw.winfo_height()
w=mw.winfo_geometry()
      ⋮
```

Képernyő

```
100
100
100x100+202+123
```

Ok!

update a paraméterek beállítása és az értékek frissítése.

cget tetszőleges paraméter kiolvasását teszi lehetővé.

```
      ⋮
w=mw.cget('background')
      ⋮
```

Képernyő

```
#cbd9d4
```

Tk widget

`after`	lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
      ⋮
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
      ⋮
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
      ⋮
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
      ⋮
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.
A probléma az, hogy ez csak egyszer hajtódik végre.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
    mw.after(1000,fgv)
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.

A megoldás.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
    mw.after(1000,fgv)
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.

A megoldás.

resizable az ablak átmérezhetőségét engedélyezi, vagy tiltja.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
    mw.after(1000,fgv)
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.

A megoldás.

resizable az ablak átmérezhetőségét engedélyezi, vagy tiltja.

```
      ⋮
mw.resizable(0,0)
      ⋮
```

x tiltva,
y tiltva.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
    mw.after(1000,fgv)
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.

A megoldás.

resizable az ablak átmérezhetőségét engedélyezi, vagy tiltja.

```
      ⋮
mw.resizable(1,0)
      ⋮
```

x engedélyezve,
y tiltva.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
    mw.after(1000,fgv)
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.

A megoldás.

resizable az ablak átmérezhetőségét engedélyezi, vagy tiltja.

```
      ⋮
mw.resizable(0,1)
      ⋮
```

x tiltva,
y engedélyezve.

Tk widget

after lehetővé teszi, hogy egy függvény késleltetve indítsunk el.

```
      ⋮
def fgv():
    print(.)
    mw.after(1000,fgv)
      ⋮
mw.after(1000,fgv)
      ⋮
```

Az **after** függvény hívása.
A késleltetés értéke [ms]-ban.
A meghívott függvény.

A megoldás.

resizable az ablak átmérezhetőségét engedélyezi, vagy tiltja.

```
      ⋮
mw.resizable(1,1)
      ⋮
```

x engedélyezve,
y engedélyezve.

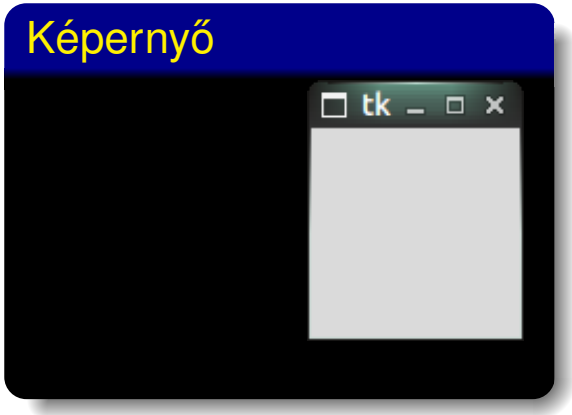
Tk widget

`protocol` az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```

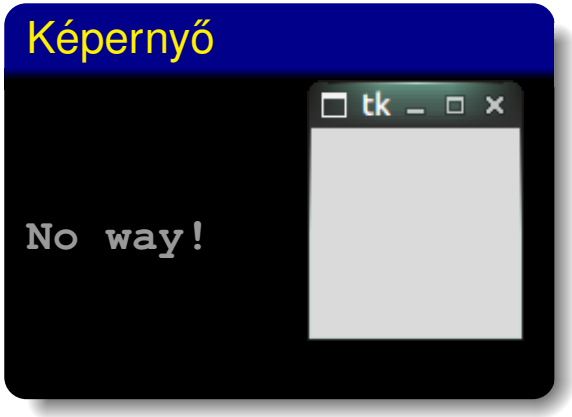


Nyomjunk Alt-F4-et!

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

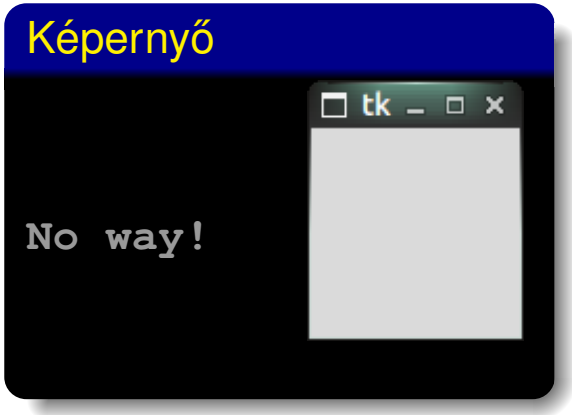
```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```



Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```

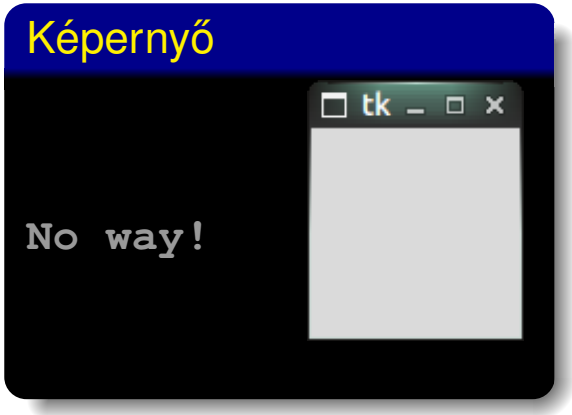


minsize megadja az ablak minimális méretét. Használata:

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```

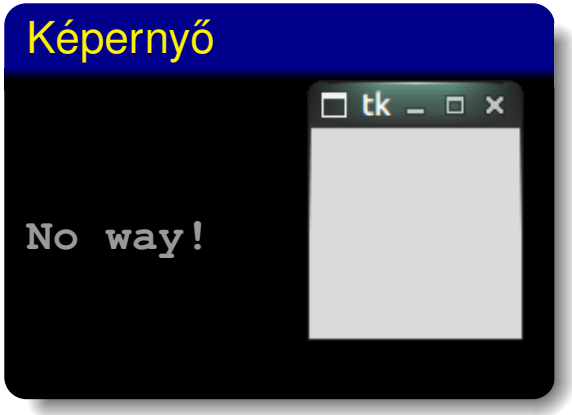


minsize megadja az ablak minimális méretét. Használata:
`mw.minsize(xmin,ymin)`

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```



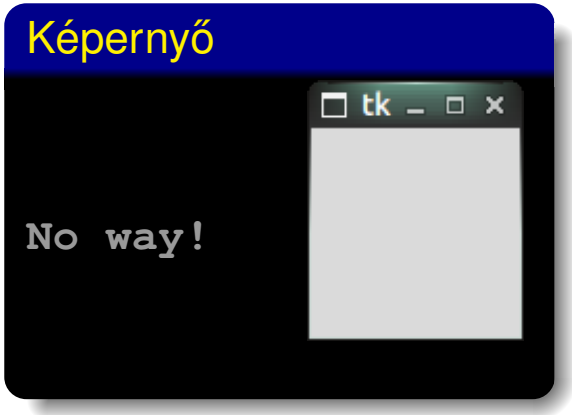
minsize megadja az ablak minimális méretét. Használata: **mw.minsize(xmin,ymin)**

maxsize megadja az ablak maximális méretét. Használata:

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```



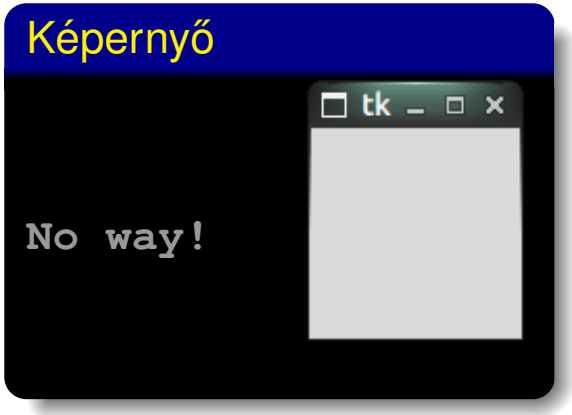
minsize megadja az ablak minimális méretét. Használata:
`mw.minsize(xmin,ymin)`

maxsize megadja az ablak maximális méretét. Használata:
`mw.maxsize(xmax,ymax)`

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```



minsize megadja az ablak minimális méretét. Használata: **mw.minsize(xmin,ymin)**

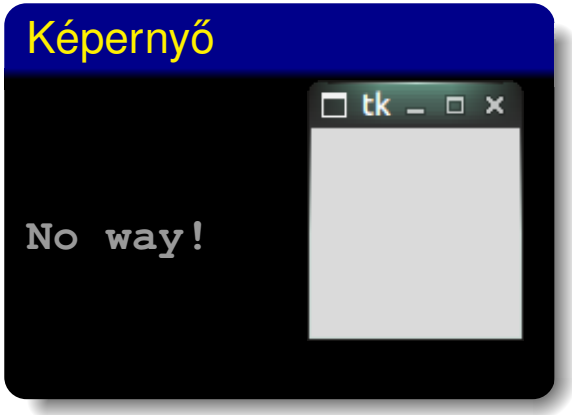
maxsize megadja az ablak maximális méretét. Használata: **mw.maxsize(xmax,ymax)**

focus_force a kérdéses widget-re teszi a fókuszt.

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```



minsize megadja az ablak minimális méretét. Használata: **mw.minsize(xmin,ymin)**

maxsize megadja az ablak maximális méretét. Használata: **mw.maxsize(xmax,ymax)**

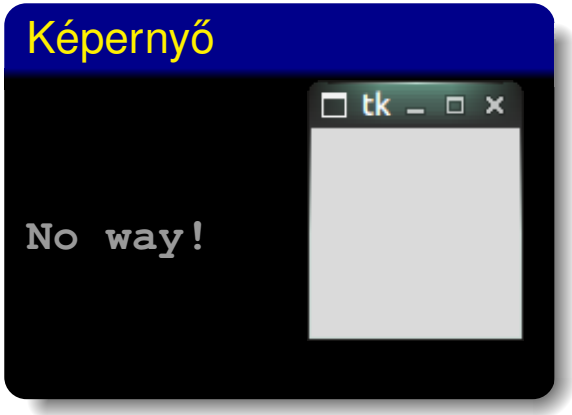
focus_force a kérdéses widget-re teszi a fókuszt.

withdraw minimalizálja az ablakot anélkül, hogy megszüntetné.

Tk widget

protocol az ablak kilépés gombjára és annak eseményére definiál egy úgynevezett callback függvényt.

```
      ⋮
def fgv():
    print("No way!")
      ⋮
mw.protocol('WM_DELETE_WINDOW',fgv)
      ⋮
```



minsize megadja az ablak minimális méretét. Használata: **mw.minsize(xmin,ymin)**

maxsize megadja az ablak maximális méretét. Használata: **mw.maxsize(xmax,ymax)**

focus_force a kérdéses widget-re teszi a fókuszt.

withdraw minimalizálja az ablakot anélkül, hogy megszüntetné.

destroy a kérdéses widget-et megszünteti. Ha a widget-nek voltak ”leszármazottai”, akkor megszünteti azokat is.

Toplevel widget

A **`Toplevel`** widget egy ugyanolyan ablakot tesz ki a képernyőre, mint a **`Tk`** widget. Csak nagyon kevés eltérés van.

Toplevel widget

A **`Toplevel`** widget egy ugyanolyan ablakot tesz ki a képernyőre, mint a **`Tk`** widget. Csak nagyon kevés eltérés van.

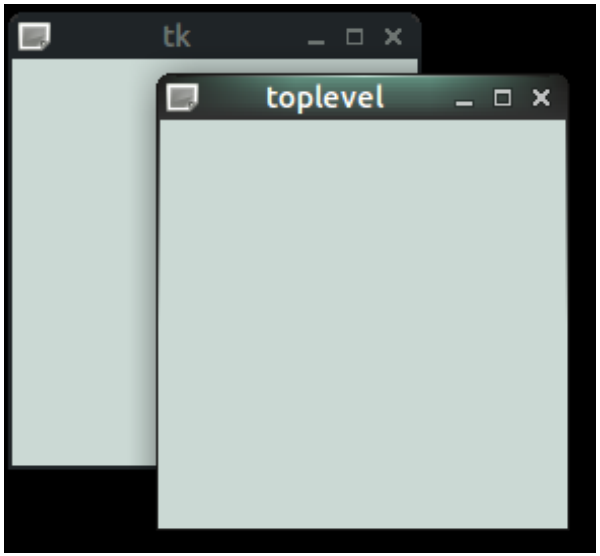
A widget létrehozása:

Toplevel widget

A **`Toplevel`** widget egy ugyanolyan ablakot tesz ki a képernyőre, mint a **`Tk`** widget. Csak nagyon kevés eltérés van.

A widget létrehozása:

```
        ⋮
mw=Tk()
        ⋮
tl=Toplevel(mw)
tl.title('toplevel')
tl.focus_force()
        ⋮
```



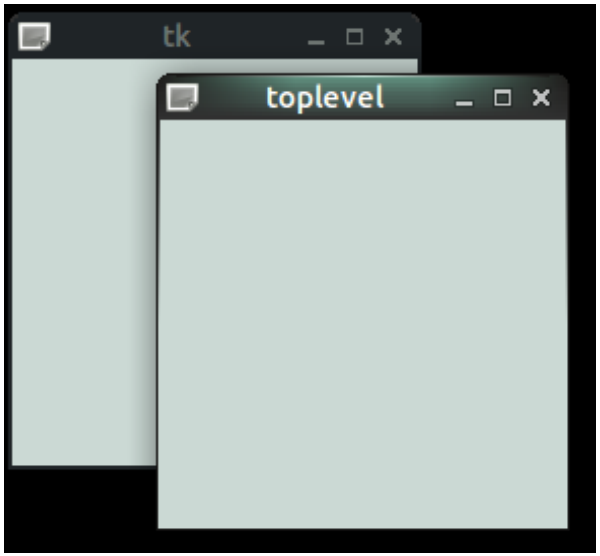
A widget létrehozása.

Toplevel widget

A **`Toplevel`** widget egy ugyanolyan ablakot tesz ki a képernyőre, mint a **`Tk`** widget. Csak nagyon kevés eltérés van.

A widget létrehozása:

```
        ⋮
mw=Tk()
        ⋮
tl=Toplevel(mw)
tl.title('toplevel')
tl.focus_force()
        ⋮
```



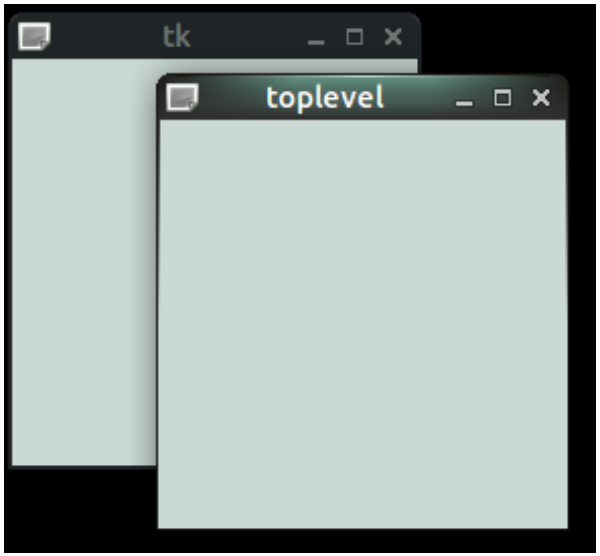
A widget létrehozása. Megadjuk, hogy kitől származik.

Toplevel widget

A **`Toplevel`** widget egy ugyanolyan ablakot tesz ki a képernyőre, mint a **`Tk`** widget. Csak nagyon kevés eltérés van.

A widget létrehozása:

```
        ⋮
mw=Tk()
        ⋮
tl=Toplevel(mw)
tl.title('toplevel')
tl.focus_force()
        ⋮
```



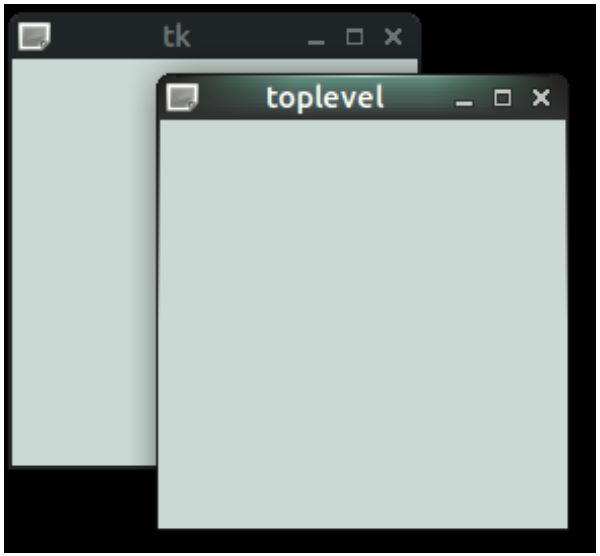
A widget létrehozása. Megadjuk, hogy kitől származik.
Adunk neki nevet, hogy tudjuk melyik.

Toplevel widget

A **`Toplevel`** widget egy ugyanolyan ablakot tesz ki a képernyőre, mint a **`Tk`** widget. Csak nagyon kevés eltérés van.

A widget létrehozása:

```
      ⋮
mw=Tk()
      ⋮
tl=Toplevel(mw)
tl.title('toplevel')
tl.focus_force()
      ⋮
```



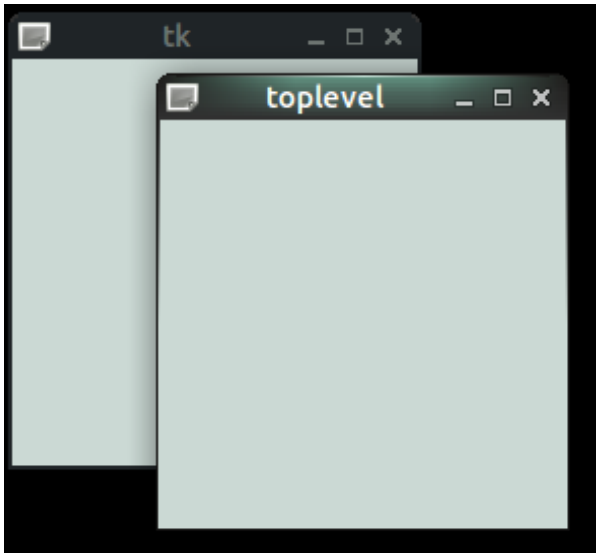
A widget létrehozása.
Megadjuk, hogy kitől származik.
Adunk neki nevet, hogy tudjuk melyik.
Rátesszük a fókuszt.

Toplevel widget

A **`Toplevel`** widget egy ugyanolyan ablakot tesz ki a képernyőre, mint a **`Tk`** widget. Csak nagyon kevés eltérés van.

A widget létrehozása:

```
        ⋮
mw=Tk()
        ⋮
tl=Toplevel(mw)
tl.title('toplevel')
tl.focus_force()
        ⋮
```



A widget létrehozása.
Megadjuk, hogy kitől származik.
Adunk neki nevet, hogy tudjuk melyik.
Rátesszük a fókuszt.

A példában a két ablakot széthúztuk, hogy jobban látszódjanak.

Toplevel widget

A **`Toplevel`** esetén a létrehozáskor is beállítható a konfiguráció.

Toplevel widget

A **`Toplevel`** esetén a létrehozáskor is beállítható a konfiguráció.

Példa:

Toplevel widget

A **`Toplevel`** esetén a létrehozáskor is beállítható a konfiguráció.

Példa:

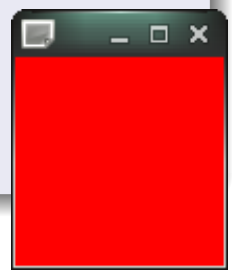
```
      ⋮
tl=Toplevel(mw,width=100,height=100,background='red')
tl.title('toplevel')
      ⋮
```

Toplevel widget

A **`Toplevel`** esetén a létrehozáskor is beállítható a konfiguráció.

Példa:

```
      ⋮
tl=Toplevel(mw,width=100,height=100,background='red')
tl.title('toplevel')
      ⋮
```

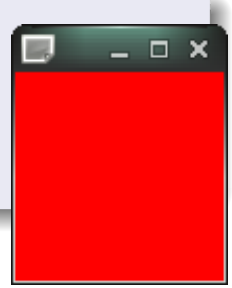


Toplevel widget

A **`Toplevel`** esetén a létrehozáskor is beállítható a konfiguráció.

Példa:

```
      ⋮
tl=Toplevel(mw,width=100,height=100,background='red')
tl.title('toplevel')
      ⋮
```



A **`title`** nem a Tkinter-nek ad át paramétert, hanem az ablakkezelőnek. Ezért nem "sima" paraméter.

Frame widget

A **`Frame`** widget arra szolgál, hogy más widget-eket geometriailag könnyen csoportba tudjunk rendezni.

Frame widget

A **`Frame`** widget arra szolgál, hogy más widget-eket geometriailag könnyen csoportba tudjunk rendezni.

Példa:

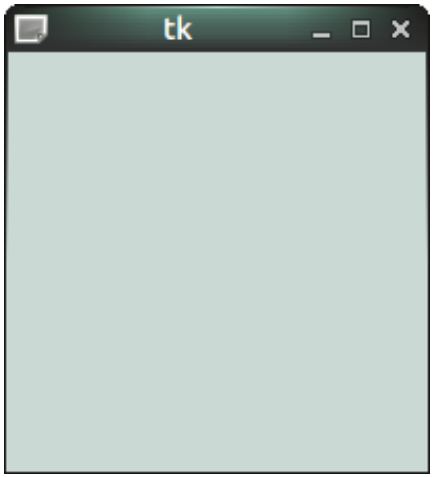
Frame widget

A **`Frame`** widget arra szolgál, hogy más widget-eket geometriailag könnyen csoportba tudjunk rendezni.

Példa:

```
        ⋮
mw=Tk()
mw.geometry('200x200+100+100')
fr=Frame(mw,width=200,height=100,bg='green')
fr.pack()
        ⋮
```

Létrehozzuk a fő ablakot.

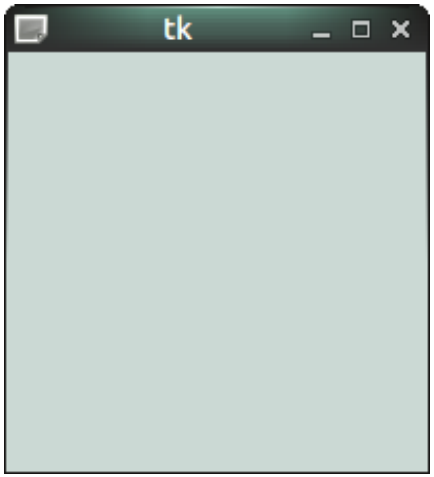


Frame widget

A **`Frame`** widget arra szolgál, hogy más widget-eket geometriailag könnyen csoportba tudjunk rendezni.

Példa:

```
        ⋮
mw=Tk()
mw.geometry('200x200+100+100')
fr=Frame(mw,width=200,height=100,bg='green')
fr.pack()
        ⋮
```



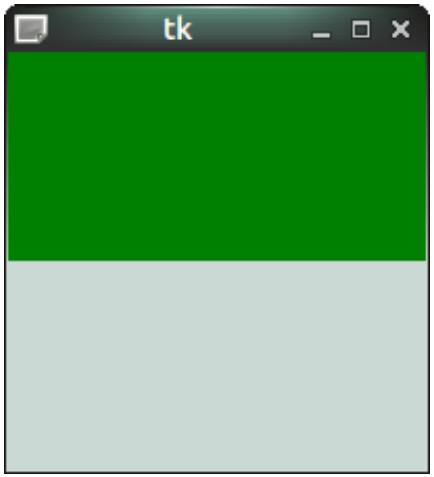
Létrehozzuk a fő ablakot.
Létrehozzuk és felparaméterezzük a **`Frame`**-t.

Frame widget

A **`Frame`** widget arra szolgál, hogy más widget-eket geometriailag könnyen csoportba tudjunk rendezni.

Példa:

```
        ⋮
mw=Tk()
mw.geometry('200x200+100+100')
fr=Frame(mw,width=200,height=100,bg='green')
fr.pack()
        ⋮
```



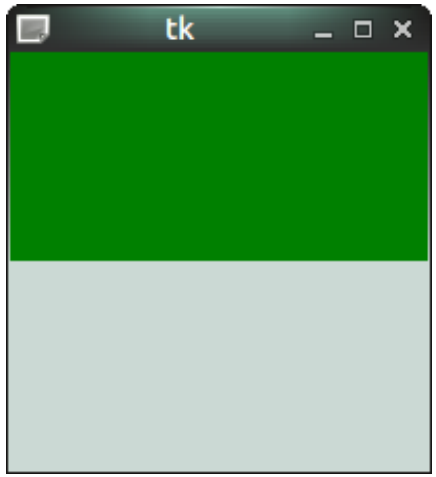
Létrehozzuk a fő ablakot.
Létrehozzuk és felparaméterezzük a **`Frame`**-t.
A **`Frame`** elhelyezése az ablakban.

Frame widget

A **Frame** widget arra szolgál, hogy más widget-eket geometriailag könnyen csoportba tudjunk rendezni.

Példa:

```
        ⋮
mw=Tk()
mw.geometry('200x200+100+100')
fr=Frame(mw,width=200,height=100,bg='green')
fr.pack()
        ⋮
```



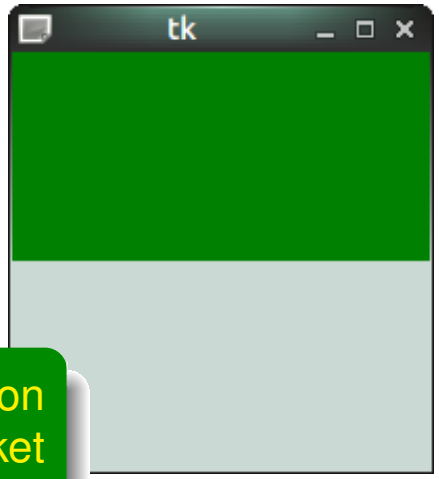
Létrehozzuk a fő ablakot.
Létrehozzuk és felparaméterezzük a **Frame**-t.
A **Frame** elhelyezése az ablakban.
A **pack** függvény egy geometiai szervező függvény, még lesz róla szó.

Frame widget

A **Frame** widget arra szolgál, hogy más widget-eket geometriailag könnyen csoportba tudjunk rendezni.

Példa:

```
        ⋮
mw=Tk()
mw.geometry('200x200+100+100')
fr=Frame(mw,width=200,height=100,bg='green')
fr.pack()
        ⋮
```



Létrehozzuk a fő ablakot.
Létrehozzuk és felparaméterezzük a
A **Frame** elhelyezése az ablakban.
A **pack** függvény egy geometiai szer
szó.

A **Frame** widget minazon függvényeket és paramétereket ismeri, amelyek értelmezhetőek rá.

PanedWindow

A **`PanedWindow`** widget egy rácsszerű elhelyezést tesz lehetővé.

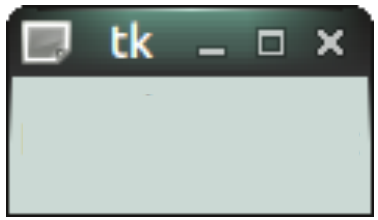
PanedWindow

A **`PanedWindow`** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,text="right")
m1.add(right)
      ⋮
```



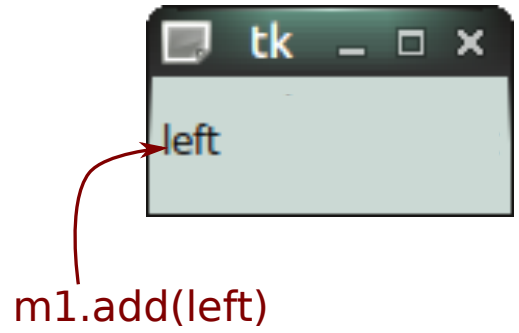
A **PanedWindow** létrehozása.

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,
m1.add(right)
      ⋮
```

A **Label** widget később jön.

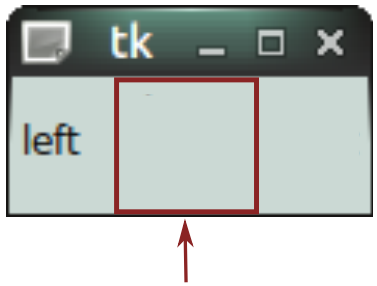


A **PanedWindow** létrehozása.
Az első widget.

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,text="right")
m1.add(right)
      ⋮
```

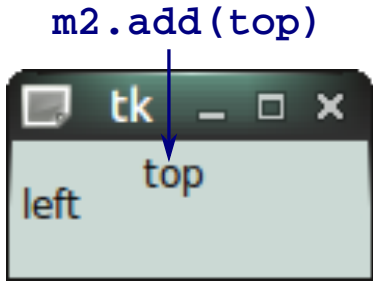


A **PanedWindow** létrehozása.
Az első widget.
A második **PanedWindow**, mint widget (vertikális).

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,text="right")
m1.add(right)
      ⋮
```

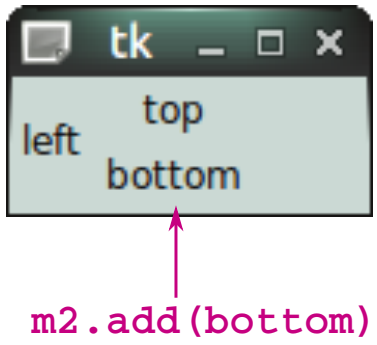


A **PanedWindow** létrehozása.
Az első widget.
A második **PanedWindow**, mint widget (vertikális).
A függőleges első widget.

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,text="right")
m1.add(right)
      ⋮
```



A **PanedWindow** létrehozása.
Az első widget.
A második **PanedWindow**, mint widget (vertikális).
A függőleges első widget.
A függőleges második widget.

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,text="right")
m1.add(right)
      ⋮
```

A **PanedWindow** létrehozása.
Az első widget.
A második **PanedWindow**, mint widget (vertikális).
A függőleges első widget.
A függőleges második widget.
A vízszintes harmadik widget.

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw,orient=HORIZONTAL)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,text="right")
m1.add(right)
      ⋮
```

A **PanedWindow** létrehozása.
Az első widget.
A második **PanedWindow**, mint widget (vertikális).
A függőleges első widget.
A függőleges második widget.
A vízszintes harmadik widget.
A horizontális irány, ez a default.

PanedWindow

A **PanedWindow** widget egy rácsszerű elhelyezést tesz lehetővé.
Példa:

```
      ⋮
m1=PanedWindow(mw)
m1.pack(fill=BOTH,expand=1)
left=Label(m1,text="left")
m1.add(left)
m2=PanedWindow(m1,orient=VERTICAL)
m1.add(m2)
top=Label(m2,text="top")
m2.add(top)
bottom=Label(m2,text="bottom")
m2.add(bottom)
right=Label(m1,text="right")
m1.add(right)
      ⋮
```

A **PanedWindow** létrehozása.
Az első widget.
A második **PanedWindow**, mint widget (vertikális).
A függőleges első widget.
A függőleges második widget.
A vízszintes harmadik widget.
A horizontális irány, ez a default.
A függőleges irány.