

Kurs rozszerzony języka Python

Wykład 12.

Marcin Młotkowski

11 stycznia 2023

Plan wykładu

- 1 GUI w Pythonie: GTK+
 - Wprowadzenie do GTK+
 - PyGTK
- 2 Przykład aplikacji
 - Okno główne aplikacji
 - Pakowanie kontroltek
 - Obsługa kontroltek
- 3 Wykorzystanie Glade

Plan wykładu

- 1 GUI w Pythonie: GTK+
 - Wprowadzenie do GTK+
 - PyGTK
- 2 Przykład aplikacji
 - Okno główne aplikacji
 - Pakowanie kontrolek
 - Obsługa kontrolek
- 3 Wykorzystanie Glade

Biblioteki okienkowe w Pythonie

- curses: interfejs tekstowy
- Tkinter (Tk interface): biblioteka okienkowa Tk + Tix (Tk extension)
- Pygtk, pygnome: API do środowiska Gtk/Gnome
- PyQt: API do QT
- wxWindows
- OpenGL
- PyWin32

GTK+/GNU

GTK+

The **G**IMP **T**oolkit

GNOME

GNU Network Object Model Environment

Moduły towarzyszące

Moduły współistniejące z GTK+

- GObject
- ATK
- Pango
- Cairo
- Glade

Środowisko GTK+

Elementy składowe

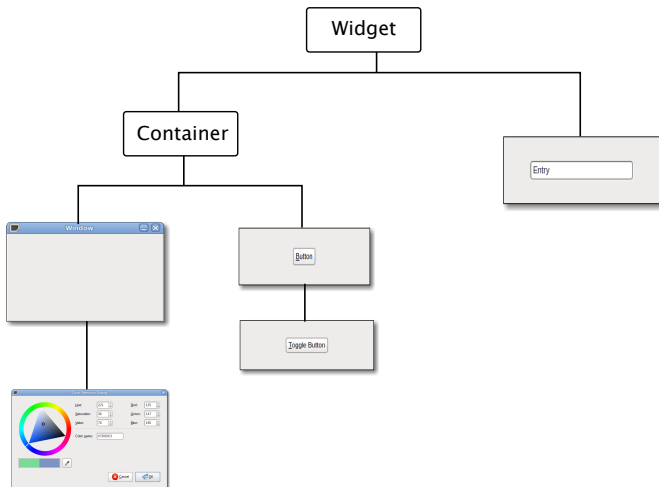
- okna;
- kontrolki;
- zdarzenia

Okna i kontrolki

Rola kontrolki (bardzo nieformalnie):

- kontrolka ma być widoczna (najlepiej ładna);
- kontrolka czasem ma reagować, np. na kliknięcie myszą;
- kontrolka czasem może zawierać inne kontrolki.

Hierarchia kontrolek (uproszczone)



GTK+ i Python

Biblioteka PyGTK

PyGTK: podstawowe elementy

Biblioteka się nazywa gtk.

PyGTK: podstawowe elementy

Biblioteka się nazywa gtk.

W bibliotece są odpowiednie klasy Window, Entry, Button etc.

Plan wykładu

- 1 GUI w Pythonie: GTK+
 - Wprowadzenie do GTK+
 - PyGTK
- 2 Przykład aplikacji
 - Okno główne aplikacji
 - Pakowanie kontroltek
 - Obsługa kontroltek
- 3 Wykorzystanie Glade

Kalkulator graficzny

Zadanie: prosty kalkulator obliczający zadane wyrażenie
artymetyczne.

Nagłówki

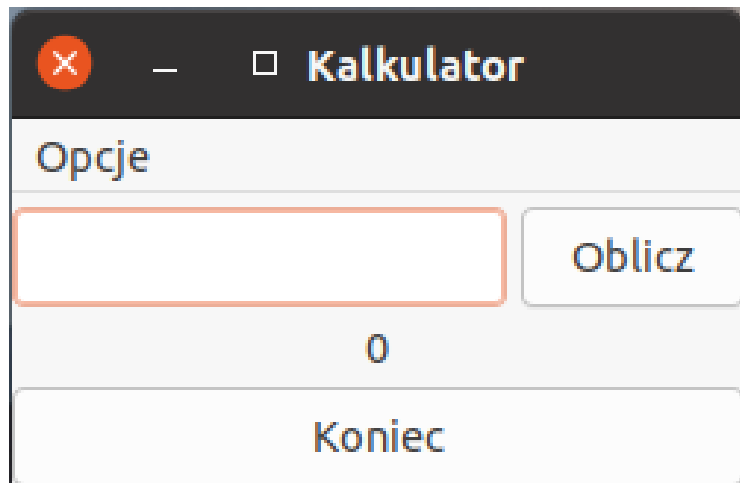
```
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk, Gdk
```

Budowanie okna

```
class Kalkulator(Gtk.Window):  
    def __init__(self):  
        super(Kalkulator, self).__init__()  
  
        self.set_title("Kalkulator Gtk")  
  
        self.connect("destroy", Gtk.main_quit)  
        self.kontrolki()  
        self.show_all()
```


Uruchomienie aplikacji

```
r = Kalkulator()  
Gtk.main()
```



The image shows a window titled "Kalkulator" (Calculator) with a dark header bar. The window contains a light gray area with the text "Opcje" (Options) at the top. Below this is a large white rectangular input field with an orange border. To the right of the input field is a button labeled "Oblicz" (Calculate). Below the input field, the number "0" is displayed. At the bottom of the window is a large button labeled "Koniec" (End).

Kalkulator

Opcje

Oblicz

0

Koniec

Dokładanie kontroltek

- Window jest kontrolką "widzialną";
- Window jest też kontenerem, można wstawić element;
- do Window można wstawić tylko jeden element.

Pudełka

Do układania elementów służą pudełka pionowe i poziome.

Pudełka

Do układania elementów służą pudełka pionowe i poziome.

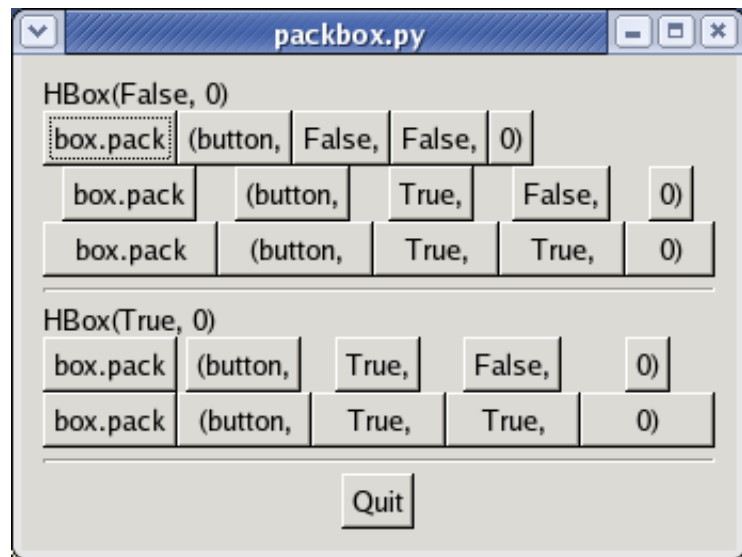
Gtk.Box: Gtk.VBox i Gtk.HBox

```
.pack_start(Widget, expand, fill, padding)  
.pack_end(Widget, expand, fill, padding)
```

gdzie

- `expand(bool)`: kontrolki włożone do pudełka mają wypełniającą całe pudełko, wypełnieniem jest pusta przestrzeń;
- `fill (bool)`: kontrolki wypełniają całą przestrzeń, ale przy okazji powiększane są kontrolki;
- `padding (int)`: dodatkowy odstęp od sąsiada

Ilustracja pakowania



Parę uwag dodatkowych

Pudełka można pakować w pudełka (pionowe w poziome, poziome w pionowe etc)

Parę uwag dodatkowych

Pudełka można pakować w pudełka (pionowe w poziome, poziome w pionowe etc)

Alternatywa: `Gtk.Grid`

"Kratka" komórek, do których wkłada się kontrolki.

Wprowadzanie tekstu

Gtk.Entry

- `.get_text()`: pobranie tekstu z bufora kontrolki;
- `.set_text("text")`: wyczyszczenie bufora i wstawienie tekstu;
- `.insert_text("insert", pos)`: wstawienie tekstu od *pos*.

Wyliczanie wyrażenia: kontrolki

```
self.entry = Gtk.Entry()  
self.oblicz = Gtk.Button("Oblicz")  
self.wynik = Gtk.Label("0")  
self.oblicz.connect('clicked', self.wylicz)
```

Wyliczanie wyrażenia: akcja

```
def wylicz(self, param):  
    wyrazenie = self.entry.get_text()  
    wartosc = eval(wyrazenie)  
    self.wynik.set_text(str(wartosc))
```

Przyciski

```
Gtk.Button('tekst')  
  
endb = Gtk.Button("Koniec")  
endb.connect("clicked", lambda x: self.destroy())
```

Zdarzenia (*events*)

Zdarzenie w GTK+

Informacja, że zaszło jakieś zdarzenie, np. kliknięcie przycisku, likwidacja jakiejś kontrolki.

Zdarzenia (*events*)

Zdarzenie w GTK+

Informacja, że zaszło jakieś zdarzenie, np. kliknięcie przycisku, likwidacja jakiejś kontrolki.

- Zdarzenia są związane z kontrolkami.
- Zdarzenia mają swoje nazwy.

Funkcje zwrotne (ang. *callbacks*)

Funkcje zwrotne to są funkcje wywoływane jako reakcja na zdarzenia.

Funkcje zwrotne (ang. *callbacks*)

Funkcje zwrotne to są funkcje wywoływane jako reakcja na zdarzenia.

Postać funkcji zwrotnej

```
def funkcja_zwrotna(kontrolka, dane)
```


Funkcje zwrotne (ang. *callbacks*)

Funkcje zwrotne to są funkcje wywoływane jako reakcja na zdarzenia.

Postać funkcji zwrotnej

```
def funkcja_zwrotna(kontrolka, dane)
```

Łączenie kontrolek, zdarzeń i funkcji zwrotnych

```
kontrolka.connect("nazwa zdarzenia", funkcja_zwrotna, dane)
```

Plan wykładu

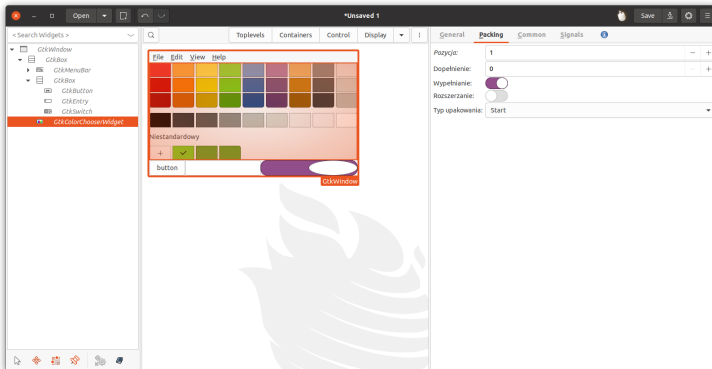
- 1 GUI w Pythonie: GTK+
 - Wprowadzenie do GTK+
 - PyGTK
- 2 Przykład aplikacji
 - Okno główne aplikacji
 - Pakowanie kontrolek
 - Obsługa kontrolek
- 3 Wykorzystanie Glade

Co to jest

Glade to graficzne narzędzie do projektowania interfejsów dla środowiska GTK+/GNOME.

Schemat działania (Glade-3)

- Glade produkuje plik XML, w którym jest opisany interfejs;
- Aplikacja "wczytuje" ten plik i buduje interfejs;
- Glade-3 jest niezależny od języka.



Szkic rozwiązania

```
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk

builder = Gtk.Builder()
builder.add_from_file("wyklad.glade")

window = builder.get_object("okno")
window.connect('destroy', Gtk.main_quit)
window.show_all()

Gtk.main()
```

Bardziej obiektowo

```
class Rysownik:
    def __init__(self):
        builder = Gtk.Builder()
        builder.add_from_file("wyklad.glade")
        self.window = builder.get_object("okno")
        builder.connect_signals(self)

    def on_window_destroy(self, widget, data=None): pass

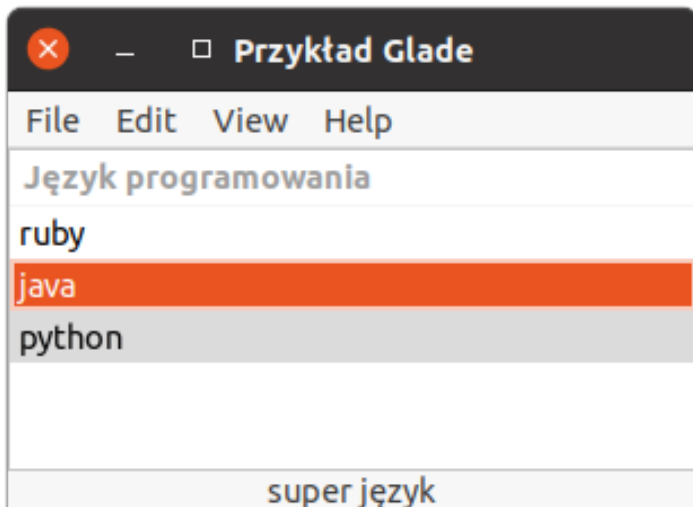
    def koniec(self, widget): pass

rysunek = Rysownik()
rysunek.window.show()
Gtk.main()
```

Przykład

Aplikacja która:

- wyświetla listę, np. listę języków programowania;
- o każdym języku wyświetla krótką informację



```
dane = { 'java' : 'super język', 'python': 'extra', 'ruby'  
info = builder.get_object('info')  
def on_change_select(widget):  
    model, treeiter = widget.get_selected()  
    if treeiter is not None:  
        jezyk = model[treeiter][0]  
        info.set_text(dane[jezyk])  
  
tree = builder.get_object('widoczne')  
select = tree.get_selection()  
select.connect('changed', on_change_select)
```