RAD in Python: Glade

Glade

- Graphical Interface Designer per GNOME
  - Memorizzazione della gerarchia dei widget in un file XML
  - Integrabile con IDE moderni (Anjuta, Eclipse)
  - Language-independent
- Storia
  - 1998: v0.1
  - 2006: v3.0
  - 2011: v3.10.1

Una applicazione testuale

- ESEMPI: tutcli.py

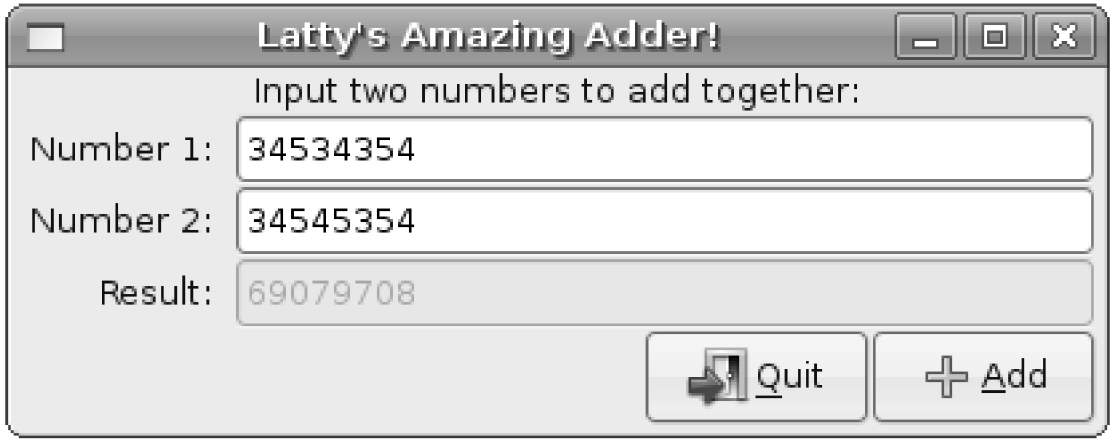
- **Partiamo da una applicazione testuale, tutcli.py**
- **Tale applicazione implementa una classe adder con i seguenti metodi**
  - **somma di due numeri interi**
  - **stampa il risultato**
- **Vogliamo trasformare questa applicazione testuale in una applicazione grafica, tutgui.py**
- **Vogliamo usare Glade per la creazione dell'interfaccia grafica**

Prima fase: sketch interfaccia

- L'interfaccia grafica di una applicazione va sempre abbozzata (sketched) e prototipata
  - su carta
  - tramite strumenti informatici
- Motivazioni
  - Progettazione dei diversi use case (scenari di utilizzo dell'applicazione); una GUI non è una sola schermata!
  - Piazzamento dei widget secondo criteri di usabilità (human interface guidelines)
- Strumenti per il GUI prototyping: http://c2.com/cgi/wiki?GuiPrototypingTools

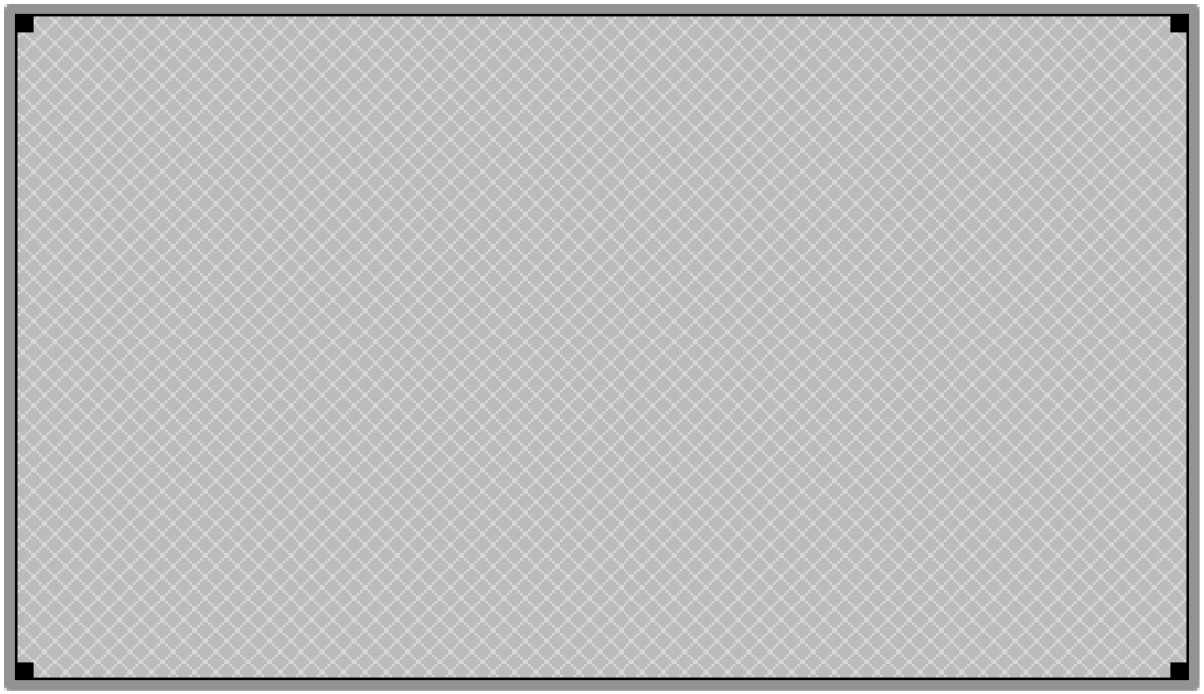
Prima fase: sketch interfaccia

- Nel nostro caso, non c'è molto da progettare...
  - Due caselle di testo contenenti i numeri da sommare
  - Una label contenente il risultato
  - Un bottone per l'operazione di somma
  - Un bottone per l'operazione di quit



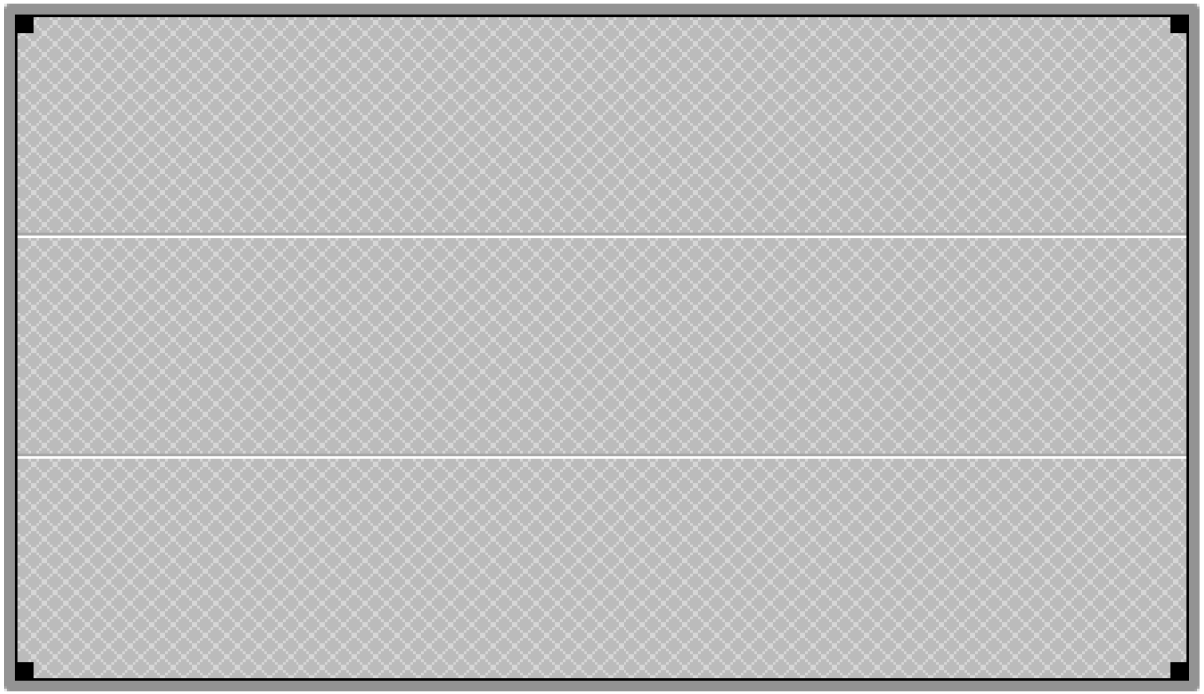
Seconda fase: creazione GUI

- Apriamo Glade eseguendo il comando glade
- Scegliamo il progetto di tipo “Glade” (non GtkBuilder)
- Creiamo una nuova finestra selezionando la prima icona sotto il tab “Livelli Principali”



Seconda fase: creazione GUI

- Inseriamo una vertical box di tre elementi all'interno della finestra, selezionando la seconda icona sotto il tab “Contenitori”

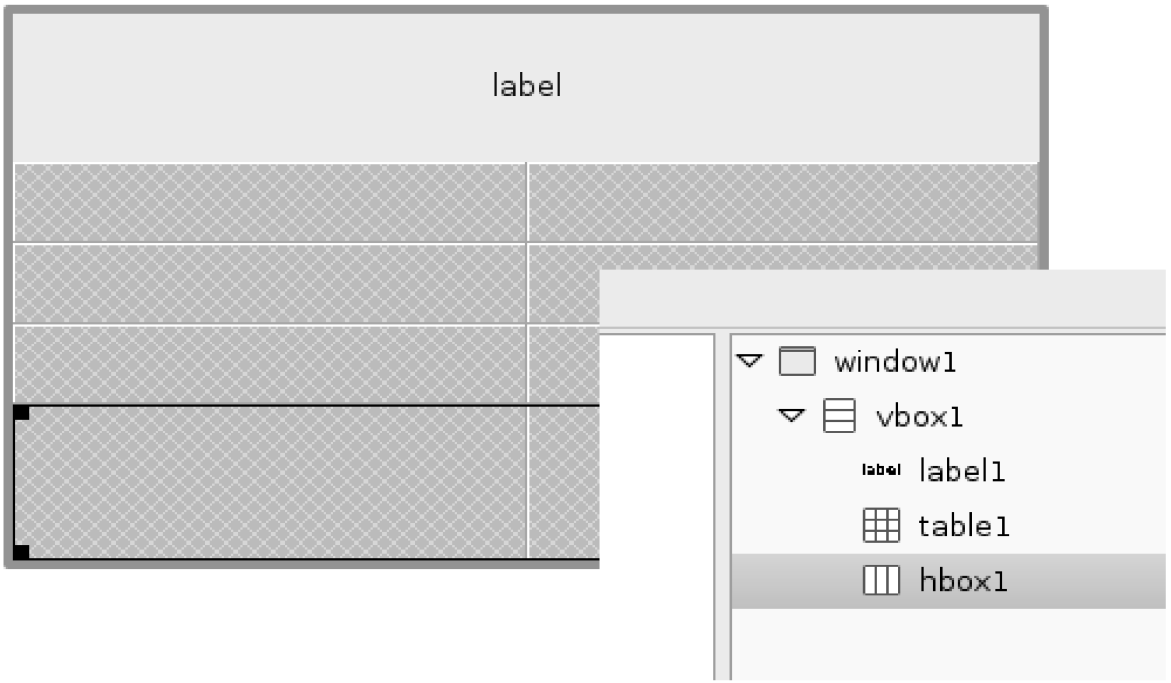


Seconda fase: creazione GUI

- Inseriamo una label nella prima box
  - selezionando la prima icona nella terza fila sotto il tab “Controllo e visualizzazione”
  - cliccando il primo elemento della box
- Inseriamo una tabella 2 colonne x 3 righe nella seconda box
  - selezionando la terza icona nella prima fila sotto il tab “Contenitori”
  - cliccando il secondo elemento della box

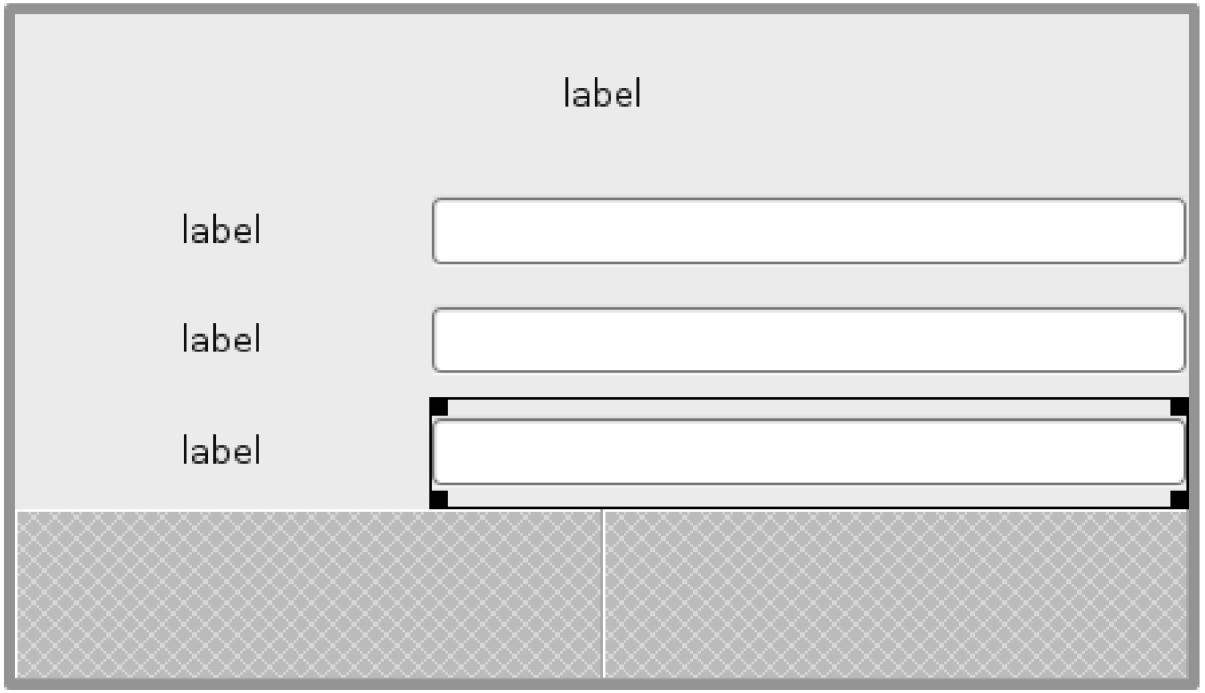
Seconda fase: creazione GUI

- Inseriamo una box orizzontale nella terza box
  - selezionando la prima icona nella prima fila sotto il tab “Contenitori”
  - cliccando il terzo elemento della box



Seconda fase: creazione GUI

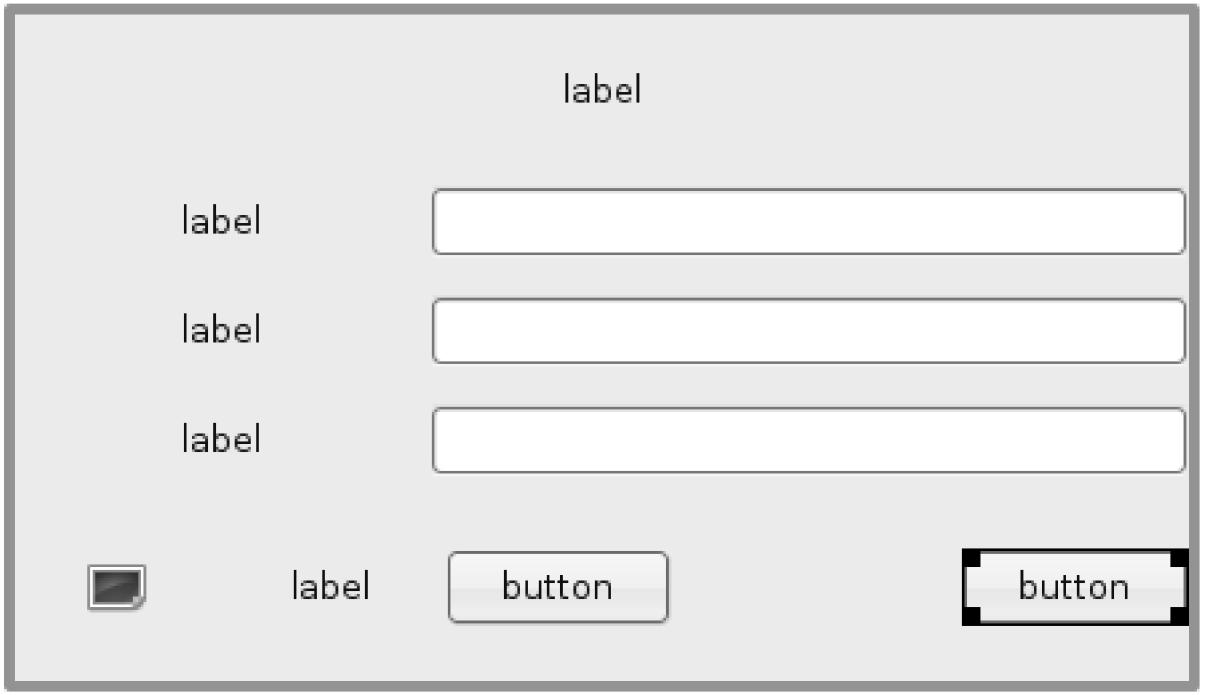
- Consideriamo la tabella 2x3
- Inseriamo
  - tre label di testo negli elementi di sinistra
  - tre caselle di testo negli elementi di destra



Seconda fase: creazione GUI

- Consideriamo la box orizzontale
- Inseriamo
  - una box orizzontale nell'elemento di sinistra, contenente
    - un'immagine alla sinistra
    - una label alla destra
  - una casella di pulsanti orizzontali con due elementi nell'elemento di destra, contenente
  - un bottone alla sinistra
  - un bottone alla destra
- La casella di pulsanti orizzontali è raggiungibile tramite la quinta icona nella seconda riga sotto il tab “Contenitori”

Seconda fase: creazione GUI



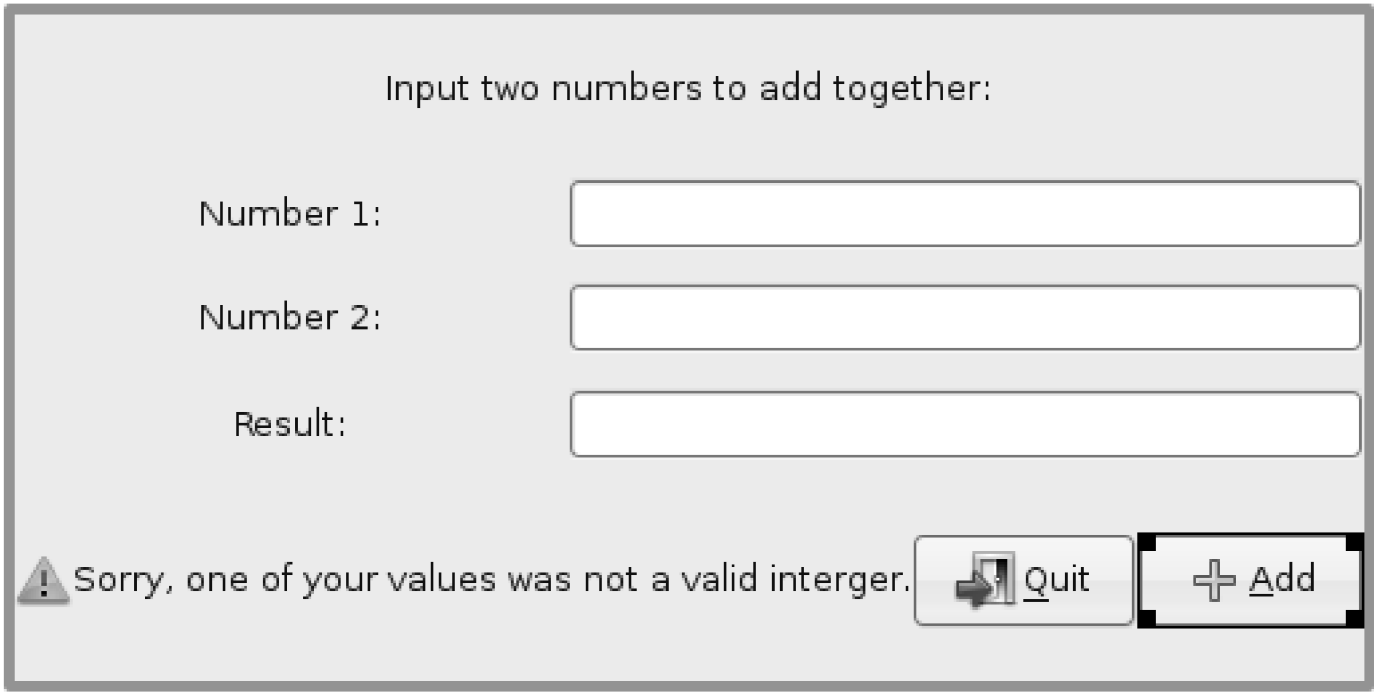
Seconda fase: creazione GUI

- Popoliamo le label dei due numeri e del risultato con del testo
- Usiamo l'area delle proprietà alla destra dell'area di design; cerchiamo l'elemento “Etichetta” e modifichiamolo
- Selezioniamo l'immagine ed individuiamo, nell'area delle proprietà, il campo “ID dell'oggetto nello stock”
- Scegliamo l'icona “Avvertimento”
- Scriviamo un testo di warning nella label vicina all'immagine

Seconda fase: creazione GUI

- Selezioniamo il bottone più a sinistra nella casella dei pulsanti
- Abilitiamo il radio button “Pulsante stock” (che attiva una delle azioni predefinite)
- Selezioniamo l'icona “Esci”
- Selezioniamo il bottone più a destra nella casella dei pulsanti
- Abilitiamo il radio button “Pulsante stock”
- Selezioniamo l'icona “Aggiungi”

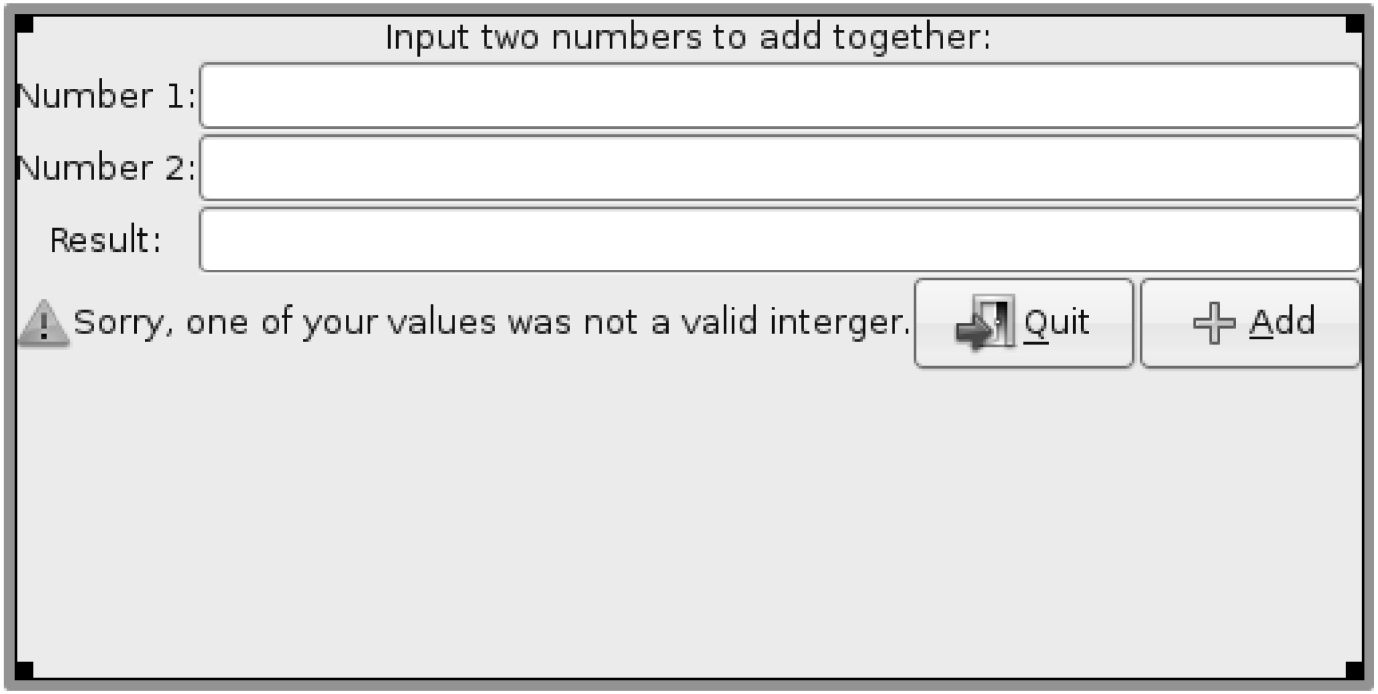
Seconda fase: creazione GUI



Seconda fase: creazione GUI

- Applichiamo dei correttivi estetici all'interfaccia
- Le label e le caselle di testo sono troppo alte perché si espandono in verticale sulla dimensione della box che le contiene
- Selezioniamole, andiamo nel tab “Posizionamento”, identifichiamo l'elemento “Opzioni verticali” e togliamo “espandi”
- Le label Number1, Number2, Result sono troppo larghe → togliamo anche l'”espandi” orizzontale

Seconda fase: creazione GUI



Seconda fase: creazione GUI

- Selezioniamo l'elemento window1 (la finestra contenitrice) nell'area in alto a destra
- Selezioniamo il tab “Generale” in basso a destra
- Diamo un identificatore più sensato alla finestra: scriviamo “windowMain” nel campo “Nome”
- Diamo un titolo alla finestra: “Latty's Amazing Adder!”

Seconda fase: creazione GUI

- Selezioniamo il tab “Segnali” della finestra contenitrice
- Selezioniamo l'evento GtkObject → destroy
- Clicchiamo sulla tendina nella colonna “Gestore”
- Selezioniamo l'handler on_windowMain_destroy
- Abbiamo appena scelto il callback da invocare quando chiudiamo la finestra

Seconda fase: creazione GUI

- Diamo un identificatore sensato al bottone di quit: buttonQuit
- Scegliamo l'handler dell'evento “clicked” per il bottone Quit: on_buttonQuit_clicked
  - Tale metodo sarà mappato al metodo quit() della classe adder
- Diamo un identificatore sensato al bottone di Add: buttonAdd
- Scegliamo l'handler dell'evento “clicked” per il bottone Add: on_buttonAdd_clicked
  - Tale metodo sarà mappato al metodo add() della classe adder

Seconda fase: creazione GUI

- Diamo un identificatore sensato alle caselle di testo
  - EntryNumber1, entryNumber2, entryResult
- Diamo un identificatore sensato alla label di warning
  - hboxWarning

Seconda fase: creazione GUI

- Rendiamo inizialmente invisibile il warning
- Clicchiamo sulla label contenente la frase di warning
- Selezioniamo il tab “Comuni” nell'area a destra
- Cerchiamo l'opzione “Visibile” ed impostiamola a “No”
- Non vogliamo rendere editabile la casella di testo “Result”
- Clicchiamo sulla terza casella di testo
- Selezioniamo il tab “Comuni” nell'area a destra
- Cerchiamo l'opzione “Sensibile” ed impostiamola a “No”

Seconda fase: creazione GUI

- Rendiamo inizialmente visibile l'intera finestra
- Identifichiamo il widget windowMain nel tab in alto a destra
- Selezioniamo il tab “Comuni” nell'area a destra
- Cerchiamo l'opzione “Visibile” ed impostiamola a “Si”
- Abbiamo finito!
  - Con la GUI, almeno...

Terza fase: modifica codice

- ESEMPI: tutgui.py

- **Modifichiamo ora il codice sorgente dell'applicazione in modo tale da**
  - **inizializzare il modulo software glade**
  - **caricare il file XML contenente la descrizione della GUI**
  - **aggiungere i callback per gli eventi abilitati nella GUI**

Terza fase: modifica codice

- ESEMPI: tutgui.py

- **Inizializzazione modulo glade**
- **Deve essere importato il modulo gtk.glade**

```
import sys
try:
    import pygtk
    pygtk.require("2.0")
except:
    pass
try:
    import gtk
    import gtk.glade
except:
    print("GTK Not Availible")
    sys.exit(1)
```

Terza fase: modifica codice

- ESEMPI: tutgui.py

- **Aggiungiamo una classe per l'attivazione della interfaccia grafica**

```
class guiadder:
    wTree = None
    def __init__( self ):
        self.wTree = gtk.glade.XML( "main.glade" )
        dic = {
            "on_buttonQuit_clicked" : self.quit,
            "on_buttonAdd_clicked" : self.add,
            "on_windowMain_destroy" : self.quit,
        }
        self.wTree.signal_autoconnect( dic )
        gtk.main()
```

Struttura gerarchica dei widget (Window Tree).

Traduzione del file XML in un Window Tree.

Associazione dei segnali ai callback.

Terza fase: modifica codice

- ESEMPI: tutgui.py

- **Aggiungiamo alla classe adder il callback add() per l'evento clicked del bottone buttonAdd**
- **wTree.get_widget(name) → handle a widget di nome name**

```
def add(self, widget):
        try:
            Thistime =
adder( self.wTree.get_widget("entryNumber1").get_text(),
self.wTree.get_widget("entryNumber2").get_text() )
        except ValueError:
            self.wTree.get_widget("hboxWarning").show()
self.wTree.get_widget("entryResult").set_text("ERROR")
            return 0
        self.wTree.get_widget("hboxWarning").hide()
self.wTree.get_widget("entryResult").set_text(thistime.giveResult())
```

Terza fase: modifica codice

- ESEMPI: tutgui.py

- **Aggiungiamo alla classe adder il callback quit() per l'evento clicked del bottone buttonQuit**

```
def quit(self, widget):
        sys.exit(0)
```