Python GTK

1. Catatan....3
2. GTK+ dan PyGTK....4
3. Hello World....6
4. Layout Container....8
gtk.Box....8
gtk.Table....10
gtk.Fixed....12
5. Signal dan callback....14
Koneksi signal-callback....14
Koneksi event-callback....21
6. Beragam widget....26
gtk.Label....26
gtk.Button....26
gtk.ToggleButton....27
gtk.CheckButton....27
gtk.RadioButton....27
gtk.SpinButton....27
gtk.ColorButton....28
gtk.FontButton....28
gtk.FileChooserButton....29
gtk.Entry....29
gtk.ComboBox....30
gtk.Frame....31
gtk.Image....31
gtk.Tooltips....32
gtk.Scale....32
gtk.ProgressBar....33
gtk.ScrolledWindow....33
gtk.TextView....34
gtk.Statusbar....34
gtk.Expander....35
gtk.ButtonBox....35
gtk.Paned....36
gtk.Notebook....37
gtk.Calendar....38
gtk.Menu, gtk.MenuItem, gtk.MenuBar....39
gtk.Toolbar, gtk.Toolitem....41
7. Dialog....43
gtk.Dialog....43
gtk.MessageDialog....45
8. Lain-lain....49

Timeout....49
Idle....50
Events pending....51

1. Catatan

a) Dianjurkan untuk membaca materi Python Dasar terlebih dahulu. Aturan penulisan kode dan OOP dapat dibaca pada materi Python Dasar.
b) GTK+ yang digunakan adalah 2.x:
   1. PyGTK yang digunakan adalah 2.x.
   2. Fitur spesifik platform tidak dibahas.
   3. Pemrograman GUI sangatlah luas. Training hanya akan membahas dasar-dasar pemrograman GUI dengan GTK+/PyGTK. Hanya sebagian widget yang akan dibahas.
   4. Referensi class (termasuk hirarki) dapat dibaca pada dokumentasi referensi pustaka.
   5. Diharapkan untuk membaca dokumentasi lain yang mendukung.
   6. Pemrograman menggunakan paradigma object-oriented (kecuali pada potongan kode, untuk kesederhanaan), dengan kesepakatan:
      a) Setiap window/dialog dituliskan dalam class masing-masing.
      b) Constructor class dapat digunakan untuk mengatur widget.
      c) Nama class untuk window utama adalah Main()
c) Pengembangan tetap mengacu pada multiplatform (setidaknya Windows dan Linux), kecuali disebutkan berbeda.
d) Pembahasan beragam widget (Bab 6) hanya akan membahas contoh sederhana penggunaan widget. Selengkapnya, peserta bisa merujuk ke dokumentasi.

2. GTK+ dan PyGTK

a) GTK+ (GIMP Tool Kit): GUI toolkit multiplatform yang lengkap, terdokumentasi baik, tersedia binding ke berbagai bahasa pemrograman (default C, dengan ide berorientasi objek), telah digunakan di berbagai proyek besar (contoh: GNOME) dan dilisensikan di bawah LGPL (free software, dapat digunakan pada aplikasi proprietary).
b) GTK+ dibangun di atas GDK (GIMP Drawing Kit), yang merupakan wrapper untuk fungsi drawing low level platform.
c) PyGTK: modul Python yang menyediakan interface ke GTK+. PyGTK merupakan binding official GNOME.
d) Informasi selengkapnya:
   1. GTK+: http://gtk.org
   2. PyGTK: http://pygtk.org
e) Untuk bekerja dengan PyGTK:
   1. import pygtk
   2. pygtk.require('2.0')
      a) Kita ingin menggunakan PyGTK versi 2.x dan mencegah digunakannya versi lain apabila terinstall.
   3. import gtk
f) Dalam bekerja dengan beragam widget GTK+, developer bisa mempergunakan stock GTK+ yang menyediakan ID icon siap pakai (dan mengandung informasi stock id, string label, modifier, keyval dan translation domain). Berikut adalah tabel stock PyGTK 2.10 (GTK+ versi 2.10).

gtk.STOCK_ABOUT	gtk.STOCK_ADD	gtk.STOCK_APPLY	gtk.STOCK_BOLD
gtk.STOCK_CANCEL	gtk.STOCK_CDROM	gtk.STOCK_CLEAR	gtk.STOCK_CLOSE
gtk.STOCK_COLOR_PICKER	gtk.STOCK_CONVERT	gtk.STOCK_CONNECT	gtk.STOCK_COPY
gtk.STOCK_CUT	gtk.STOCK_DELETE	gtk.STOCK_DIALOG_AUTHENTICATION	gtk.STOCK_DIALOG_ERROR
gtk.STOCK_DIALOG_INFO	gtk.STOCK_DIALOG_QUESTION	gtk.STOCK_DIALOG_WARNING	gtk.STOCK_DIRECTORY
gtk.STOCK_DISCONNECT	gtk.STOCK_DND	gtk.STOCK_DND_MULTIPLE	gtk.STOCK_EDIT
gtk.STOCK_EXECUTE	gtk.STOCK_FILE	gtk.STOCK_FIND	gtk.STOCK_FIND_AND_REPLACE
gtk.STOCK_FLOPPY	gtk.STOCK_FULLSCREEN	gtk.STOCK_GOTO_BOTTOM	gtk.STOCK_GOTO_FIRST
gtk.STOCK_GOTO_LAST	gtk.STOCK_GOTO_TOP	gtk.STOCK_GO_BACK	gtk.STOCK_GO_DOWN
gtk.STOCK_GO_FORWARD	gtk.STOCK_GO_UP	gtk.STOCK_HARDDISK	gtk.STOCK_HELP
gtk.STOCK_HOME	gtk.STOCK_INDENT	gtk.STOCK_INDEX	gtk.STOCK_INFO
gtk.STOCK_ITALIC	gtk.STOCK_JUMP_TO	gtk.STOCK_JUSTIFY_	gtk.STOCK_JUSTIFY_

		CENTER	FILL
gtk.STOCK_JUSTIFY_LEFT	gtk.STOCK_JUSTIFY_RIGHT	gtk.STOCK_LEAVE_FULLSCREEN	gtk.STOCK_MEDIA_FORWARD
gtk.STOCK_MEDIA_NEXT	gtk.STOCK_MEDIA_PAUSE	gtk.STOCK_MEDIA_PLAY	gtk.STOCK_MEDIA_PREVIOUS
gtk.STOCK_MEDIA_RECORD	gtk.STOCK_MEDIA_REWIND	gtk.STOCK_MEDIA_STOP	gtk.STOCK_MISSING_IMAGE
gtk.STOCK_NETWORK	gtk.STOCK_NEW	gtk.STOCK_NO	gtk.STOCK_OK
gtk.STOCK_OPEN	gtk.STOCK_ORIENTATION_LANDSCAPE	gtk.STOCK_ORIENTATION_PORTRAIT	gtk.STOCK_ORIENTATION_REVERSE_LANDSCAPE
gtk.STOCK_ORIENTATION_REVERSE_PORTRAIT	gtk.STOCK_PASTE	gtk.STOCK_PREFERENCES	gtk.STOCK_PRINT
gtk.STOCK_PRINT_PREVIEW	gtk.STOCK_PROPERTIES	gtk.STOCK_QUIT	gtk.STOCK_REDO
gtk.STOCK_REFRESH	gtk.STOCK_REMOVE	gtk.STOCK_REVERT_TO_SAVED	gtk.STOCK_SAVE
gtk.STOCK_SAVE_AS	gtk.STOCK_SELECT_ALL	gtk.STOCK_SELECT_COLOR	gtk.STOCK_SELECT_FONT
gtk.STOCK_SORT_ASCENDING	gtk.STOCK_SORT_DESCENDING	gtk.STOCK_SPELL_CHECK	gtk.STOCK_STOP
gtk.STOCK_STRIKETHROUGH	gtk.STOCK_UNDELETE	gtk.STOCK_UNDERLINE	gtk.STOCK_UNDO
gtk.STOCK_UNINDENT	gtk.STOCK_YES	gtk.STOCK_ZOOM_100	gtk.STOCK_ZOOM_FIT
gtk.STOCK_ZOOM_FIT	gtk.STOCK_ZOOM_OUT		

3. Hello World

Dalam program contoh hello.py, sebuah window kosong dengan title 'Hello World' akan dibuat dan ditampilkan.

hello.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Hello World')

        self.window.show_all()

app = Main()
gtk.main()
```

Jalankanlah dengan perintah:

```
$ python hello.py
```

Untuk keluar dari program, tombol close pada window tidak dapat digunakan. Kembali ke shell dan tekanlah kombinasi tombol CTRL-C. Pesan berikut bisa diabaikan:

```
Traceback (most recent call last):
  File "hello.py", line 15, in <module>
    gtk.main()
KeyboardInterrupt
```

Penjelasan:

- Untuk membuat window, gunakan: gtk.Window(type=gtk.WINDOW_TOPLEVEL)
  - Argumen type bisa berupa:
    - gtk.WINDOW_TOPLEVEL: window toplevel
    - gtk.WINDOW_POPUP: dapat digunakan untuk window pop-up seperti pop-up menu atau tooltip.
  - Mengembalikan objek gtk.Window
- Untuk mengatur title Window, gunakan: gtk.Window.set_title(title)
- Agar window dapat ditampilkan, gunakan: gtk.Widget.show_all().
  - Perhatikanlah bahwa show_all() adalah method gtk.Widget. Class gtk.Window diturunkan dari: gobject.GObject -> gtk.Object -> gtk.Widget -> gtk.Container -> gtk.Bin.
  - Method show_all() akan menampilkan widget secara rekursif, termasuk semua child widget.
  - Untuk widget tunggal, method show() bisa dipergunakan.
  - Sebelum di-show, widget tidak akan ditampilkan.

– Selengkapnya, bacalah class reference untuk class-class terkait.

4. Layout Container

Container merupakan widget yang dapat mengandung widget lain (child widget). Beberapa widget container, selain mengandung widget lain, dapat pula melakukan layout (layout container). Dengan menggunakan layout container, peletakan berbagai widget dalam satu container dapat dilakukan dengan mudah.

Sebagai catatan, tidak semua layout dibahas. Pemilihan layout juga bergantung pada preferensi developer dan user interface yang dirancang. Dalam materi training ini selanjutnya, karena jumlah dan posisi widget yang sederhana, gtk.Box yang akan dipergunakan. Namun, dalam penggunaan sehari-hari, dengan user interface yang rumit, gtk.Table atau kombinasi antara gtk.Box dan gtk.Table dipergunakan.

Untuk keluar dari semua contoh program di dalam bab ini, tombol close pada window tidak dapat digunakan. Kembali ke shell dan tekanlah kombinasi tombol CTRL-C.

gtk.Box

Class gtk.Box merupakan base class abstrak untuk container box:
- gtk.HBox: melayout child widget secara horizontal
- gtk.VBox: melayout child widget secara vertikal

Contoh penggunaan dapat dilihat pada hello_hbox.py dan hello_vbox.py.

Constructor:
```
gtk.HBox(homogeneous=False, spacing=0)

gtk.VBox(homogeneous=False, spacing=0)
```

- homogeneous: menentukan apakah child widget diberikan alokasi ruang yang sama.
- spacing: ruang kosong (horizontal untuk Hbox dan vertikal untuk VBox) antara child widget, dalam pixel.

Bekerja dengan child widget:
- Menambahkan child widget dari awal, gunakan: gtk.Box.pack_start(child, expand=True, fill=True, padding=0)
  - child: child widget yang akan ditambahkan.
  - expand: menentukan apakah child widget diberikan ruang kosong ekstra. Ruang kosong ekstra ini akan dibagi rata antara semua child widget yang ditambahkan dengan opsi ini.
  - fill: menentukan apakah ruang kosong ekstra yang diberikan (lewat expand) benar-benar dialokasikan untuk child widget, dan bukan sekedar digunakan sebagai padding. Opsi ini tidak memiliki efek apabila expand tidak diset True.
  - padding: ruang kosong tambahan (dalam pixel) antara child widget dan widget sekitarnya, di atas spacing gtk.Box.

- Menambahkan child widget dari akhir, gunakan: gtk.Box.pack_end(child, expand=True, fill=True, padding=0).
- Penambahan child bisa dilakukan bersarang. Dengan demikian, gtk.HBox bisa ditambahkan ke gtk.HBox lain atau gtk.VBox, dan seterusnya.

hello_hbox.py:
```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Hello World')

        self.hbox = gtk.HBox()

        self.entry = gtk.Entry()
        self.button = gtk.Button('_uppercase')

        self.hbox.pack_start(self.entry)
        self.hbox.pack_start(self.button)

        self.window.add(self.hbox)

        self.window.show_all()

app = Main()
gtk.main()
```

hello_vbox.py:
```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Hello World')

        self.vbox = gtk.VBox()

        self.entry = gtk.Entry()
        self.button = gtk.Button('_uppercase')

        self.vbox.pack_start(self.entry)
        self.vbox.pack_start(self.button)
```

```
        self.window.add(self.vbox)

        self.window.show_all()

app = Main()
gtk.main()
```

Catatan contoh:
- Contoh hello_hbox.py dan hello_vbox.py mempergunakan widget gtk.Button dan gtk.Entry. Kedua widget tersebut bisa diabaikan terlebih dahulu, dan fokus diberikan pada gtk.HBox atau gtk.VBox.
- Lebih lanjut tentang gtk.Button dan gtk.Entry dapat dilihat pada pembahasan Beragam Widget (Bab 6).
- Widget-widget ditambahkan pada gtk.HBox atau gtk.VBox. Setelah itu, gtk.HBox atau gtk.VBox yang ditambahkan ke gtk.Window.

gtk.Table

Widget-widget dapat diatur dalam baris dan kolom dengan layout container gtk.Table. Selain peletakan pada baris dan kolom, ukuran widget juga dapat diset dalam ukuran baris dan kolom. Jumlah baris dan kolom gtk.Table bisa ditentukan dan diresize.

Contoh penggunaan dapat dilihat pada hello_table.py.

Constructor:
```
gtk.Table(rows=1, columns=1, homogeneous=False)
```

- rows: jumlah baris
- columns: jumlah kolom
- homogeneous: menentukan apakah semua cell dalam tabel akan memiliki ukuran seperti ukuran cell terbesar.

Bekerja dengan child widget:
- Untuk menambahkan child widget, gunakan: gtk.Table. attach(child, left_attach, right_attach, top_attach, bottom_attach, xoptions=gtk.EXPAND|gtk.FILL, yoptions=gtk.EXPAND|gtk.FILL, xpadding=0, ypadding=0)
  - child: widget yang akan ditambahkan.
  - left_attach: kolom untuk menempatkan sisi kiri child widget. Kolom dimulai dari 0.
  - right_attach: kolom untuk menempatkan sisi kanan child widget. Kolom dimulai dari 0.
  - top_attach: baris untuk menempatkan sisi atas child widget. Baris dimulai dari 0.

  - bottom_attach: baris untuk menempatkan sisi bawah child widget. Baris dimulai dari 0.
  - xoptions: menentukan property child widget ketika tabel diresize secara horizontal. Dapat merupakan kombinasi gtk.EXPAND (ekspansi semua ruang kosong ekstra yang dialokasikan), gtk.SHRINK (disusutkan apabila cell tabel disusutkan) dan gtk.FILL (mengisi ruang kosong yang dialokasikan kepada widget dalam cell tabel).
  - yoptions: menentukan property child widget ketika tabel diresize secara vertikal. Lihat nilai untuk xoptions.
  - xpadding: padding sisi kiri dan kanan widget.
  - ypadding: padding sisi atas dan bawah widget.
- Untuk mengubah ukuran tabel, gunakan: gtk.Table.resize(rows, columns). Argumen rows dan columns masing-masing menentukan baris dan kolom yang baru.
- Untuk mengatur spasi baris: gtk.Table.set_row_spacing(row, spacing)
- Untuk mengatur spasi kolom: gtk.Table.set_col_spacing(column, spacing)

hello_table.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Hello World')

        self.table = gtk.Table(1,2)

        self.entry = gtk.Entry()
        self.button = gtk.Button('_uppercase')

        self.table.attach(self.entry, 0, 1, 0, 1)
        self.table.attach(self.button, 1, 2, 0, 1)

        self.window.add(self.table)

        self.window.show_all()

app = Main()
gtk.main()
```

Catatan contoh:
- Contoh hello_table.py mempergunakan widget gtk.Button dan gtk.Entry. Kedua widget tersebut bisa diabaikan terlebih dahulu, dan fokus diberikan pada gtk.Table.
- Lebih lanjut tentang gtk.Button dan gtk.Entry dapat dilihat pada pembahasan Beragam Widget (Bab 6).

- Widget-widget ditambahkan pada gtk.Table. Setelah itu, gtk.Table yang ditambahkan ke gtk.Window.

gtk.Fixed

Untuk mengatur posisi child widget pada koordinat tertentu, gunakanlah layout container gtk.Fixed. Walaupun relatif lebih mudah digunakan, penggunaan gtk.Fixed cukup merepotkan apabila container/window diresize, dan harus tetap mempertahankan rasio ukuran dan posisi child widget.

Contoh penggunaan dapat dilihat pada hello_fixed.py.

Constructor:
gtk.Fixed()

Bekerja dengan child widget:
- Untuk menambahkan child widget, gunakan: gtk.Fixed.put(widget, x, y):
  - widget: widget yang akan ditambahkan.
  - x: posisi x
  - y: posisi y
- Untuk memindahkan child widget ke lokasi lain, gunakanlah: gtk.Fixed.move(widget, x, y):
  - widget: widget yang akan ditambahkan.
  - x: posisi x baru
  - y: posisi y baru
- Nilai x=0 dan y=0 adalah sudut kiri atas.

hello_fixed.py:
```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Hello World')
        self.window.set_size_request(300,100)

        self.fixed = gtk.Fixed()

        self.entry = gtk.Entry()
        self.button = gtk.Button('_uppercase')

        self.fixed.put(self.entry, 10, 10)
        self.fixed.put(self.button, 200, 10)

        self.window.add(self.fixed)
```

```
        self.window.show_all()

app = Main()
gtk.main()
```

Catatan contoh:

- Contoh hello_fixed.py mempergunakan widget gtk.Button dan gtk.Entry. Kedua widget tersebut bisa diabaikan terlebih dahulu, dan fokus diberikan pada gtk.Fixed.
- Lebih lanjut tentang gtk.Button dan gtk.Entry dapat dilihat pada pembahasan Beragam Widget (Bab 6).
- Widget-widget ditambahkan pada gtk.Fixed sesuai posisi yang ditentukan. Setelah itu, gtk.Fixed yang ditambahkan ke gtk.Window.
- Window utama diresize untuk memudahkan demonstrasi gtk.Fixed.

5. Signal dan callback

GTK+ adalah event driven toolkit. Dengan demikian, developer bisa mengatur agar event tertentu yang terjadi bisa dihandle. Contoh event sederhana adalah klik tombol mouse pada suatu tombol. Semua event yang terjadi dimonitor di dalam gtk.main().

Terminologi event di dalam GTK+ setidaknya bisa diterima sebagai:
- signal
- event (dari windowing system).

Widget bisa menerima signal, baik signal yang umum (diturunkan dari base class), ataupun signal spesifik widget tersebut. Daftar signal yang bisa diterima oleh widget beserta deskripsinya bisa didapatkan dari class reference.

Baik signal ataupun event bisa ditangani oleh fungsi tertentu, yang umumnya disebut sebagai callback di dalam GTK+.

Contoh-contoh program sebelumnya tidak bisa ditutup dengan klik pada tombol close window. Hal ini disebabkan karena program tidak menangani event yang terjadi.

Sebagai catatan, event pada GTK+ sangatlah luas dan kompleks. Selalulah merujuk ke referensi class untuk informasi selengkapnya.

Koneksi signal-callback

Koneksi
Untuk mengatur agar signal tertentu dihubungkan ke callback tertentu, gunakanlah method connect() widget:

```
gobject.GObject.connect(detailed_signal, handler, ...).
```

- detailed_signal: string berisikan nama signal
- handler: fungsi atau method callback
- ...: argumen opsional tambahan
- Nilai kembalian adalah handler ID (integer)

Catatan:
- Berbagai signal bisa diset agar ditangani.
- Banyak callback bisa dihubungkan ke satu signal tertentu, dan akan dikerjakan sesuai urutan koneksi.

Definisi callback
Callback secara umum dapat didefinisikan sebagai berikut:

```
def callback (widget, data)
```

Apabila argumen opsional tidak digunakan pada connect(), maka callback secara umum dapat didefinisikan sebagai berikut:

```
def callback (widget, data=None)
atau
def callback (widget)
```

Catatan:
- Callback harus menyesuaikan dengan jumlah argumen opsional yang diberikan.
- Apabila callback digunakan dalam class, maka argumen pertama pada definisi callback tetap adalah self.

Block dan unblock

Koneksi signal dan callback bisa diblock atau diunblock sementara waktu dengan:

```
gobject.GObject.handler_block(handler_id)
```

dan

```
gobject.GObject.handler_unblock(handler_id)
```

handler_id adalah nilai integer yang dikembalikan oleh connect().

Diskoneksi

Apabila signal tidak lagi ingin ditangani, maka method disconnect() widget bisa digunakan:

```
gobject.GObject.disconnect(handler_id)
```

handler_id adalah nilai integer yang dikembalikan oleh connect().

Apabila diperlukan, periksalah apakah suatu handle terkoneksi dengan:

```
gobject.GObject.handler_is_connected(handler_id)
```

yang akan mengembalikan True apabila callback dengan ID handler_id terkoneksi.

Mengaktifkan (emit) signal

Signal bisa diaktifkan dengan:

```
gobject.GObject.emit(detailed_signal, ...).
```

- detailed_signal: string berisikan nama signal
- ...: argumen tambahan

Jumlah dan tipe argumen tambahan harus sesuai dengan jumlah dan tipe yang disyaratkan callback.

signal_simple.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Signal')
        self.window.connect('destroy', self.destroy)

        self.window.show_all()

    def destroy(self, widget, data=None):
        gtk.main_quit()


app = Main()
gtk.main()
```

signal_hello.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Hello World')
        self.window.connect('destroy', self.destroy)

        self.hbox = gtk.HBox()

        self.entry = gtk.Entry()
        self.button = gtk.Button('_uppercase')
        self.button.connect('clicked', self.do_uppercase, self.entry)

        self.hbox.add(self.entry)
        self.hbox.add(self.button)

        self.window.add(self.hbox)

        self.window.show_all()
```

```
    def destroy(self, widget, data=None):
        gtk.main_quit()

    def do_uppercase(self, widget, data=None):
        text = data.get_text().upper()
        data.set_text(text)

app = Main()
gtk.main()
```

signal_multi.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Signal')
        self.window.connect('destroy', self.destroy)

        self.button = gtk.Button('button')
        self.button.connect('clicked', self.button_click)
        self.button.connect('clicked', self.button_click2)
        self.button.connect('enter', self.button_enter)
        self.button.connect('leave', self.button_leave)

        self.window.add(self.button)

        self.window.show_all()

    def destroy(self, widget, data=None):
        gtk.main_quit()

    def button_click(self, widget):
        print 'Button click'

    def button_click2(self, widget):
        print 'Button click (2)'

    def button_enter(self, widget):
        print 'Button mouse enter'

    def button_leave(self, widget):
        print 'Button mouse leave'

app = Main()
```

```
gtk.main()
```

signal_block.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk
import time

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Signal block')
        self.window.connect('destroy', gtk.main_quit)

        self.vbox = gtk.VBox(homogeneous=True)
        self.hbox = gtk.HBox(homogeneous=True)

        self.label = gtk.Label()

        self.handle = -1

        self.button1 = gtk.Button('Button 1')
          self.handle = self.button1.connect('clicked', self.button1_click,
self.label)
        self.button2 = gtk.Button('Button 1 block')
        self.button2.connect('clicked', self.button2_click)
        self.button3 = gtk.Button('Button 1 unblock')
        self.button3.connect('clicked', self.button3_click)

        self.hbox.add(self.button1)
        self.hbox.add(self.button2)
        self.hbox.add(self.button3)

        self.vbox.add(self.hbox)
        self.vbox.add(self.label)

        self.window.add(self.vbox)

        self.window.show_all()

    def button1_click(self, widget, data=None):
        data.set_text(time.asctime())

    def button2_click(self, widget, data=None):
        self.button1.handler_block(self.handle)
        print 'block, handle %d' %(self.handle)

    def button3_click(self, widget, data=None):
```

```
        self.button1.handler_unblock(self.handle)
        print 'unblock, handle %d' %(self.handle)


app = Main()
gtk.main()
```

signal_disconnect.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk
import time

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Signal disconnect')
        self.window.connect('destroy', gtk.main_quit)

        self.vbox = gtk.VBox(homogeneous=True)
        self.hbox = gtk.HBox(homogeneous=True)

        self.label = gtk.Label()

        self.handle = -1

        self.button1 = gtk.Button('Button 1')
        self.button2 = gtk.Button('Button 1 Connect')
        self.button2.connect('clicked', self.button2_click)
        self.button3 = gtk.Button('Button 1 Disconnect')
        self.button3.connect('clicked', self.button3_click)

        self.hbox.add(self.button1)
        self.hbox.add(self.button2)
        self.hbox.add(self.button3)

        self.vbox.add(self.hbox)
        self.vbox.add(self.label)

        self.window.add(self.vbox)

        self.window.show_all()

    def button1_click(self, widget, data=None):
        data.set_text(time.asctime())

    def button2_click(self, widget, data=None):
        if not self.button1.handler_is_connected(self.handle):
```

```
            self.handle = self.button1.connect('clicked',
self.button1_click, self.label)
            print 'connect, handle %d' %(self.handle)

    def button3_click(self, widget, data=None):
        if self.button1.handler_is_connected(self.handle):
            self.button1.disconnect(self.handle)
            print 'disconnect, handle %d' %(self.handle)


app = Main()
gtk.main()
```

signal_emit.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk
import time

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Signal emit')
        self.window.connect('destroy', gtk.main_quit)

        self.hbox = gtk.HBox(homogeneous=True)

        self.button1 = gtk.Button('Button 1')
        self.button1.connect('clicked', self.button1_click)
        self.button2 = gtk.Button('Click Button 1')
        self.button2.connect('clicked', self.button2_click)

        self.hbox.add(self.button1)
        self.hbox.add(self.button2)

        self.window.add(self.hbox)

        self.window.show_all()

    def button1_click(self, widget, data=None):
        print 'Button 1 click'

    def button2_click(self, widget, data=None):
        self.button1.emit('clicked')


app = Main()
gtk.main()
```

Catatan contoh:
- gtk.main_quit() dapat digunakan untuk keluar dari mainloop gtk.
- Di contoh signal_simple.py, window kini bisa diclose dengan klik pada tombol close.
- Di contoh signal_hello.py, begitu tombol uppercase diklik, maka teks pada entry akan di-uppercase.
- Di contoh signal_multi.py, beberapa signal pada satu widget akan ditangani. Salah satu signal akan ditangani oleh lebih dari satu callback.
- Contoh signal_block.py mendemonstrasikan block dan unblock signal-callback.
- Contoh signal_disconnect.py mendemonstrasikan diskoneksi signal-callback.
- Contoh signal_emit.py mendemonstrasikan pengaktifan (emit) signal tertentu.

Koneksi event-callback

Pengaturan koneksi sama dengan pengaturan pada signal. Sementara, definisi callback sedikit berbeda. Untuk hal-hal lainnya, rujuklah juga kepada pembahasan koneksi signal-callback sebelumnya.

Definisi callback
Callback secara umum dapat didefinisikan sebagai berikut:

```
def callback (widget, event, data)
```

Apabila argumen opsional tidak digunakan pada connect(), maka callback secara umum dapat didefinisikan sebagai berikut:

```
def callback (widget, event, data=None)
atau
def callback (widget, event)
```

Catatan:
- Callback harus menyesuaikan dengan jumlah argumen opsional yang diberikan.
- Apabila callback digunakan dalam class, maka argumen pertama pada definisi callback tetap adalah self.
- Argumen event akan dilewatkan sebagai GdkEvent. Bacalah referensi untuk gtk.gdk.Event.
- Return value callback, apabila True, diartikan sebagai event telah ditangani dan tidak dipropagasi lagi. Sementara, apabila False, event handling normal akan dilakukan.

Daftar event
Daftar event diantaranya:

- event
- button_press_event
- button_release_event
- scroll_event
- motion_notify_event
- delete_event
- destroy_event
- expose_event
- key_press_event
- key_release_event
- enter_notify_event
- leave_notify_event
- configure_event
- focus_in_event
- focus_out_event
- map_event
- unmap_event
- property_notify_event
- selection_clear_event
- selection_request_event
- selection_notify_event
- proximity_in_event
- proximity_out_event
- visibility_notify_event
- client_event
- no_expose_event
- window_state_event

Representasi tipe data Python dapat dibaca pada referensi untuk gtk.gdk.Event.

event_simple.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Event')
        self.window.connect('destroy', self.destroy)
        self.window.connect('delete_event', self.delete_event)

        self.window.show_all()

    def delete_event(self, widget, event, data=None):
        #return True #will NOT be destroyed
        return False #will be destroyed

    def destroy(self, widget):
```

```
        gtk.main_quit()

app = Main()
gtk.main()
```

event_3button.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Event')
        self.window.connect('destroy', self.destroy)

        self.button = gtk.Button('triple click me')
        self.button.connect('button_press_event', self.button_tripleclick)

        self.window.add(self.button)

        self.window.show_all()

    def button_tripleclick(self, widget, event):
        if event.type == gtk.gdk._3BUTTON_PRESS:
            print 'Great!'

    def destroy(self, widget):
        gtk.main_quit()

app = Main()
gtk.main()
```

event_keypress.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Event')
        self.window.set_size_request(300, 200)
        self.window.connect('destroy', self.destroy)
```

```
        self.label = gtk.Label()

        self.window.connect('key_press_event', self.window_keypress,
self.label)

        self.window.add(self.label)

        self.window.show_all()

    def window_keypress(self, widget, event, label):
        label.set_label(event.string)

    def destroy(self, widget):
        gtk.main_quit()

app = Main()
gtk.main()
```

event_keypress2.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title('Event')
        self.window.set_size_request(500, 200)
        self.window.connect('destroy', self.destroy)

        self.label = gtk.Label()

        self.window.connect('key_press_event', self.window_keypress,
self.label)

        self.window.add(self.label)

        self.window.show_all()

    def window_keypress(self, widget, event, label):
        state = event.state.value_names
        if state:
            text =  '+'.join(state) + ' ' + event.string
        else:
            text = event.string
        label.set_label(text)
```

```
    def destroy(self, widget):
        gtk.main_quit()

app = Main()
gtk.main()
```

Catatan contoh:

- Contoh event_simple.py mendemonstrasikan delete_event untuk window (window ditutup). Apabila callback mengembalikan True, maka event dianggap telah ditangani. Window tidak ditutup. Apabila mengembalikan False, maka window ditutup. Penanganan event seperti ini umum ditemukan seperti pada konfirmasi keluar dari program.
- Contoh event_3button.py mendemonstrasikan button_press_event. Tulisan 'Great!' akan dicetak ke stdout apabila tombol ditriple-click. Event button_press_event adalah event ketika tombol mouse ditekan. Kita perlu memeriksa atribut type milik event untuk mengetahui apakah tombol mouse ditriple-click (gtk.gdk._3BUTTON_PRESS).
- Contoh event_keypress.py mendemonstrasikan key_press_event. Apa yang dilakukan hanyalah mengatur label milik self.label menjadi event.string.
- Contoh event_keypress2.py melengkapi event_keypress2.py sehingga dapat mendeteksi penekanan CTRL, SHIFT, ALT(MOD1) dan lainnya.

6. Beragam widget

GTK+ menyediakan widget set lengkap untuk membangun aplikasi GUI. Widget-widget tersebut diturunkan dari gtk.Widget, dan memiliki diantaranya beberapa method berikut:

- set_sensitive(sensitive). Apabila True, maka widget dapat digunakan (enabled). Sebaliknya, widget tidak dapat digunakan (disabled).
- show(). Menampilkan widget.
- show_all(). Menampilkan widget dan setiap child widget.
- hide(). Menyembunyikan widget.
- hide_all(). Menyembunyikan widget dan setiap child widget.
- destroy(). Menghapus widget. Apabila widget berada dalam container, maka widget akan dihapus dari container. Apabila widget adalah toplevel, maka semua child widget juga akan dihapus. Oleh karena itu, destroy() umumnya hanya perlu digunakan pada toplevel.
- reparent(new_parent). Memindahkan widget dari satu container ke container lainnya.
- grab_focus(). Menjadikan widget memiliki fokus keyboard.
- grab_default(). Menjadikan suatu widget menjadi default, yang dapat diaktifkan ketika user menekan Enter.
- set_size_request(width, height). Mengatur ukuran minimum widget.

Untuk informasi selengkapnya, lihatlah class reference untuk widget yang digunakan, ditambah dengan class reference untuk class-class orangtuanya.

gtk.Label

Widget untuk menampilkan teks terbatas, yang hanya dapat dibaca.

```
gtk.Label(str=None)
```

Contoh:

```
w = gtk.Label('Hello')
```

gtk.Button

Widget tombol, yang dapat pula berisikan gambar tertentu (stock atau eksternal). Gunakan underscore sebelum karakter accelerator key dalam label button.

```
gtk.Button(label=None, stock=None, use_underline=True)
```

Contoh:

```
w = gtk.Button('_Save', gtk.STOCK_SAVE)
```

gtk.ToggleButton

Merupakan button dengan fitur dapat di-toggle. Gunakan underscore sebelum karakter accelerator key dalam label button.

```
gtk.ToggleButton(label=None, use_underline=True)
```

Contoh:

```
w = gtk.ToggleButton('Option _1')
```

gtk.CheckButton

Merupakan toggle button dengan checkbox dan label. Gunakan underscore sebelum karakter accelerator key dalam label button.

```
gtk.CheckButton(label=None, use_underline=True)
```

Contoh:

```
w = gtk.CheckButton('Option _1')
```

gtk.RadioButton

Merupakan checkbutton namun hanya satu yang dapat aktif dalam satu group. Dapat digunakan sebagai pilihan berganda.

```
gtk.RadioButton(group=None, label=None, use_underline=True)
```

Catatan:

- Argumen group akan membentuk group radio button.
- Group baru: group = None.
- Anggota group: group = radio button yang mengawali group.
- Group dapat pula diset dengan set_group(group).

Contoh:

```
w1 = gtk.RadioButton(None, 'Option _1')
w2 = gtk.RadioButton(w1, 'Option _2')
w3 = gtk.RadioButton(w1, 'Option _3')
```

gtk.SpinButton

SpinButton dapat digunakan untuk mendapatkan nilai integer atau floating-point dari user, dimana user dapat klik pada tombol panah atas dan bawah untuk menambahkan atau mengurangi nilai.

```
gtk.SpinButton(adjustment=None, climb_rate=0.0, digits=0)
```

Catatan:
Untuk memudahkan pengaturan nilai, batas bawah, batas atas, increment, page increment dan page size, buatlah gtk.Adjustment terlebih dahulu.

```
gtk.Adjustment(value=0, lower=0, upper=0, step_incr=0, page_incr=0, page_size=0)
```

Contoh:

```
adj = gtk.Adjustment(0, 0, 10, 1, 4)
w = gtk.SpinButton(adj)
```

gtk.ColorButton

Merupakan button yang ketika diklik, akan membuka dialog pemilihan warna. Warna yang dipilih kemudian akan tampil pada button.

```
gtk.ColorButton(color=gtk.gdk.Color(0,0,0))
```

Catatan:
- Untuk mendapatkan warna aktif/yang dipilih, gunakanlah get_color(), yang akan mengembalikan objek gtk.gdk.Color.

```
gtk.gdk.Color(red=0, green=0, blue=0, pixel=0)
```

- Widget untuk pemilihan warna (yang tampil pada dialog) adalah gtk.ColorSelection.
- Dialog untuk pemilihan warna adalah gtk.ColorSelectionDialog.

Contoh:

```
w = gtk.ColorButton()
```

gtk.FontButton

Merupakan button yang ketika diklik, akan membuka dialog pemilihan font. Font yang dipilih kemudian akan tampil pada button.

```
gtk.FontButton(fontname=None)
```

Catatan:
- Gunakan get_font_name() untuk mendapatkan nama font.
- Widget untuk pemilihan font (yang tampil pada dialog) adalah gtk.FontSelection.
- Dialog untuk pemilihan font adalah gtk.FontSelectionDialog.

Contoh:
```
w = gtk.FontButton()
```

gtk.FileChooserButton

Merupakan button yang ketika diklik, akan membuka dialog pemilihan file. Nama file yang dipilih kemudian akan tampil pada button.

```
gtk.FileChooserButton(title, backend=None)
gtk.FileChooserButton(dialog)
```

Catatan:
- Untuk mendapatkan pilihan user dan pengaturan lainnya, lihatlah method-method pada referensi class gtk.FileChooser.
- Dialog untuk pemilihan file adalah gtk.FileChooserDialog.
- Lihatlah juga gtk.FileChooserWidget dan gtk.FileSelection.

Contoh:
```
w = gtk.FileChooserButton('Pilih file')
w.set_current_folder('/tmp')
```

gtk.Entry

Entry adalah widget yang menyediakan input berupa satu baris teks.

```
gtk.Entry(max=0)
```

Catatan:
- Untuk input berupa password, dimana setiap karakter yang diketik user di-mask dengan karakter tertentu, seperti * atau x, lakukan langkah-langkah:
  - set_visibility(False)
  - set_invisible_char(ch). Contoh: set_invisible_char('x'). Nilai default ch adalah *.
- Untuk mendapatkan teks yang diinput, gunakanlah get_text().
- Untuk dapat bekerja dengan text completion (dari sejumlah predefined-text), sehingga user dapat menekan beberapa karakter pertama dan sebuah popup akan ditampilkan sesuai input user, lakukanlah langkah-langkah berikut:
  - Buat sebuah gtk.ListStore dengan satu kolom bertipe str.
  - Tambahkan setiap predefined-text ke objek gtk.ListStore yang dibuat, dalam bentuk sequence (contoh: list).
  - Buat sebuah gtk.EntryCompletion dan set_model ke objek gtk.ListStore yang dibuat sebelumnya.
  - Tentukan kolom pada model untuk mendapatkan teks. Dalam contoh completionn sederhana ini, teks didapat dari kolom 0. Gunakanlah gtk.EntryCompletion.set_text_column(0).
  - Set completion Entry dengan gtk.Entry.set_completion(completion).
  - Lebih lanjut, lihatlah pada contoh 4.

Contoh 1:
```
w = gtk.Entry(3) #maksimal 3
```

Contoh 2:
```
w = gtk.Entry()
w.set_text('input your password here')
```

Contoh 3:
```
w = gtk.Entry()
w.set_visibility(False)
w.set_invisible_char('x')
```

Contoh 4:
```
w = gtk.Entry()

liststore = gtk.ListStore(str)
texts = ['abc', 'bcd', 'bce', 'cde', 'cdf', 'def', 'deg']
for t in texts:
     liststore.append([t])

completion = gtk.EntryCompletion()
completion.set_model(liststore)
completion.set_text_column(0)

w.set_completion(completion)
```

gtk.ComboBox

ComboBox menyediakan pilihan untuk user, dalam bentuk drop down list. ComboBox di PyGTK dapat digunakan setidaknya dengan dua cara:
- cara sederhana
- menggunakan gtk.TreeModel

Cara sederhana
Dalam cara sederhana, combobox dibuat dengan:

```
gtk.combo_box_new_text()
```

Fungsi tersebut akan mengembalikan objek gtk.ComboBox untuk item berupa teks. Selanjutnya, untuk bekerja dengan combobox:
- append (menambahkan di akhir) teks: gtk.ComboBox.append_text(text)
- insert teks: gtk.ComboBox.insert_text(position, text)
- prepend (menambahkan di awal): gtk.ComboBox.prepend_text(text)
- menghapus teks: gtk.ComboBox.remove_text(position)
- mendapatkan item terpilih: gtk.ComboBox.get_active_text()

Contoh:

```
w = gtk.combo_box_new_text()
w.append_text('Pilihan 2')
```

```
w.prepend_text('Pilihan 1')
w.append_text('Pilihan 3')
w.insert_text(2, 'Pilihan 2 lain')
w.remove_text(2)
```

Menggunakan gtk.TreeModel
Untuk cara yang lebih kompleks, namun fleksibel dan powerful (seperti dapat menambahkan image), combobox dapat dibuat dengan:

```
gtk.ComboBox(model=None)
```

Argumen model sendiri adalah gtk.TreeModel. Berikut adalah contoh untuk menghadirkan combobox dengan beberapa item string (model adalah gtk.ListStore).

```
liststore = gtk.ListStore(str)
texts = ['Pilihan 1', 'Pilihan 2', 'Pilihan 3']
for t in texts:
     liststore.append([t])

w = gtk.ComboBox(liststore)
cell = gtk.CellRendererText()
w.pack_start(cell, True)
w.add_attribute(cell, 'text', 0)
```

Catatan:
Untuk menyediakan fasilitas combobox bagi user, namun user dapat mengisikan langsung nilai selain yang terdapat pada pilihan, gunakanlah gtk.ComboBoxEntry. Prinsip penggunaannya mirip dengan gtk.ComboBox dan dapat dibuat secara mudah dengan gtk.combo_box_entry_new_text().

Lihatlah referensi class untuk gtk.ComboBoxEntry.

gtk.Frame

Frame merupakan sebuah container, dengan bingkai dan label opsional.

```
gtk.Frame(label=None)
```

Contoh:
```
w = gtk.Frame('Configuration')
```

gtk.Image

Image merupakan widget yang dapat menampilkan gambar. Gambar bisa didapat diantaranya dari pixmap, gtk.gdk.image, file, gtk.gdk.Pixbuf, GTK stock, icon set, atau gtk.gdk.PixbufAnimation.

```
gtk.Image()
```

Catatan:
- Untuk mengambil gambar dari file, gunakan gtk.Image.set_from_file(filename)
- Untuk menghapus gambar, gunakan gtk.Image.clear()

Contoh:
```
w = gtk.Image()
w.set_from_file('./smile.png')
```

gtk.Tooltips

Tooltips dapat digunakan untuk menambahkan tips (hint) ke suatu widget.

```
gtk.Tooltips()
```

Catatan:
Gunakan gtk.Tooltips.set_tip(widget, tip_text, tip_private=None) untuk mengatur tip untuk suatu widget dengan tip adalah tip_text.

Contoh:
```
w = gtk.Button('Klik')
t = gtk.Tooltips()
t.set_tip(w, 'Klik untuk keluar dari aplikasi')
```

gtk.Scale

Scale dapat digunakan untuk memilih nilai tertentu dalam batasan, dengan mempergunakan slider. Terdapat dua jenis scale:
- gtk.HScale: gtk.HScale(adjustment=None)
- gtk.VScale: gtk.VScale(adjustment=None)

Keduanya diturunkan dari gtk.Scale, dan gtk.Scale sendiri diturunkan dari gtk.Range. Beberapa method gtk.Range:
- mendapatkan value: gtk.Range.get_value()
- mengatur range: gtk.Range.set_range(min, max)
- memberikan nilai tertentu: gtk.Range.set_value(value)
- mengatur adjustment: gtk.Range.set_adjustment(adjustment)

Catatan:
Untuk memudahkan pengaturan nilai, batas bawah, batas atas, increment, page increment dan page size, buatlah gtk.Adjustment terlebih dahulu.

```
gtk.Adjustment(value=0, lower=0, upper=0, step_incr=0, page_incr=0, page_size=0)
```

Contoh:
```
adj = gtk.Adjustment(1, 1, 10, 1)
```

```
w = gtk.VScale(adj)
```

gtk.ProgressBar

Widget Progressbar umum digunakan untuk menampilkan progres secara visual.

```
gtk.ProgressBar()
```

Catatan:
- Mengatur teks pada progress: gtk.ProgressBar.set_text(text).
- Mengatur nilai pada progress: gtk.ProgressBar.set_fraction(fraction). Nilai fraction antara 0.0 dan 1.0.
- Mendapatkan nilai progress: gtk.ProgressBar.get_fraction(). Nilai fraction antara 0.0 dan 1.0.
- Orientasi progressbar dapat diset dengan: gtk.ProgressBar.set_orientation(orientation), dimana orientation adalah salah satu dari: gtk.PROGRESS_LEFT_TO_RIGHT, gtk.PROGRESS_RIGHT_TO_LEFT, gtk.PROGRESS_BOTTOM_TO_TOP atau gtk.PROGRESS_TOP_TO_BOTTOM.

Contoh:
```
w = gtk.ProgressBar()
val = 0.5
w.set_fraction(val)
w.set_text('Percentage: %d%%' %(val*100))
w.set_orientation(gtk.PROGRESS_BOTTOM_TO_TOP)
```

gtk.ScrolledWindow

Widget ScrolledWindow akan menambahkan scrollbar ke child widget. Sangat berguna untuk child widget yang mungkin membutuhkan fungsionalitas scroll seperti gtk.TextView.

```
gtk.ScrolledWindow(hadjustment=None, vadjustment=None)
```

Catatan:
Untuk mengatur bagaimana scrollbar ditampilkan, gunakan:

```
gtk.ScrolledWindow.set_policy(hscrollbar_policy, vscrollbar_policy)
```

hscrollbar_policy dan vscrollbar_policy dapat berupa: gtk.POLICY_ALWAYS (scrollbar selalu tampil), gtk.POLICY_AUTOMATIC (scrollbar tampil ketika diperlukan), gtk.POLICY_NEVER (scrollbar tidak pernah ditampilkan).

Contoh:
Lihat pembahasan gtk.TextView.

gtk.TextView

TextView dapat digunakan untuk menampilkan teks, baik sederhana (hanya teks berukuran kecil) ataupun kompleks (teks dengan atribut tertentu ataupun widget, berukuran kecil ataupun besar).

```
gtk.TextView(buffer=None)
```

Catatan:
- Widget TextView sangatlah powerful. Training hanya akan membahas fungsionalitas dasar, yaitu menampilkan teks.
- Teks yang ditampilkan dan atributnya disimpan dalam gtk.TextBuffer.

```
gtk.TextBuffer(table=None)
```

Contoh:
```
text = ''
for i in range(1, 100):
     text += 'line %d\n' %(i)

buf  = gtk.TextBuffer()
buf.set_text(text)

w = gtk.TextView()
w.set_buffer(buf)

scrolled = gtk.ScrolledWindow()
scrolled.set_policy(gtk.POLICY_AUTOMATIC, gtk.POLICY_AUTOMATIC)
scrolled.add(w)
```

gtk.Statusbar

Statusbar dapat digunakan untuk menampilkan status aplikasi, dan umum diposisikan di bagian bawah window. Widget ini memiliki sebuah resize grip yang dapat digunakan untuk meresize window yang mengandung statusbar.

Statusbar di PyGTK bekerja dengan konsep stack, dimana teks status bisa dipush ke stack atau dipop dari stack. Teks status yang paling atas yang akan ditampilkan.

Catatan:
- Ketika mem-push sebuah teks status dengan gtk.Statusbar.push(context_id, text), context_id bisa didapatkan dengan gtk.Statusbar.get_context_id(context_description).
- Untuk mem-pop teks status dari stack, gunakan gtk.Statusbar.pop(context_id).
- Untuk menentukan apakah resize grip ditampilkan, gunakan set_has_resize_grip(setting). Argumen setting bernilai True atau False.
- gtk.Statusbar diturunkan dari gtk.HBox dan oleh karenanya, kita dapat menambahkan widget lain (seperti gtk.ProgressBar) ke dalamnya.

Contoh 1:
```
w = gtk.Statusbar()
id = w.get_context_id('info')
w.push(id, 'System ready.')
```

Contoh 2:
```
w = gtk.Statusbar()
id = w.get_context_id('info')
w.push(id, 'System ready.')
w.pop(id) #no more status text
```

Contoh 3:
```
w = gtk.Statusbar()
id = w.get_context_id('info')
w.push(id, 'System ready.')
w.set_has_resize_grip(False)
```

Contoh 4:
```
w = gtk.Statusbar()
w.push(w.get_context_id('info'), 'System ready.')
p = gtk.ProgressBar()
p.set_fraction(0.5)
w.pack_end(p)
```

gtk.Expander

Expander merupakan widget container yang dapat menyembunyikan child widgetnya menggunakan sebuah tombol panah kecil.

```
gtk.Expander(label=None)
```

Catatan:
Agar accelerator key dapat digunakan dengan karakter underline, gunakan gtk.Expander.set_use_underline(use_underline), dengan use_underline adalah True.

Contoh:
```
lbl = gtk.Label('Detail text')
w = gtk.Expander('_Detail')
w.set_use_underline(True)
w.add(lbl)
```

gtk.ButtonBox

ButtonBox dapat digunakan untuk menempatkan sekelompok tombol secara konsisten. Developer dapat menggunakan:
- gtk.HButtonBox: gtk.HButtonBox()
- gtk.VButtonBox: gtk.VButtonBox()

Catatan:

- Untuk mengatur layout tombol, gunakanlah gtk.ButtonBox.set_layout(layout_style) dengan layout_style adalah salah satu dari gtk.BUTTONBOX_SPREAD, gtk.BUTTONBOX_EDGE, gtk.BUTTONBOX_START atau gtk.BUTTONBOX_END.
- Untuk menambahkan tombol, gunakan: gtk.Container.add(widget), gtk.Box.pack_start(child, expand=True, fill=True, padding=0) atau gtk.Box.pack_end(child, expand=True, fill=True, padding=0).
- Untuk mengatur spacing, gunakan: gtk.Box.set_spacing(spacing).

Contoh:
```
b1 = gtk.Button('Button 1')
b2 = gtk.Button('Button 2')
b3 = gtk.Button('Button 3')
b4 = gtk.Button('Button 4')
w = gtk.HButtonBox()
w.set_layout(gtk.BUTTONBOX_SPREAD)
w.pack_start(b1)
w.pack_start(b2)
w.pack_start(b3)
w.pack_start(b4)
```

gtk.Paned

Paned dapat digunakan untuk menempatkan dua widget, dimana diantara keduanya dipisahkan oleh resizer. Developer dapat menggunakan:
- gtk.HPaned: gtk.HPaned()
- gtk.VPaned: gtk.VPaned()

Catatan:
- Untuk menambahkan widget di sisi kiri atau atas, gunakan gtk.Paned.add1(child) atau gtk.Paned.pack1(child, resize=False, shrink=True).
- Untuk menambahkan widget di sisi kanan atau bawah, gunakan gtk.Paned.add2(child) atau gtk.Paned.pack2(child, resize=True, shrink=True).

Contoh 1:
```
b1 = gtk.Button('Button 1')
b2 = gtk.Button('Button 2')
w = gtk.HPaned()
w.pack1(b1)
w.pack2(b2)
```

Contoh 2:
```
b1 = gtk.Button('Button 1')
b2 = gtk.Button('Button 2')
box = gtk.HButtonBox()
box.set_spacing(10)
box.set_layout(gtk.BUTTONBOX_END)
box.add(b1)
```

```
box.add(b2)


t = gtk.TextView()

w = gtk.VPaned()
w.pack1(t)
w.pack2(box)
```

gtk.Notebook

Notebook menyediakan container yang dilengkapi dengan tab. Sangat umum ditemukan dalam dialog preferensi aplikasi.

```
gtk.Notebook()
```

Catatan:
- Notebook adalah widget yang kompleks. Rujuklah ke referensi class gtk.Notebook untuk informasi selengkapnya.
- Untuk menambahkan (append) tab baru, gunakan: gtk.Notebook.append_page(child, tab_label=None). Argumen child adalah widget yang ingin ditambahkan. Sementara, argumen tab_label merupakan widget yang digunakan sebagai label suatu tab. Dengan demikian, label tab tidak harus selalu teks (lihat contoh 2). Method akan mengembalikan indeks halaman.
- Untuk sekedar mengatur title tab, gunakan: gtk.Notebook.set_tab_label_text(child, tab_text). Argumen child adalah child widget yang ditambahkan pada notebook. Sementara, tab_text adalah title untuk widget tersebut.
- Untuk mengatur posisi tab, gunakan: gtk.Notebook.set_tab_pos(pos), dimana pos adalah salah satu dari: gtk.POS_LEFT, gtk.POS_RIGHT, gtk.POS_TOP atau gtk.POS_BOTTOM.
- Untuk mengatur agar area tab notebook berisikan tombol untuk scroll apabila jumlah tab terlalu banyak, gunakan: gtk.Notebook.set_scrollable((scrollable), dimana scrollable adalah True atau False.
- Untuk menghadirkan popup menu berisikan tab-tab yang ada, gunakan: gtk.Notebook.popup_enable().

Contoh 1:

```
t = gtk.TextView()
f = gtk.FontSelection()

w = gtk.Notebook()

p1 = w.append_page(t)
p2 = w.append_page(f)
w.set_tab_label_text(t, 'Text View')
w.set_tab_label_text(f, 'Select font')
```

```
w.set_tab_pos(gtk.POS_BOTTOM)
w.set_scrollable(True)
w.popup_enable()
```

Contoh 2:

```
t = gtk.TextView()
f = gtk.FontSelection()

t_title = gtk.Image()
t_title.set_from_file('./smile.png')

w = gtk.Notebook()

p1 = w.append_page(t, t_title)
p2 = w.append_page(f)
w.set_tab_label_text(f, 'Select font')
```

gtk.Calendar

Calendar merupakan widget yang menampilkan kalender, sehingga pemilihan tanggal dapat dilakukan dengan mudah.

```
gtk.Calendar()
```

Catatan:

- Secara default, tanggal hari ini akan ditampilkan.
- Untuk mengatur bulan dan tahun, gunakan: gtk.Calendar.select_month(month, year). Argumen month berada dalam range 0 (Januari) sampai 11 (Desember).
- Untuk mengatur tanggal, gunakan: gtk.Calendar.select_day(day). Argumen day adalah dari 1 sampai 31. Apabila diisi dengan 0, maka tanggal tidak lagi terpilih.
- Untuk menandai tanggal tertentu secara visual (misal: diformat bold), gunakan: gtk.Calendar.mark_day(day). Untuk menghilangkan penanda, gunakan gtk.Calendar.unmark_day(day).
- Untuk membersihkan semua penanda tanggal, gunakan: gtk.Calendar.clear_marks().
- Untuk mendapatkan tanggal terpilih, gunakan: gtk.Calendar.get_date(), yang akan mengembalikan tuple berisikan year, month dan day.
- Untuk mengatur opsi penampilan kalender, gunakan gtk.Calendar.set_display_options(flags), dimana flags merupakan kombinasi dari: gtk.CALENDAR_SHOW_HEADING (bulan dan tanggal ditampilkan), gtk.CALENDAR_SHOW_DAY_NAMES (nama hari dalam 3 huruf ditampilkan), gtk.CALENDAR_NO_MONTH_CHANGE (bulan tidak dapat diganti), gtk.CALENDAR_SHOW_WEEK_NUMBERS (menampilkan nomor minggu dalam tahun tersebut), gtk.CALENDAR_WEEK_START_MONDAY (mengawali minggu pada senin, bukan default minggu).

Contoh:

```
w = gtk.Calendar()
w.select_month(0, 2020)
w.select_day(1)
w.mark_day(27)
w.mark_day(17)
w.set_display_options(  gtk.CALENDAR_WEEK_START_MONDAY |
                        gtk.CALENDAR_NO_MONTH_CHANGE |
                        gtk.CALENDAR_SHOW_WEEK_NUMBERS |
                        gtk.CALENDAR_SHOW_DAY_NAMES |
                        gtk.CALENDAR_SHOW_HEADING)
```

gtk.Menu, gtk.MenuItem, gtk.MenuBar

Untuk membuat menubar lengkap dengan menu dan menu item, dua cara berikut bisa dilakukan:
- manual: menu disusun item demi item secara manual.
- menggunakan gtk.UIManager: pembuatan menu dengan cara mudah.

Materi training hanya akan menggunakan cara manual.

Untuk membuat menu secara manual, lakukanlah langkah-langkah berikut:
- Buatlah sebuah objek gtk.MenuBar, yang akan berfungsi sebagai menubar. Menubar ini dapat ditambahkan ke container seperti objek lainnya.
- Untuk setiap menu (seperti File, Edit, Help, dan lainnya):
  - Buat sebuah menu (gtk.Menu), yang merepresentasikan menu yang dibuat.
  - Buat berbagai menuitem (gtk.MenuItem), yang mewakili setiap item menu (seperti Open, Save, About, dan lainnya). Menu item tidak harus berupa teks (gtk.MenuItem), tapi bisa berupa separator (gtk.SeparatorMenuItem), tearoff (gtk.TearoffMenuItem), check button (gtk.CheckMenuItem), radio button (gtk.RadioMenuItem), dan image (gtk.ImageMenuItem).
  - Tambahkan setiap menuitem yang dibuat ke menu (gtk.Menu, yang dibuat sebelumnya) dengan gtk.MenuShell.append(child). Argumen child adalah menuitem yang dibuat.
  - Buat satu lagi menuitem (gtk.MenuItem), yang akan menjadi menu yang terlihat pada menubar, yang merupakan menu seperti File, Edit, Help dan lain sebagainya. Setelah itu, set submenu milik menuitem tersebut dengan menu (gtk.Menu) yang dibuat sebelumnya. Untuk mengatur submenu, gunakan: gtk.MenuItem.set_submenu(submenu). Untuk menuitem yang ingin ditempatkan rata kanan (seperti Help pada umumnya), gunakan: gtk.MenuItem.set_right_justified(True).
  - Tambahkan menuitem yang dibuat sebelumnya (mewakili File, Edit, Help dan lain sebagainya) ke menubar dengan: gtk.MenuShell.append(child).
- Catatan: langkah-langkah manual ini pada awalnya terlihat rumit, namun fleksibel dan powerful.

gtk.MenuBar:
```
gtk.MenuBar()
```

gtk.Menu:
```
gtk.Menu()
```

gtk.MenuItem dan turunannya:
```
gtk.MenuItem(label=None, use_underline=True)
gtk.CheckMenuItem(label=None, use_underline=True)
gtk.ImageMenuItem(stock_id=None, accel_group=None)
gtk.RadioMenuItem(group=None, label=None, use_underline=True)
gtk.SeparatorMenuItem()
gtk.TearoffMenuItem()
```

Contoh:
```
menubar = gtk.MenuBar()

#file
menu_file = gtk.Menu()
item_new = gtk.MenuItem('_New')
item_open = gtk.MenuItem('_Open')
item_save = gtk.MenuItem('_Save')
item_quit = gtk.MenuItem('_Quit')
sep1 = gtk.SeparatorMenuItem()
menu_file.append(item_new)
menu_file.append(item_open)
menu_file.append(item_save)
menu_file.append(sep1)
menu_file.append(item_quit)

item_file = gtk.MenuItem('_File')
item_file.set_submenu(menu_file)



#extra, check, radio, tearoff
menu_extra = gtk.Menu()
item_tearoff = gtk.TearoffMenuItem()
item_check1 = gtk.CheckMenuItem('_Active')
item_radio1 = gtk.RadioMenuItem(None, 'Pilihan _1')
item_radio2 = gtk.RadioMenuItem(item_radio1, 'Pilihan _2')
item_radio3 = gtk.RadioMenuItem(item_radio1, 'Pilihan _3')
sep2 = gtk.SeparatorMenuItem()
menu_extra.append(item_tearoff)
menu_extra.append(item_check1)
menu_extra.append(sep2)
menu_extra.append(item_radio1)
menu_extra.append(item_radio2)
menu_extra.append(item_radio3)


item_extra = gtk.MenuItem('_Extra')
item_extra.set_submenu(menu_extra)
```

```
#help, right justify, image
menu_help = gtk.Menu()
item_content = gtk.MenuItem('_Content')
item_about = gtk.ImageMenuItem(gtk.STOCK_ABOUT)
menu_help.append(item_content)
menu_help.append(item_about)

item_help = gtk.ImageMenuItem(gtk.STOCK_HELP)
item_help.set_submenu(menu_help)
item_help.set_right_justified(True)



menubar.append(item_file)
menubar.append(item_extra)
menubar.append(item_help)
```

gtk.Toolbar, gtk.Toolitem

Untuk membuat toolbar lengkap dengan berbagai tombol dan item lain, dua cara berikut bisa dilakukan:
- manual: tombol dan item disusun secara manual.
- menggunakan gtk.UIManager: pembuatan toolbar dengan cara mudah.

Materi training hanya akan menggunakan cara manual.

Untuk membuat toolbar secara manual, lakukanlah langkah-langkah berikut:
- Buatlah terlebih dahulu semua item yang ingin ditempatkan pada toolbar. Item dapat berupa tombol (gtk.ToolButton), menu (gtk.MenuToolButton), radio button (gtk.RadioToolButton), separator (gtk.SeparatorToolItem) ataupun toggle button (gtk.ToggleToolButton).
- Buat objek gtk.Toolbar dan tambahkan item dengan gtk.Toolbar.insert(item, pos). Argumen item adalah item yang ingin ditambahkan dan pos adalah posisi. Posisi 0 adalah posisi paling awal. Posisi dapat pula diisikan sebagai bilangan negatif dan akan ditambahkan pada akhir toolbar.

gtk.Toolbar:

```
gtk.Toolbar()
```

gtk.ToolItem dan turunannya:

```
gtk.ToolButton(icon_widget=None, label=None)
gtk.MenuToolButton(stock_id)
gtk.RadioToolButton(group=None, stock_id=None)
gtk.SeparatorToolItem()
gtk.ToggleToolButton(stock_id=None)
```

Catatan:
Untuk item berupa gtk.MenuToolButton, pastikan semua menuitem ditampilkan dengan gtk.Widget.show().

Contoh:
```
btn_new = gtk.ToolButton(gtk.STOCK_NEW)
btn_fs = gtk.ToggleToolButton(gtk.STOCK_FULLSCREEN)
sep1 = gtk.SeparatorToolItem()
btn_quit = gtk.ToolButton(gtk.STOCK_QUIT)

menu_extra = gtk.Menu()
item_tearoff = gtk.TearoffMenuItem()
item_check1 = gtk.CheckMenuItem('_Active')
item_radio1 = gtk.RadioMenuItem(None, 'Pilihan _1')
item_radio2 = gtk.RadioMenuItem(item_radio1, 'Pilihan _2')
item_radio3 = gtk.RadioMenuItem(item_radio1, 'Pilihan _3')
sep = gtk.SeparatorMenuItem()
item_tearoff.show()
item_check1.show()
item_radio1.show()
item_radio2.show()
item_radio3.show()
sep.show()
menu_extra.append(item_tearoff)
menu_extra.append(item_check1)
menu_extra.append(sep)
menu_extra.append(item_radio1)
menu_extra.append(item_radio2)
menu_extra.append(item_radio3)

btn_menu = gtk.MenuToolButton(gtk.STOCK_OPEN)
btn_menu.set_menu(menu_extra)

w = gtk.Toolbar()
w.insert(btn_new, 0)
w.insert(btn_fs, 1)
w.insert(sep1, 2)
w.insert(btn_quit, 3)
w.insert(btn_menu, 4)
```

7. Dialog

Dialog umum digunakan diantaranya untuk meminta input, menampilkan pesan dan aksi lainnya. Materi training akan membahas gtk.Dialog untuk dialog umum (contoh: menampilkan pesan, meminta input) dan gtk.MessageDialog (menampilkan pesan dengan mudah).

gtk.Dialog

Widget gtk.Dialog terbagi dalam dua bagian secara vertikal, dipisahkan gtk.HSeparator:
- bagian atas: gtk.VBox, tempat developer mem-pack berbagai widget tambahan yang menyusun dialog.
- bagian bawah: action_area, sebuah gtk.HBox, tempat developer menempatkan berbagai tombol.

Pembuatan dialog
Untuk membuat gtk.Dialog, gunakan:

```
gtk.Dialog(title=None, parent=None, flags=0, buttons=None)
```

- title: title dialog
- parent: parent dialog
- flags: dapat merupakan kombinasi dari:
  - gtk.DIALOG_MODAL: mengatur dialog menjadi modal.
  - gtk.DIALOG_DESTROY_WITH_PARENT: didestroy ketika parentnya didestroy.
  - gtk.DIALOG_NO_SEPARATOR: separator tidak digunakan.
- buttons: tuple yang mengandung pasangan tombol/response ID (contoh: ('ok',1,'cancel',2,...)). Tombol dapat berupa teks atau GTK stock. Sementara, response ID adalah integer yang dapat berupa GTK response atau yang diset oleh developer. Daftar GTK response:
  - gtk.RESPONSE_NONE
  - gtk.RESPONSE_REJECT
  - gtk.RESPONSE_ACCEPT
  - gtk.RESPONSE_DELETE_EVENT
  - gtk.RESPONSE_OK
  - gtk.RESPONSE_CANCEL
  - gtk.RESPONSE_CLOSE
  - gtk.RESPONSE_YES
  - gtk.RESPONSE_NO
  - gtk.RESPONSE_APPLY
  - gtk.RESPONSE_HELP

Catatan:
- Langkah-langkah dasar bekerja dengan dialog:
  - Buat dialog, yang akan mengembalikan objek gtk.Dialog. Set button langsung pada saat pembuatan atau tambahkan setelah ini.
  - Pack semua komponen GUI yang menyusun dialog ke vbox dialog.
  - Tampilkan dialog dengan gtk.Dialog.run(). Method ini akan mengembalikan response ID. Response ID bisa didapat dari response ID

  tombol yang diset dan/atau GTK response (lihat bagian pembuatan dialog sebelumnya).
  - Cek response ID dan lakukan proses yang diinginkan (dapatkan teks hasil input, dan lain sebagainya).
  - Hapus dialog dengan gtk.Widget.destroy().
- Untuk menambahkan button, selain pada saat pembuatan dialog juga dapat menggunakan:
  - gtk.Dialog.add_button(button_text, response_id)
  - gtk.Dialog.add_buttons(...), dengan argumen adalah pasangan tombol/response ID.
- Contoh berbagai dialog bisa dilihat pada dialogs.py.

dialogs.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_size_request(400,100)
        self.window.connect('destroy', gtk.main_quit)

        b1 = gtk.Button('Say hello')
        b1.connect('clicked', self.do_hello, self.window)
        b2 = gtk.Button('Get response')
        b2.connect('clicked', self.do_response, self.window)
        b3 = gtk.Button('Get input')
        b3.connect('clicked', self.do_input, self.window)

        self.hbox = gtk.HBox()
        self.hbox.pack_start(b1)
        self.hbox.pack_start(b2)
        self.hbox.pack_start(b3)

        self.window.add(self.hbox)
        self.window.show_all()

    def do_hello(self, widget, parent):
        buttons = (gtk.STOCK_OK, gtk.RESPONSE_OK)

        label = gtk.Label('Hello!')
        label.show()

        d = gtk.Dialog('Informasi', parent, gtk.DIALOG_MODAL, buttons)
        d.vbox.pack_start(label)

        d.run()
        d.destroy()
```

```
    def do_response(self, widget, parent):
        buttons = (gtk.STOCK_OK, gtk.RESPONSE_OK, 'No comment', 1000)

        label = gtk.Label('Hello!')
        label.show()

        d = gtk.Dialog('Konfirmasi', parent, gtk.DIALOG_MODAL, buttons)
        d.vbox.pack_start(label)

        result = d.run()
        d.destroy()

        if result == gtk.RESPONSE_OK:
            print 'You said: OK'
        elif result == 1000:
            print 'You said: No comment'


    def do_input(self, widget, parent):
        buttons = (gtk.STOCK_OK, gtk.RESPONSE_OK,
            gtk.STOCK_CANCEL, gtk.RESPONSE_CANCEL)

        label = gtk.Label('Your name')
        entry = gtk.Entry()

        hbox = gtk.HBox()
        hbox.pack_start(label)
        hbox.pack_start(entry)
        hbox.show_all()

        d = gtk.Dialog('Input', parent, gtk.DIALOG_MODAL, buttons)
        d.vbox.pack_start(hbox)

        result = d.run()

        if result == gtk.RESPONSE_OK:
            print 'Hi, %s' %(entry.get_text())

        d.destroy()

app = Main()
gtk.main()
```

gtk.MessageDialog

Untuk menampilkan pesan dengan mudah, lengkap dengan jenis dialog (informasi/info, peringatan/warning, pertanyaan/question, atau kesalahan/error) dan icon terkait, dengan teks yang mendukung Pango markup, gunakanlah gtk.MessageDialog.

gtk.MessageDialog(parent=None, flags=0, type=gtk.MESSAGE_INFO, buttons=gtk.BUTTONS_NONE, message_format=None)

- parent: parent dialog
- flags: kombinasi gtk.DIALOG_MODAL dan gtk.DIALOG_DESTROY_WITH_PARENT (lihat pembahasan gtk.Dialog)
- type: jenis dialog, salah satu dari: gtk.MESSAGE_INFO, gtk.MESSAGE_WARNING, gtk.MESSAGE_QUESTION atau gtk.MESSAGE_ERROR.
- buttons: tombol-tombol yang telah didefinisikan: gtk.BUTTONS_NONE, gtk.BUTTONS_OK, gtk.BUTTONS_CLOSE, gtk.BUTTONS_CANCEL, gtk.BUTTONS_YES_NO, atau gtk.BUTTONS_OK_CANCEL.
- message_format: string pesan.

Catatan:
- Dialog mendukung rendering markup Pango. Untuk mengatur pesan agar dilengkapi dengan markup, gunakan: gtk.MessageDialog.set_markup(str). Lebih lanjut tentang Pango bisa ditemukan pada referensi class.
- Dialog mendukung teks tambahan/sekunder. Atur dengan:
  - gtk.MessageDialog.format_secondary_text(message_format)
  - gtk.MessageDialog.format_secondary_markup(message_format)
- Untuk mengganti gambar yang digunakan pada dialog, gunakan: gtk.MessageDialog.set_image(image). Argumen image merupakan gtk.Image.
- Widget gtk.MessageDialog diturunkan dari gtk.Dialog, lihatlah juga pembahasan gtk.Dialog.
- Untuk mengatur title dialog, gunakan: gtk.Window.set_title(title).
- Contoh berbagai dialog bisa dilihat pada msgdialogs.py.

msgdialogs.py:
```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_size_request(400,100)
        self.window.connect('destroy', gtk.main_quit)

        self.hbox = gtk.HBox()

        b1 = gtk.Button('Info')
        b1.connect('clicked', self.do_info, self.window)
        b2 = gtk.Button('Warning')
        b2.connect('clicked', self.do_warning, self.window)
        b3 = gtk.Button('Question')
        b3.connect('clicked', self.do_question, self.window)
        b4 = gtk.Button('Error')
        b4.connect('clicked', self.do_error, self.window)

        self.hbox.pack_start(b1)
```

```
        self.hbox.pack_start(b2)
        self.hbox.pack_start(b3)
        self.hbox.pack_start(b4)

        self.window.add(self.hbox)
        self.window.show_all()

    def do_info(self, widget, parent):
        d = gtk.MessageDialog(parent, gtk.DIALOG_MODAL,
                gtk.MESSAGE_INFO, gtk.BUTTONS_OK,
                'Hi!')

        img = gtk.Image()
        img.set_from_file('./smile.png')
        img.show()

        d.set_image(img)

        d.set_title('Smile :)')

        d.run()
        d.destroy()

    def do_warning(self, widget, parent):
        d = gtk.MessageDialog(parent, gtk.DIALOG_MODAL,
                gtk.MESSAGE_WARNING, gtk.BUTTONS_OK,
                'Warning')
        d.run()
        d.destroy()

    def do_question(self, widget, parent):
        d = gtk.MessageDialog(parent, gtk.DIALOG_MODAL,
                gtk.MESSAGE_QUESTION, gtk.BUTTONS_YES_NO)
        d.set_markup('Are you <b>ready</b>?')
        d.format_secondary_markup('We will take off <u>soon</u>.')

        result = d.run()
        if result == gtk.RESPONSE_YES:
            print 'Great!'
        else:
            print 'Hurry up!'

        d.destroy()

    def do_error(self, widget, parent):
        d = gtk.MessageDialog(parent, gtk.DIALOG_MODAL,
                gtk.MESSAGE_ERROR, gtk.BUTTONS_OK,
                'Unrecoverable error occured.')
        d.run()
        d.destroy()
```

```
app = Main()
gtk.main()
```

8. Lain-lain

Timeout

Untuk melakukan tindakan setiap interval tertentu, developer bisa mempergunakan gobject.timeout_add().

Untuk menambahkan timeout, gunakan:

```
gobject.timeout_add(interval, callback, ...)
```

- interval: interval waktu dalam satuan mili detik.
- callback: fungsi yang akan dipanggil setiap interval.
- ...: argumen opsional yang akan dilewatkan ke callback.
- Mengembalikan integer ID event source

Definisi callback
Callback adalah fungsi biasa, namun dengan catatan berikut:
- Jumlah argumen harus sama dengan yang dilewatkan melalui gobject.timeout_add().
- Apabila callback ingin dipanggil lagi, maka callback harus mengembalikan True.

Menghapus timeout
Untuk menghapus timeout yang dibuat, sehingga callback tidak lagi dipanggil setiap interval tertentu, selain mengembalikan False pada callback, developer juga bisa menggunakan:

```
gobject.source_remove(tag)
```

- tag adalah ID event source yang dikembalikan oleh gobject.timeout_add()

timeout.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk
import gobject
import time

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_size_request(300,200)
        self.window.connect('destroy', gtk.main_quit)

        self.source_id = -1

        self.action = gtk.Button('START')
```

```
        self.action.connect('clicked', self.action_check)

        self.label = gtk.Label()
        self.vbox = gtk.VBox()

        self.vbox.pack_start(self.action)
        self.vbox.pack_start(self.label)

        self.window.add(self.vbox)
        self.window.show_all()

    def action_check(self, widget):
        button_label = widget.get_label()
        if button_label == 'START':
            button_label = 'STOP'
            self.source_id = gobject.timeout_add(1000, self.timer)
        elif button_label == 'STOP':
            button_label = 'START'
            gobject.source_remove(self.source_id)
        widget.set_label(button_label)

    def timer(self):
        now = time.asctime()
        self.label.set_label(now)
        return True

app = Main()
gtk.main()
```

Idle

Untuk melakukan suatu tindakan ketika PyGTK sedang idle, developer bisa mempergunakan gobject.idle_add().

Untuk menambahkan idle, gunakan:

```
gobject.idle_add(callback, ...)
```

- callback: fungsi yang akan dipanggil ketika idle
- ...: argumen opsional yang akan dilewatkan ke callback.
- Mengembalikan integer ID event source

Definisi callback

Callback adalah fungsi biasa, namun dengan catatan berikut:

- Jumlah argumen harus sama dengan yang dilewatkan melalui gobject.idle_add().
- Apabila callback ingin dipanggil lagi, maka callback harus mengembalikan True.

Menghapus idle

Untuk menghapus idle yang dibuat, sehingga callback tidak lagi dipanggil ketika idle, selain mengembalikan False pada callback, developer juga bisa menggunakan:

```
gobject.source_remove(tag)
```

- tag adalah ID event source yang dikembalikan oleh gobject.idle_add()

idle.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk
import gobject

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_size_request(300,200)
        self.window.connect('destroy', gtk.main_quit)

        self.label = gtk.Label()

        gobject.timeout_add(100, self.timeout)
        gobject.idle_add(self.idle)

        self.window.add(self.label)
        self.window.show_all()

    def timeout(self):
        self.label.set_label('processing timeout')
        return True

    def idle(self):
        self.label.set_label('idle...')
        return True

app = Main()
gtk.main()
```

Events pending

Ketika berada dalam suatu tugas dengan waktu proses yang panjang, update ke komponen GUI yang dilakukan tidak akan berefek.

Sebagai contoh. Ketika suatu tombol ditekan, di dalam callback, perulangan dari 0 sampai 999 akan dilakukan. Untuk setiap perulangan, counter akan ditampilkan pada gtk.Label, dimaksudkan sebagai update. Pada kenyataannya, yang terjadi begitu tombol ditekan adalah:

- Program akan freeze sebentar (mengulang 0 sampai 999).
- Pada akhirnya, gtk.Label akan berisikan angka 999.

Sementara, yang diinginkan adalah, setiap dari 0 sampai 999 tersebut akan ditampilkan pada label. Hal yang diinginkan tersebut tidak terjadi karena:
- Semua event GTK, termasuk update pada window dan komponen lain, ditangani di dalam mainloop.
- Ketika kode-kode di dalam callback dikerjakan, mainloop tidak dapat menangani event.

Oleh karena itu, apa yang perlu dilakukan developer adalah meminta GTK untuk memroses event-event yang pending ketika proses sedang dilakukan. Caranya, sisipkan kode berikut pada bagian kode yang mungkin memakan waktu lama ketika dikerjakan:

```
 while gtk.events_pending():
   gtk.main_iteration(False)
```

Sebagai catatan, ini akan efektif ketika suatu proses merupakan rangkaian dari banyak subproses lain.

Cobalah memodifikasi contoh eventpending.py sehingga pemrosesan event-event pending tidak dilakukan dan bandingkan hasilnya.

evenpending.py:

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class Main:
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.connect('destroy', gtk.main_quit)

        self.vbox = gtk.VBox()

        self.label = gtk.Label()
        self.button = gtk.Button('Make me busy')
        self.button.connect('clicked', self.do_busy, self.label)

        self.vbox.pack_start(self.label)
        self.vbox.pack_start(self.button)

        self.window.add(self.vbox)

        self.window.show_all()

    def do_busy(self, widget, label):
        for i in range(1000):
```

```
            while gtk.events_pending():
                gtk.main_iteration(False)

            label.set_label('%d' %(i))

app = Main()
gtk.main()
```