

— Jollpi —

# A Lightweight, Simple and Reliable Text Editor with Python 3, GTK4 and GtkSourceView 5

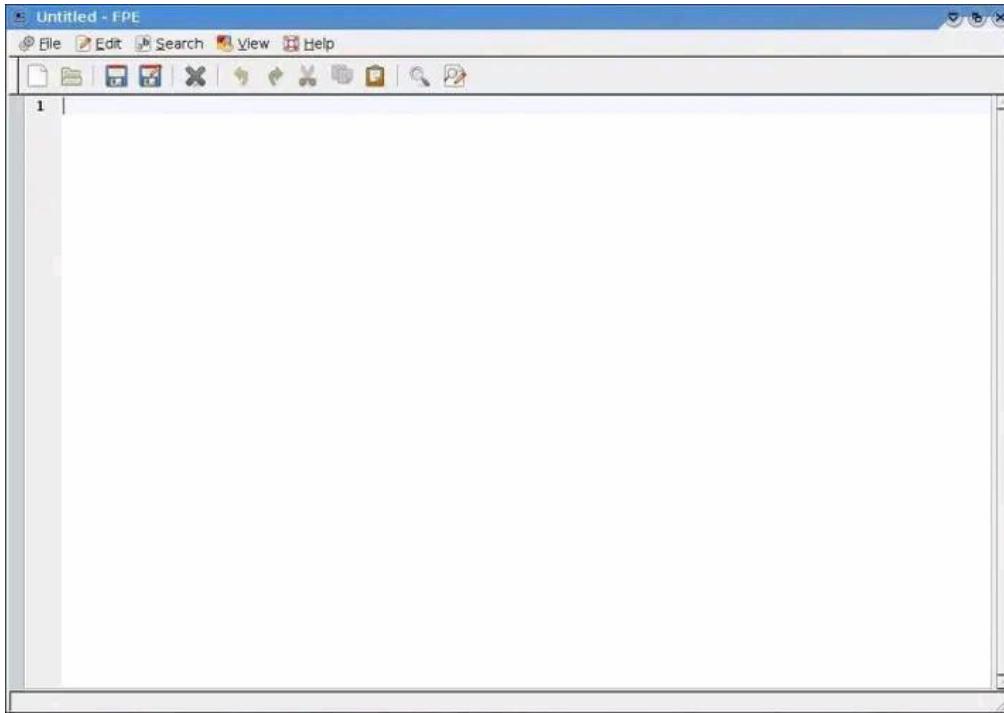
Zulfian  
GNOME Asia Summit 2025 — Tokyo, Japan  
14 December 2025

# About Me

---

- Python Developer
- Linux user since 2008 (with a very long pause)
- Creator of Jollpi (2010 → rewrite 2025)

# Flashback: Jollpi 2010



*First prototype*

A screenshot of a window titled "main.py - JOLLPI". It shows a multi-tabbed interface with several tabs visible in the tab bar. The active tab is "main.py", which contains Python code. The code is as follows:

```
1 #!/usr/bin/python
2
3 import sys
4 LIBS = "/usr/share/jollpi"
5 sys.path.append(LIBS)
6
7 try:
8     import pygtk
9     pygtk.require('2.0')
10    import gtk
11 except ImportError:
12     print "please install python-gtk"
13     sys.exit(1)
14
15 import mainPro as mainPro
16
17 def main():
18     mainpr = mainPro.MainWindow()
19     mainpr.show_all()
20     gtk.main()
21
22 if __name__ == "__main__":
23     try:
24         main()
25     except KeyboardInterrupt:
26         print "Bye"
```

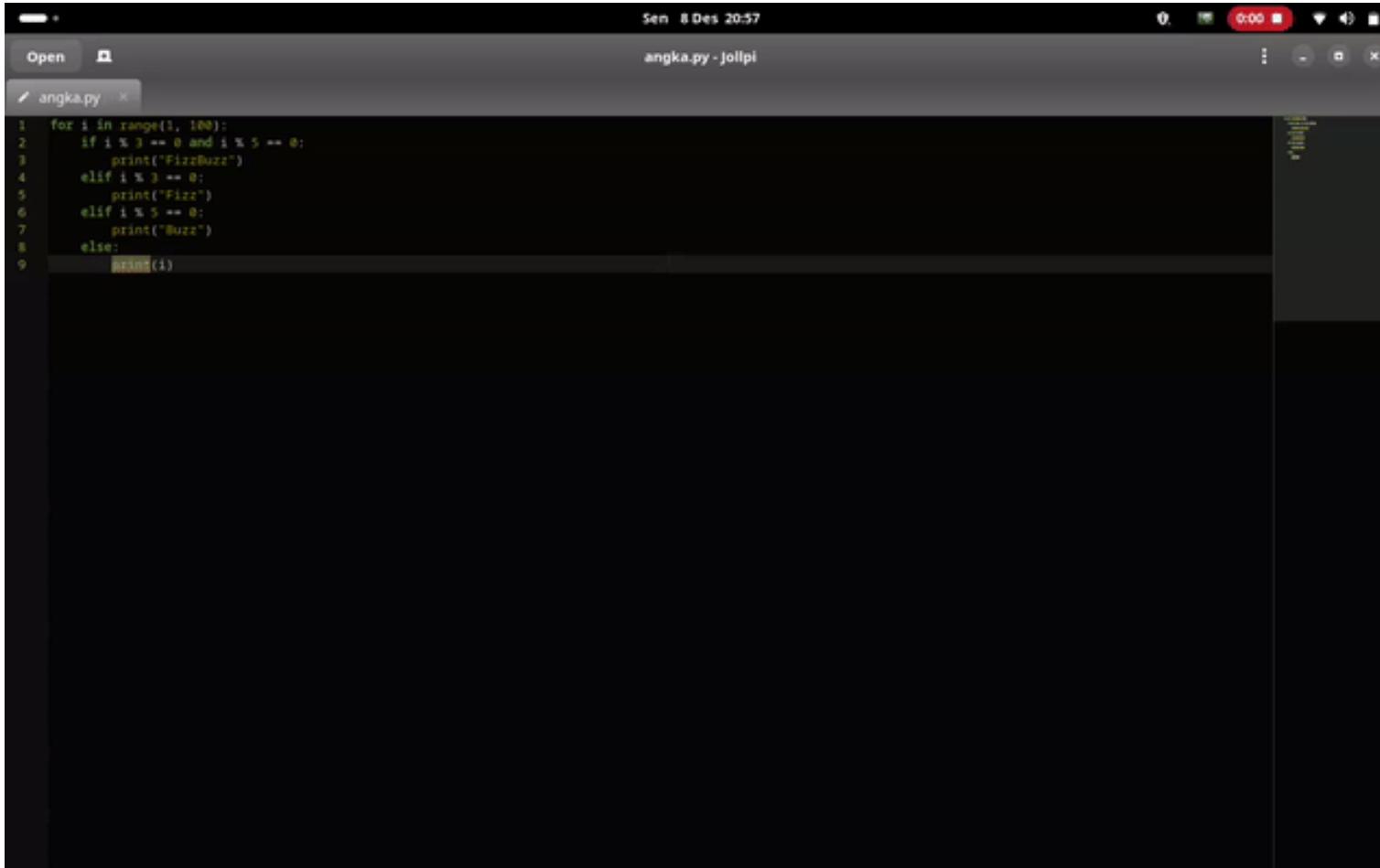
The status bar at the bottom right indicates "Line: 1, Col: 1" and "INS".

*Last stable release*

**Python 2 – GTK2 – GtkSourceView 2 – Minimal – Multi-tab**

- » SourceForge: [sourceforge.net/projects/jollpi](http://sourceforge.net/projects/jollpi)
- » GitLab: [gitlab.com/zulfian1732/jollpi2-legacy](https://gitlab.com/zulfian1732/jollpi2-legacy)

# Jollpi 2025 — The Rewrite



A screenshot of a Jollpi code editor window titled "angka.py - jollpi". The status bar at the top right shows "Sen 8 Des 20:57". The code editor displays the following Python script:

```
1 for i in range(1, 100):
2     if i % 3 == 0 and i % 5 == 0:
3         print("FizzBuzz")
4     elif i % 3 == 0:
5         print("Fizz")
6     elif i % 5 == 0:
7         print("Buzz")
8     else:
9         print(i)
```

# Why Rewrite ?

- ✓ Python 2 is end-of-life
- ✓ GTK4 is a completely different world
- ✓ GtkSourceView 5 introduces a new approach
- ✓ User expectations changed
- ✓ Keep it light and minimal

# Philosophy

---

- ✓ Simple and responsive – even with large files
- ✓ Only core components: Python, GTK4, GtkSourceView5
- ✓ Maintainable codebase
- ✓ Typing should feel effortless

# Architecture



Python 3



GTK4 (GObject, GDK, GLib)



GtkSourceView 5

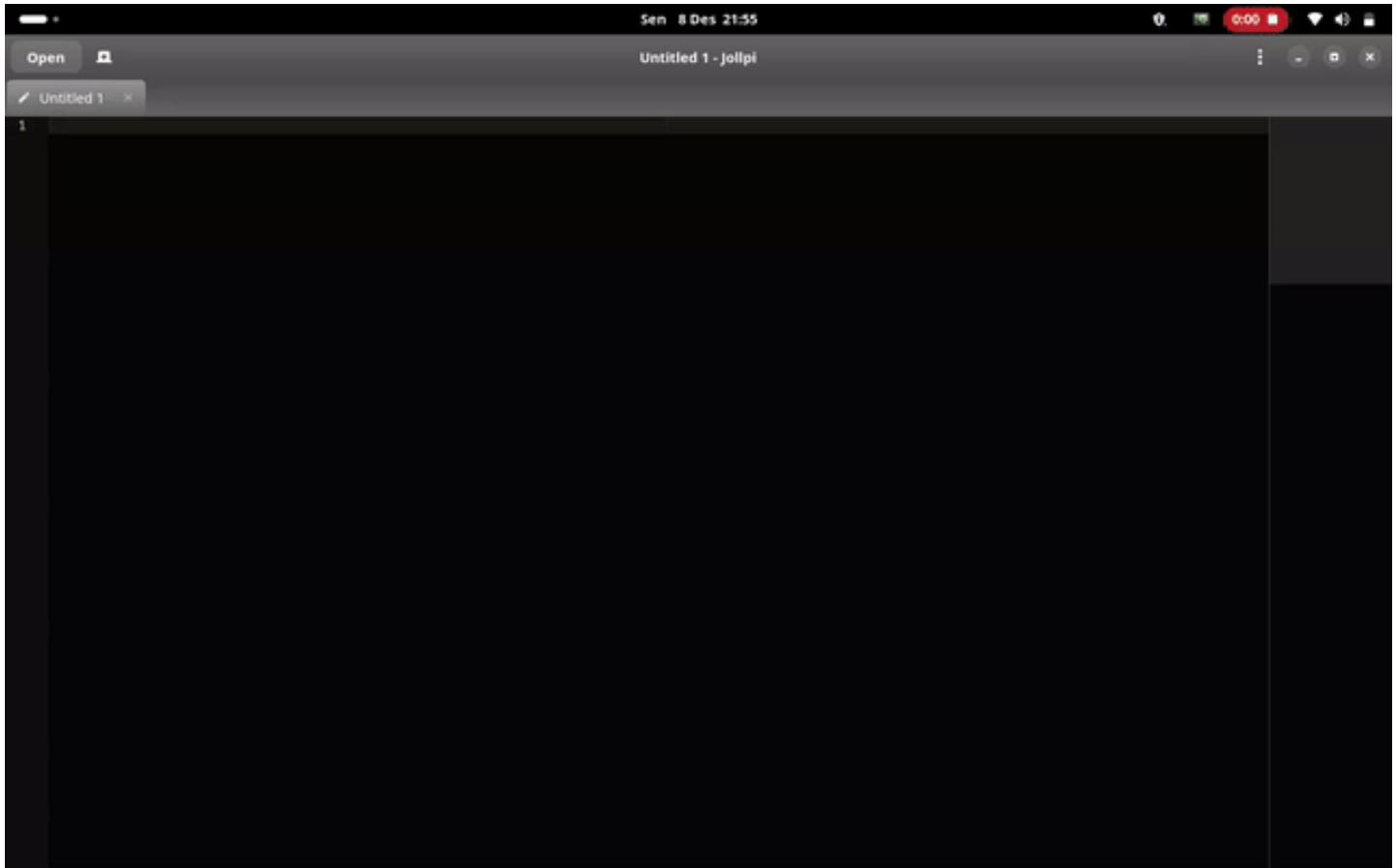


Gio File Handling + Monitoring

# Async Loading

- ✓ Uses GtkSourceView.FileLoader
- ✓ Fully cancellable
- ✓ UI stays responsive while loading

```
loader = GtkSource.FileLoader.new(  
    buffer,  
    src_file  
)  
loader.load_async(  
    priority,  
    cancellable,  
    None,  
    None,  
    on_loaded,  
    ()  
)
```



# Async Search

- ✓ Async search using SearchContext
- ✓ Live result count
- ✓ Replace step-by-step without UI freezing

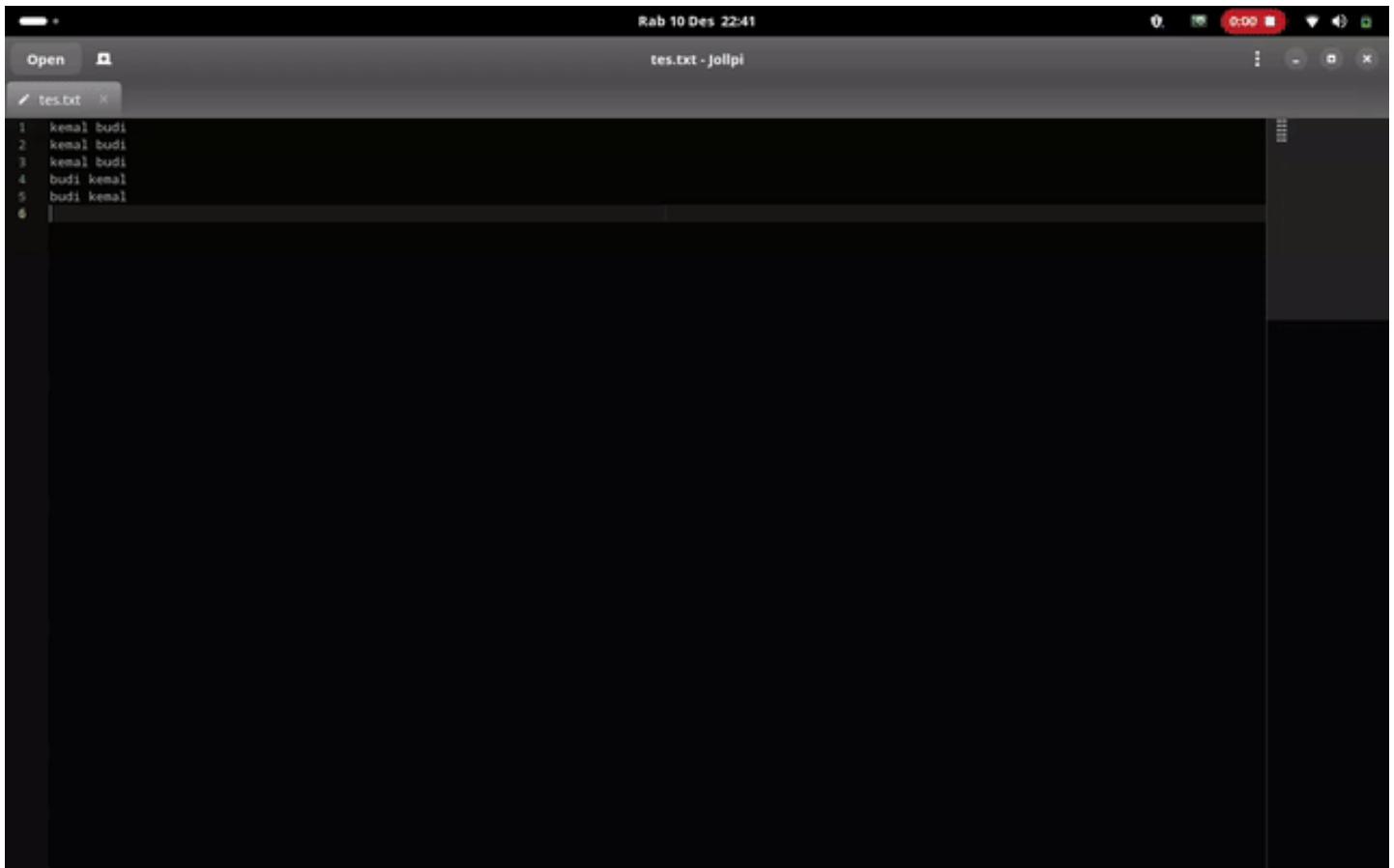
```
ctx = GtkSource.SearchContext.new(
    buffer,
    GtkSource.SearchSettings()
)
ctx.forward_async(
    iter,
    cancellable,
    on_match,
    None
)
```



# File Monitoring

- ✓ Uses Gio.FileMonitor
- ✓ Detects external modifications
- ✓ Offers reload / ignore options

```
gfile = Gio.File.new_for_path(  
    filepath  
)  
monitor = gfile.monitor_file(  
    Gio.FileMonitorFlags.NONE,  
    None  
)  # returns Gio.FileMonitor  
monitor.connect(  
    "changed",  
    on_file_changed  
)
```



# Modern GTK4 UI

The screenshot shows a code editor window titled "layout.py - Jollpi". The code is written in Python using the GTK4 library. The code defines a class `MyWindow` which inherits from `Gtk.ApplicationWindow`. It sets the title to "Contoh Sidebar Toggle + ListView" and the default size to 800x600. The code then creates a `Gtk.HeaderBar`, adds a toggle button and an add button to it, and sets it as the titlebar for the window. It also creates a main horizontal box and a sidebar listview with 50 items. The search and status bar overlays are shown at the bottom of the window.

```
7  class MyWindow(Gtk.ApplicationWindow):
8      def __init__(self, app):
9          super().__init__(application=app)
10         self.set_title("Contoh Sidebar Toggle + ListView")
11         self.set_default_size(800, 600)
12
13         # === HEADER BAR ===
14         header = Gtk.HeaderBar()
15         self.set_titlebar(header)
16
17         self.toggle_button = Gtk.Button(icon_name="sidebar-show-symbolic")
18         self.toggle_button.connect("clicked", self.on_toggle_sidebar)
19         header.pack_start(self.toggle_button)
20
21         # Tombol Add Menu
22         self.add_button = Gtk.Button(icon_name="mail-message-new-symbolic")
23         self.add_button.connect("clicked", self.on_add_menu)
24         header.pack_start(self.add_button)
25
26         # === MAIN BOX ===
27         main_box = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
28         self.set_child(main_box)
29
30         # === SIDEBAR LISTVIEW ===
31         self.items = Gio.ListStore.new(Gtk.StringObject)
32         for i in range(1, 26):  # bikin 50 item biar panjang
33             self.items.append(Gtk.StringObject.new(f"Chat {i}"))
34
35         factory = Gtk.SignalListItemFactory()
36         factory.connect("setup", self.on_factory_setup)
37         factory.connect("bind", self.on_factory_bind)
38
39         self.selection = Gtk.SingleSelection.new(self.items)
40         # self.selection.connect("notify::selected", self.on_item_selected)
41         self.selection.connect("selection-changed", self.on_selection_changed)
42
43         listview = Gtk.ListView(model=self.selection, factory=factory)
44         listview.add_css_class("navigation-sidebar")
45
```

HeaderBar

StatusBar Overlay

Search Overlay

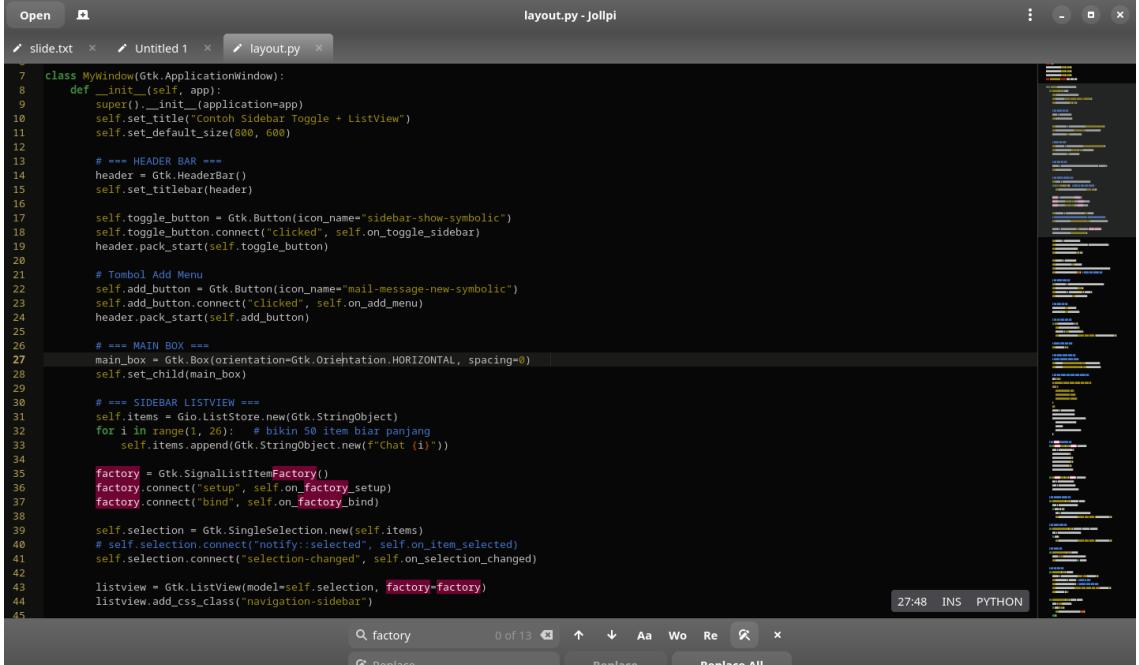
# Custom Theme

```
<style-scheme id="jollpi" _name="Jollpi" version="1.0">

<author>Zulfian</author>
<_description>
Just another scheme, which was taken and modified from oblivion
style
</_description>

<!-- Palette -->
<color name="butter1" value="#F9ED87"/>
<color name="butter2" value="#9F8E10"/>
<color name="butter3" value="#c4a000"/>
<color name="chameleon1" value="#8ae234"/>

<!-- ..... -->
<!-- Global Settings -->
<style name="selection" background="butter1"/>
<style name="cursor" foreground="aluminium2"/>
<style name="secondary-cursor" foreground="aluminium4-3-blend"/>
<style name="current-line" background="aluminium5"/>
<style name="draw-spaces" foreground="aluminium4"/>
<style name="background-pattern" background="aluminium6-5-blend"/>
```



The screenshot shows a code editor window titled "layout.py - Jollpi". The code is a Python script using the GTK library. It defines a class `MyWindow` that inherits from `Gtk.ApplicationWindow`. The script sets up a header bar with a toggle button for the sidebar. It also creates a main box containing a sidebar list view and a main area. The sidebar list view uses a signal list item factory to generate items. The code is annotated with comments explaining the structure and components of the application.

```
7  class MyWindow(Gtk.ApplicationWindow):
8      def __init__(self, app):
9          super().__init__(application=app)
10         self.set_title("Conico Sidebar Toggle + ListView")
11         self.set_default_size(800, 600)
12
13     # *** HEADER BAR ***
14     header = Gtk.HeaderBar()
15     self.set_titlebar(header)
16
17     self.toggle_button = Gtk.Button(icon_name="sidebar-show-symbolic")
18     self.toggle_button.connect('clicked', self.on_toggle_sidebar)
19     header.pack_start(self.toggle_button)
20
21     # Tombol Add Menu
22     self.add_button = Gtk.Button(icon_name="mail-message-new-symbolic")
23     self.add_button.connect('clicked', self.on_add_menu)
24     header.pack_start(self.add_button)
25
26     # *** MAIN BOX ***
27     main_box = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
28     self.set_child(main_box)
29
30     # *** SIDEBAR LISTVIEW ***
31     self.items = Gio.ListStore.new(Gtk.StringObject)
32     for i in range(1, 26): # bikin 50 item biar panjang
33         self.items.append(Gtk.StringObject.new(f"Chat ({i})"))
34
35     factory = Gtk.SignalListItemFactory()
36     factory.connect("setup", self.on_factory_setup)
37     factory.connect("bind", self.on_factory_bind)
38
39     self.selection = Gtk.SingleSelection.new(self.items)
40     # self.selection.connect("notify::selected", self.on_item_selected)
41     self.selection.connect("selection-changed", self.on_selection_changed)
42
43     listview = Gtk.ListView(model=self.selection, factory=factory)
44     listview.add_css_class("navigation-sidebar")
```

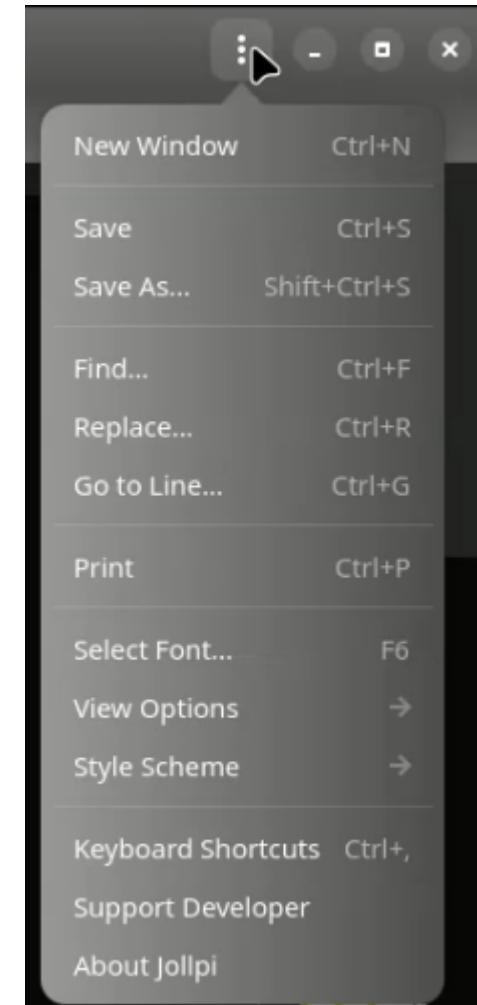
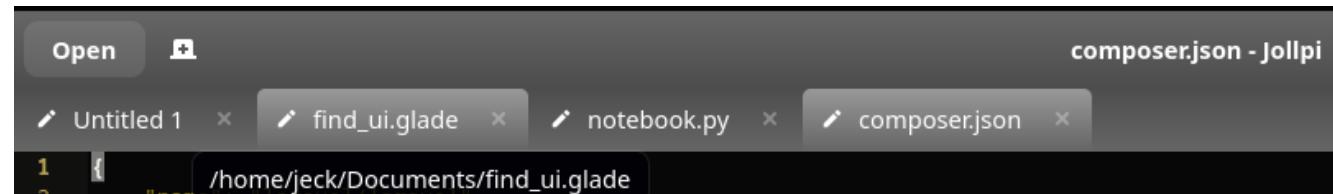
Optional custom theme inspired by Oblivion — modified and simplified.

# GTK Custom CSS

- ✓ Custom CSS via Gtk.CssProvider
- ✓ Consistent look and feel
- ✓ Lightweight — no extra libraries

```
provider = gtk.CssProvider()
provider.load_from_file(custom.css)

gtk.StyleContext.add_provider_for_display(
    gdk.Display.get_default(),
    provider,
    gtk.STYLE_PROVIDER_PRIORITY_USER
)
```

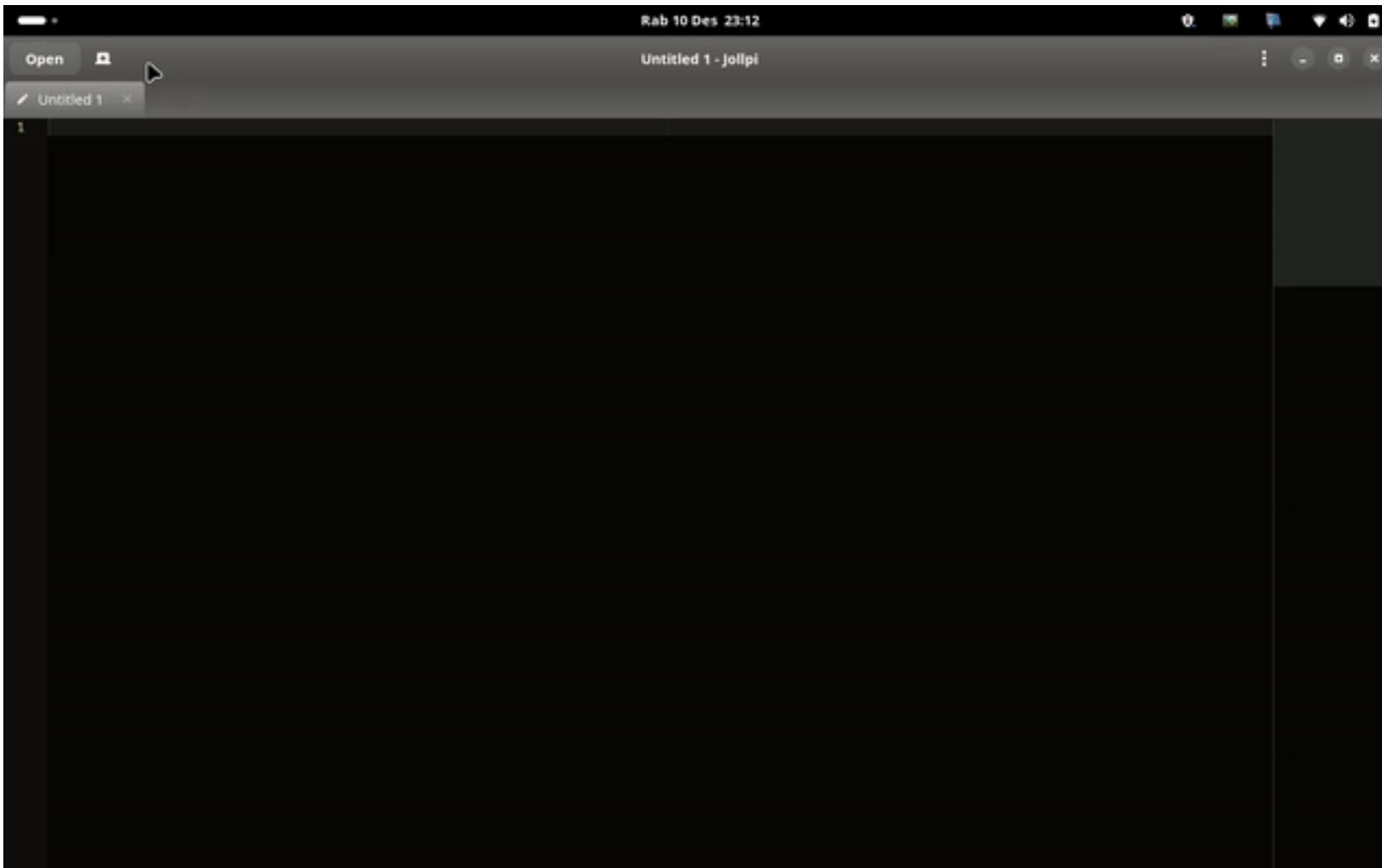


# Installer / install.sh

- ✓ Automated installation
- ✓ Simple, self-contained script
- ✓ Handles setup without extra tools

```
[zulfian@pdinp-PC jollpi3]$ ./install.sh
Usage: ./install.sh [-i | -u]
    -i : install
    -u : uninstall
[zulfian@pdinp-PC jollpi3]$ ]
```

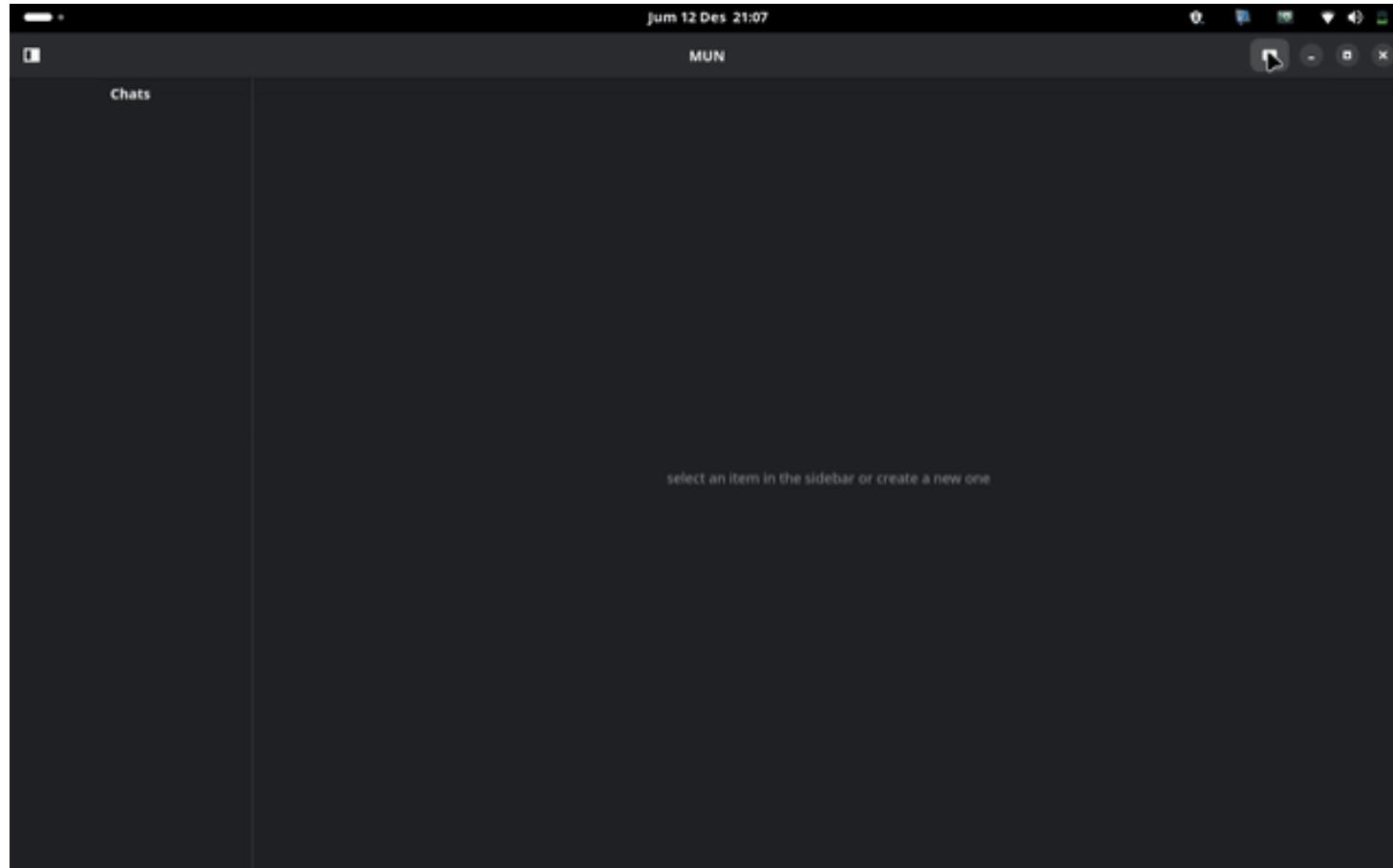
# Demo — Jollpi (2025)



# From Jollpi to MUN-AI

---

- Same toolkit
- Same lightweight philosophy
- Same maintainability mindset



# Jollpi is now public

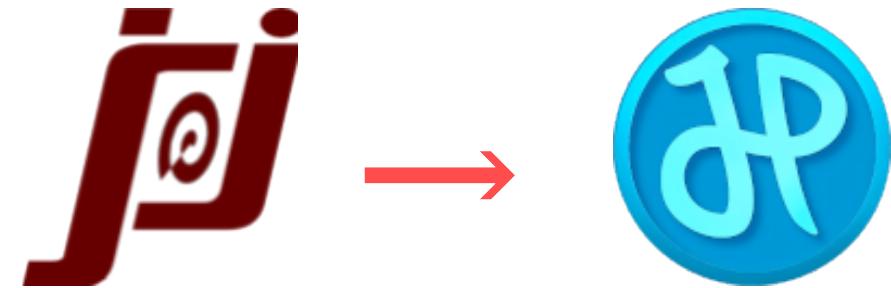
[gitlab.com/zulfian1732/jollpi-text-editor](https://gitlab.com/zulfian1732/jollpi-text-editor)



Use it. Learn from it

~ *Thank You* ~

"old code isn't failure.  
Sometimes it's just waiting  
for a better version of us."



- » LinkedIn: ZULFIAN S.Kom
- » GitLab: @zulfian1732
- » GitHub: @zulfian1732
- » Mastodon: @zulfian
- » E-mail: zulfian1732@gmail.com