



# How to Create GUI Applications Under Linux Desktop Using PyGObject

[Read](#)[Courses](#)[Practice](#)[Jobs](#)

The creation of applications in Linux can be done through various methods. But, the most efficient way of creating a GUI application in Linux can be done through PyGObject in Python. PyGObject is the next generation from the PyGTK library in Python, we can say that PyGObject = Python + GTK3. So, in this article, we will be creating a GUI application under a Linux environment using PyGObject.

**Note:** Make sure you have installed [Python on your Linux machine](#).

There are two ways for creating GUI applications in Linux.

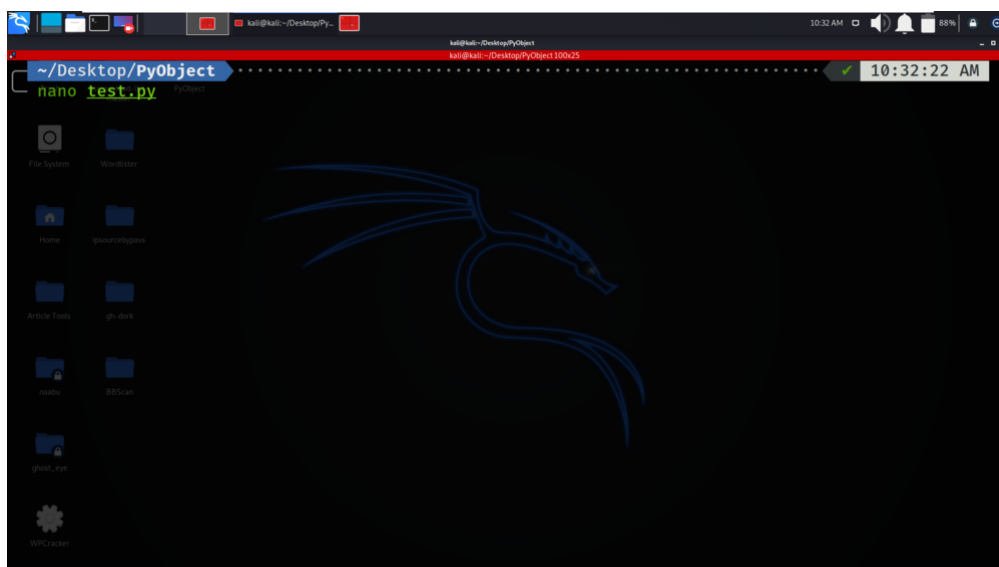
- The Code-Only Method
- Glade Designer Method

## Method 1: The Code Only Method

In this method, we will be creating the GUI components directly by program code, rather than using any drag and drop approach. So follow the below steps to create the application using the code-only method.

**Step 1:** Create the test.py file by using a text editor.

```
nano test.py
```



**Step 2:** Write the following code in test.py file.

## Python3

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

from gi.repository import Gtk

class ourwindow(Gtk.Window):

    def __init__(self):
        Gtk.Window.__init__(self, title="Demonstration\
of PyObject GUI Application Creation")
        Gtk.Window.set_default_size(self, 400,325)
        Gtk.Window.set_position(self, Gtk.WindowPosition.CENTER)

        button1 = Gtk.Button("GeeksforGeeks")
        button1.connect("clicked", self.whenbutton1_clicked)

        self.add(button1)

    def whenbutton1_clicked(self, button):
        print("GeeksforGeeks")

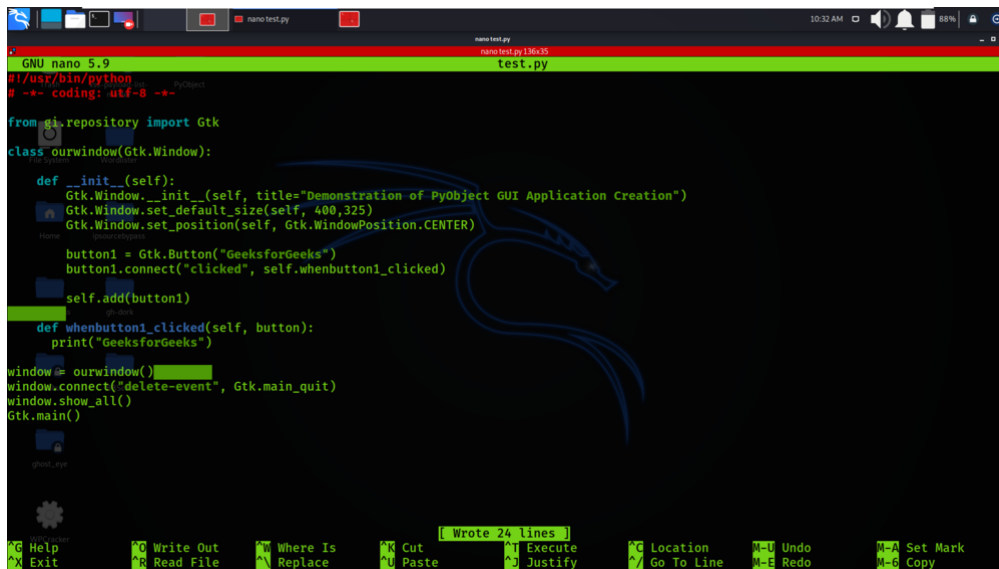
window = ourwindow()
window.connect("delete-event", Gtk.main_quit)
window.show_all()
Gtk.main()
```



### Code Explanation:

1. `#!/usr/bin/python3`: This line specifies the default Python3 interpreter path.
2. `# -*- coding: utf-8 -*-`: In this line of code, we have specified the default Unicode as UTF-8.
3. `from gi.repository import Gtk`: Here, we are importing the GTK 3 library.
4. `Class ourwindow(Gtk.Window)`: In this, we are creating a class named as `ourWindow` and setting the class object type as `Gtk.Window`.
5. `def __init__(self)`: In this, we are defining the main window components.
6. `Gtk.Window.__init__(self, title="Demonstration of PyObject GUI Application Creation")`: In this we are setting the title to the Window.
7. `Gtk.Window.set_default_size(self, 400,325)`: In this, we are setting up window width and height.
8. `Gtk.Window.set_position(self, Gtk.WindowPosition.CENTER)`: In this line of code, we are setting up the default position of the window as `CENTER`.
9. `button1 = Gtk.Button("GeeksforGeeks")`: In this we are creating a button named `GeeksforGeeks`.
10. `button1.connect("clicked", self.whenbutton1_clicked)`: In this, we are activating the event `clicked` when the button is been pressed.
11. `self.add(button1)`: In this, we are adding a button on the created window.

12. `def whenbutton1_clicked(self, button):`: In this, we are defining the function named `whenbutton1_clicked` and writing the event code into it.
13. `print("GeeksforGeeks")`: We are printing the GeeksforGeeks message.
14. `window = ourwindow()`: In this, we are creating a global variable which we can use later in the code.
15. `window.connect("delete-event", Gtk.main_quit):` In this, we are deleting all the widgets after we close our program window automatically.
16. `window.show_all()`: Showing the window.
17. `Gtk.main()`: Running the Gtk library.



```
GNU nano 5.9 test.py
#!/usr/bin/python
# -*- coding: utf-8 -*-

from gi.repository import Gtk

class ourwindow(Gtk.Window):

    def __init__(self):
        Gtk.Window.__init__(self, title="Demonstration of PyObject GUI Application Creation")
        Gtk.Window.set_default_size(self, 400,325)
        Gtk.Window.set_position(self, Gtk.WindowPosition.CENTER)

        button1 = Gtk.Button("GeeksforGeeks")
        button1.connect("clicked", self.whenbutton1_clicked)

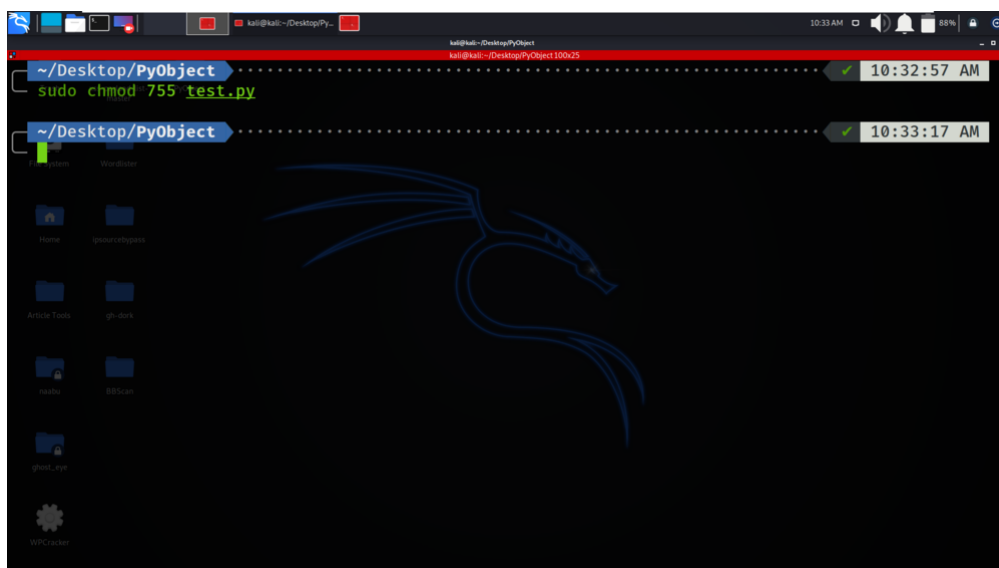
        self.add(button1)

    def whenbutton1_clicked(self, button):
        print("GeeksforGeeks")

window = ourwindow()
window.connect("delete-event", Gtk.main_quit)
window.show_all()
Gtk.main()
```

**Step 3:** Change the permissions of test.py file by using the following command

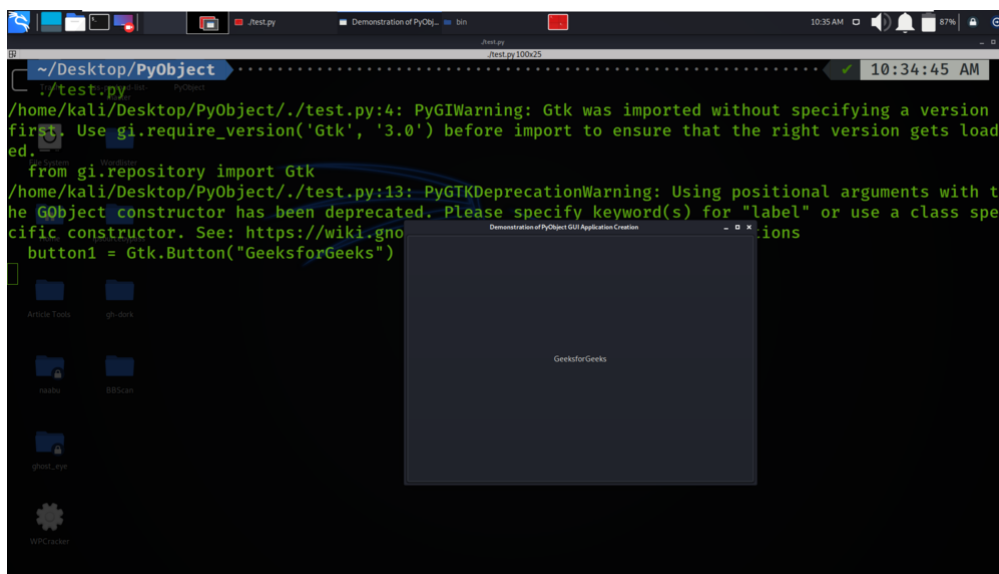
```
sudo chmod 755 test.py
```



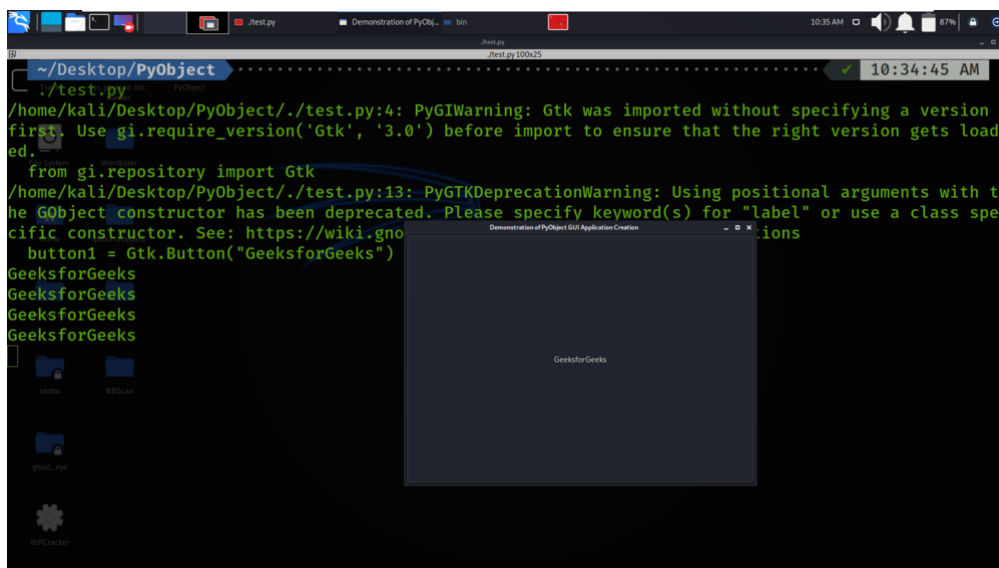
```
~/Desktop/PyObject
$ sudo chmod 755 test.py
$
```

**Step 4:** Run the test.py file by using the following command.

```
./test.py or
python3 test.py
```



**Step 5:** Now, you can see that the application window have been displayed and the button is been visible on the screen. So after clicking on the “GeeksforGeeks” button. The message is been printed on the terminal.



So, in this way, we can create GUI applications directly from the code itself.

## Method 2: Glade Designer Method

In this method, we will be creating the GUI components by using Glade Designer which provides the drag and drops facility of window components and also controls components like buttons, labels etc. So follow the below steps to create an application using Glade Designer method.

**Step 1:** Install the Glade package on a Linux environment by using the following command.

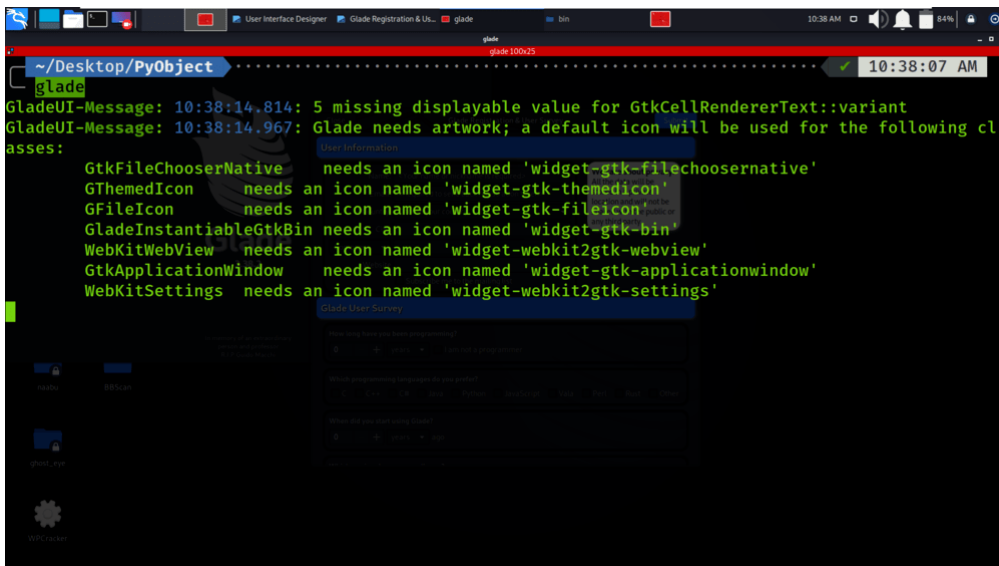
```
sudo apt-get install glade
```

```
~/Desktop/PyObject
sudo apt-get install glade
Reading package lists... Done
```

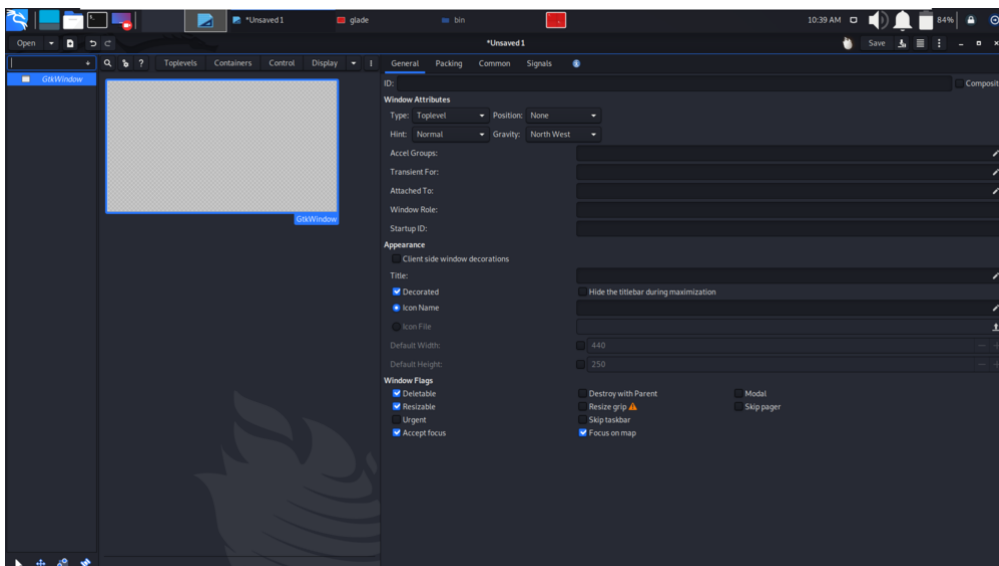
```
gnome-session-canberra kazam libatk-5-0 libatk-5-common libavresample4 libdb5.1 libevent-2.0-5
libgtksourceview2.0-0 libgtksourceview2.0-common libindicator3-7 libmpdec2 libncursesw5
libnugget-core-cil libplymouth4 libpoppler82 libpython3.8-minimal libpython3.8-stdlib
libreadline6 librest-0.7-0 libservlet3.1-java libssl1.0.0 libtepl-5-0 node-ansi node-jquery
node-jsonstream node-leven node-lockfile node-puka node-resolve-from node-through openjdk-8-jre
python-cairo python-dbus python-enchanted python-gobject-2 python-numpy python-six python2.6
python2.6-minimal python3-chameleon python3-editor python3-flask-babel python3-ipynb
python3-ipython-genutils python3-waitress python3-webtest python3-xdg python3-zope.component
python3-zope.hookable python3.8 python3.8-minimal qt5-gtk2-platformtheme ruby-connection-pool
ruby-molinillo ruby-net-http-persistent ruby-thor xfce4-mailwatch-plugin
xfce4-smartbookmark-plugin xfce4-statusnotifier-plugin xfce4-weather-plugin
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
devhelp devhelp-common fontconfig-config gir1.2-atk-1.0 gir1.2-atspi-2.0 gir1.2-gtk-3.0
gir1.2-pango-1.0 icu-devtools libatk-bridge2.0-dev libatk1.0-0 libatk1.0-data libatk1.0-dev
libatk1.0-doc libatspi2.0-dev libblkid-dev libblkid1 libbrotli-dev libbrotli1
libcairo-script-interpreter2 libcairo2-dev libdatatr1-dev libdeflate-dev libdevhelp-3-6
libegl-dev libegl1 libegl1-mesa-dev libepoxy-dev libfontconfig-dev libfontconfig1
```

Step 2: Launch the Glade Designer application by terminal itself or by navigating through all apps on Linux applications.

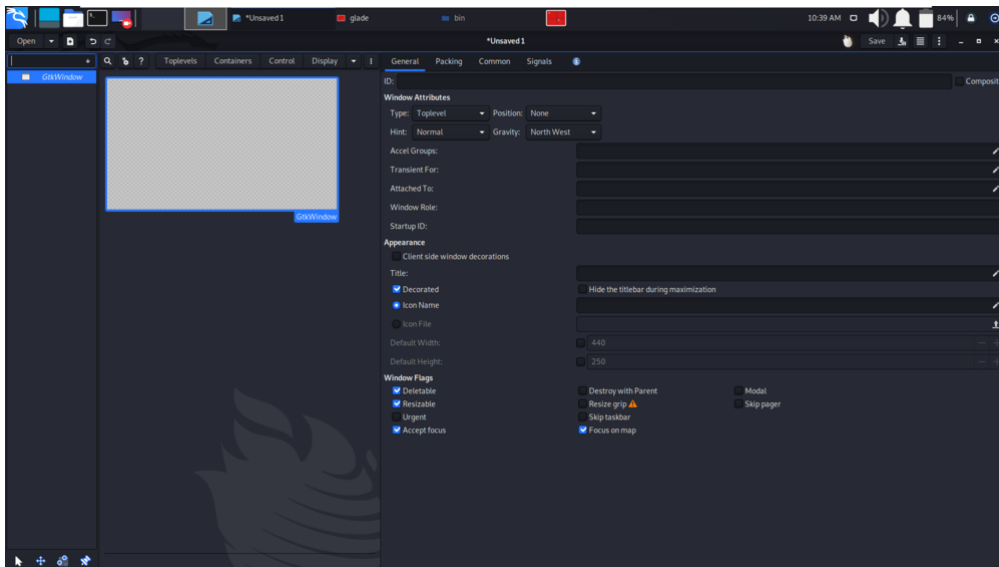
glade



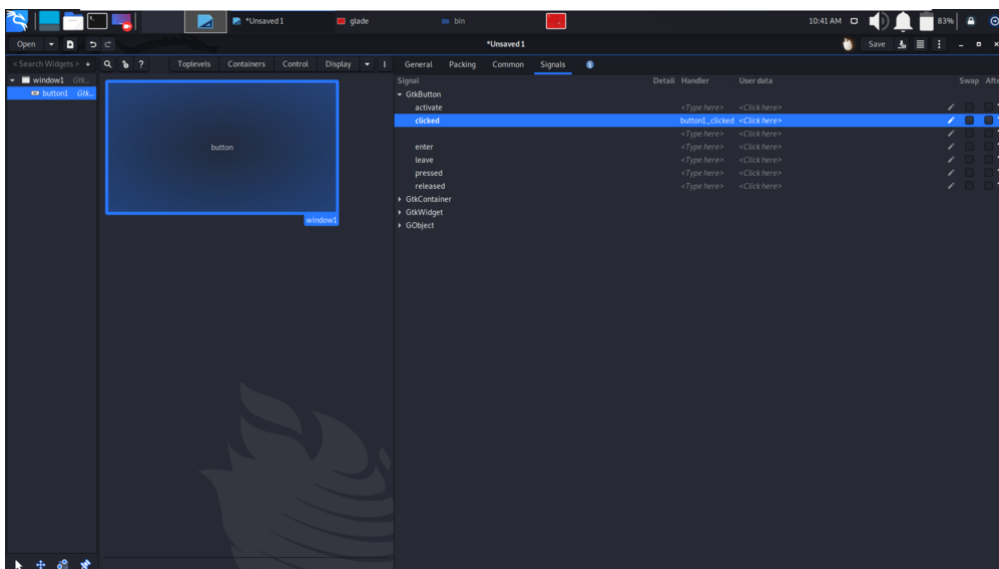
Step 3: The following screenshot displays the GUI of Glade application.



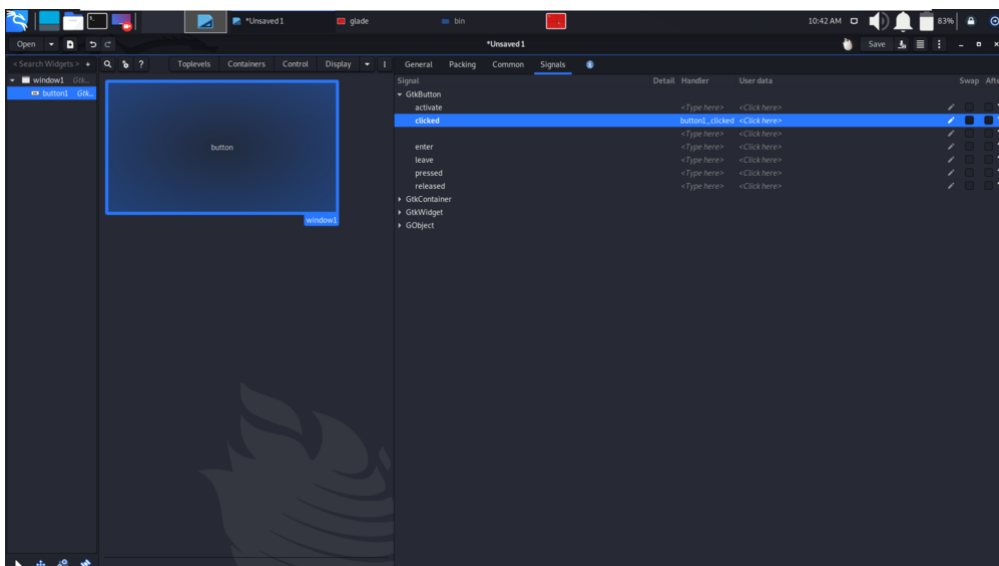
**Step 4:** Now, we will be creating the Window on which control objects are been placed. So, select the Window(GtkWindow) from Top Levels option.



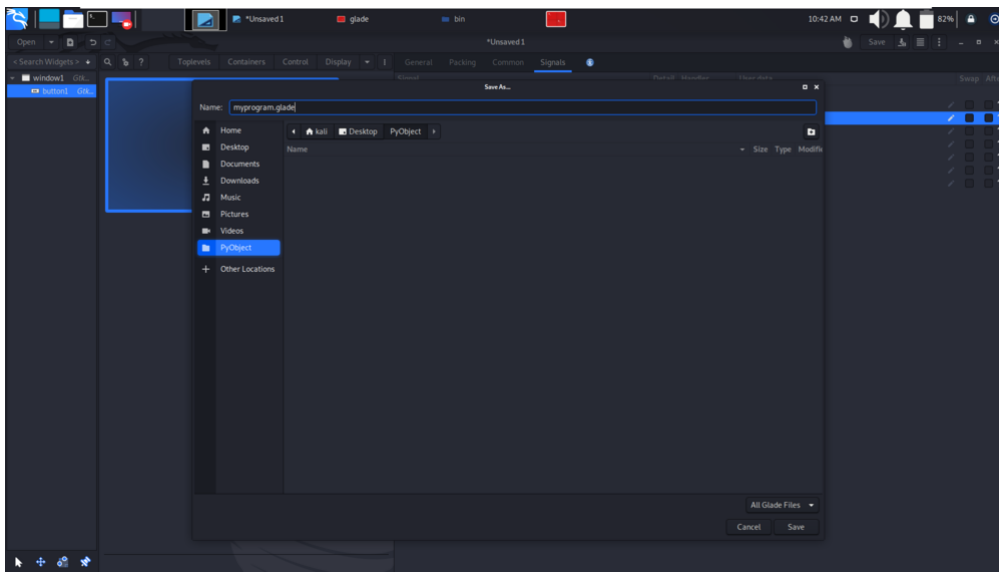
**Step 5:** Now, place the button on the Window from Control option.



**Step 6:** Make sure to click the clicked option in signals tab and give the name to the handler as button1\_clicked.



**Step 7:** Click on the Save option and save the file as myprogram.glade.



**Step 8:** Now, create a new python file by using a text editor and adding the below line of code into it. Make sure to add the file name of .glade file in program code.

```
nano glademethod.py
```

## Python3

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from gi.repository import Gtk

class Handler:
    def button_1clicked(self, button):
        print("Hello GeeksForGeeks using Glade")

builder = Gtk.Builder()
builder.add_from_file("myprogram.glade")
builder.connect_signals(Handler())

ournewbutton = builder.get_object("button1")
ournewbutton.set_label("Demo using Glade!")

window = builder.get_object("window1")

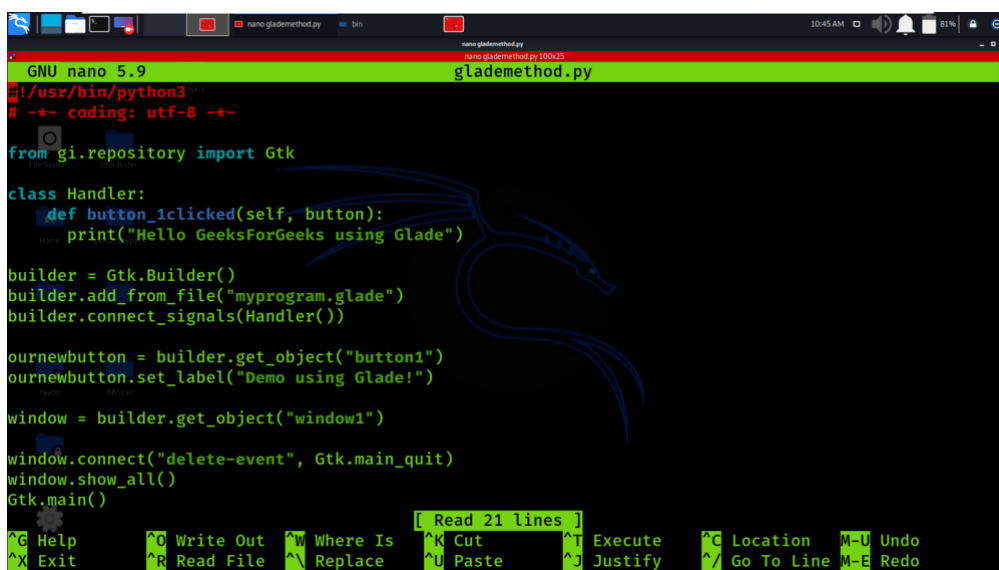
window.connect("delete-event", Gtk.main_quit)
window.show_all()
Gtk.main()
```

### Code Explanation:

1. `#!/usr/bin/python3`: This line specifies the default Python3 interpreter path.
2. `# -*- coding: utf-8 -*-`: In this line of code, we have specified the default Unicode as UTF-8.
3. `from gi.repository import Gtk`: Here, we are importing the GTK 3 library.
4. `class Handler`: In this line, we are creating a new class named Handler.



5. `def button1_clicked(self, button):`: In this, we are defining the function named `button1_clicked` and writing the event code into it.
6. `print("Hello GeeksForGeeks using Glade")`: We are printing the Hello GeeksForGeeks using Glade message.
7. `builder = Gtk.Builder()`: We are creating a global variable `builder` which is used to import the glade file.
8. `builder.add_from_file("myprogram.glade")`: Here we're importing the "myprogram.glade" file to use it as a default GUI for our program.
9. `builder.connect_signals(Handler())`: In this we are connecting glade file with handler class.
10. `ournewbutton.set_label("Hello, World!")`: We are setting the default button text as Demo using Glade!
11. `window = builder.get_object("window1")`: In this line we have called the "window1" object from the .glade file in order to show it later in the program.
12. `window.connect("delete-event", Gtk.main_quit)`: In this, we are deleting all the widgets after we close our program window automatically.
13. `window.show_all()`: Showing the window.
14. `Gtk.main()`: Running the Gtk library.



```
GNU nano 5.9 glademethod.py
#!/usr/bin/python3
# -*- coding: utf-8 -*-

from gi.repository import Gtk

class Handler:
    def button1_clicked(self, button):
        print("Hello GeeksForGeeks using Glade")

builder = Gtk.Builder()
builder.add_from_file("myprogram.glade")
builder.connect_signals(Handler())

ournewbutton = builder.get_object("button1")
ournewbutton.set_label("Demo using Glade!")

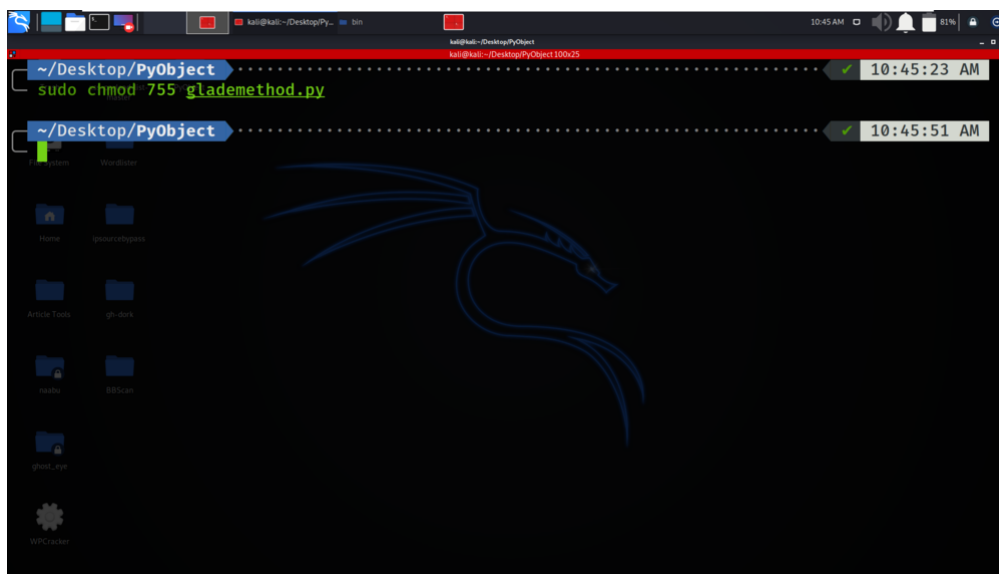
window = builder.get_object("window1")

window.connect("delete-event", Gtk.main_quit)
window.show_all()
Gtk.main()
```

**Step 9:** Change the permissions of `glademethod.py` file by using the following command.

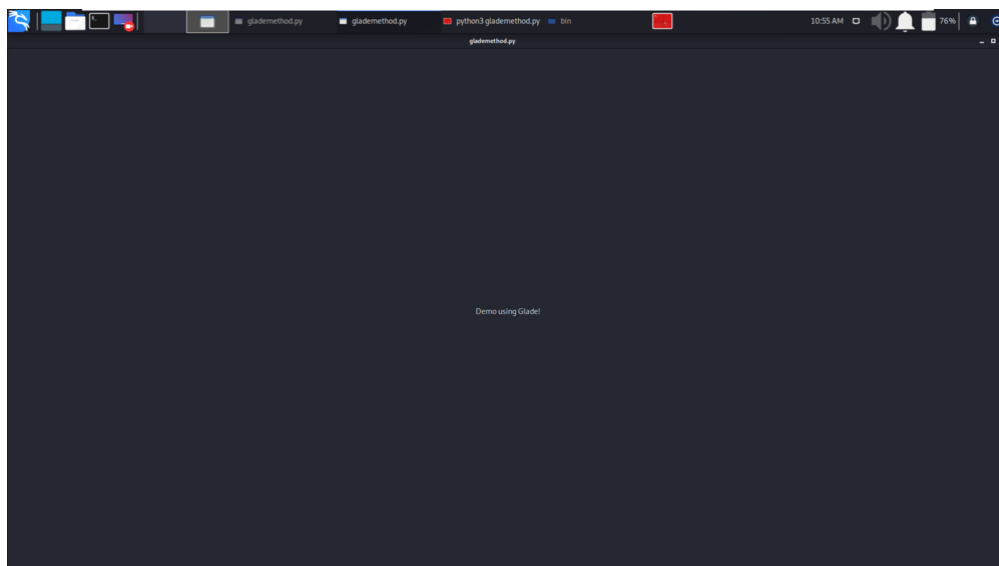
```
sudo chmod 755 glademethod.py
```



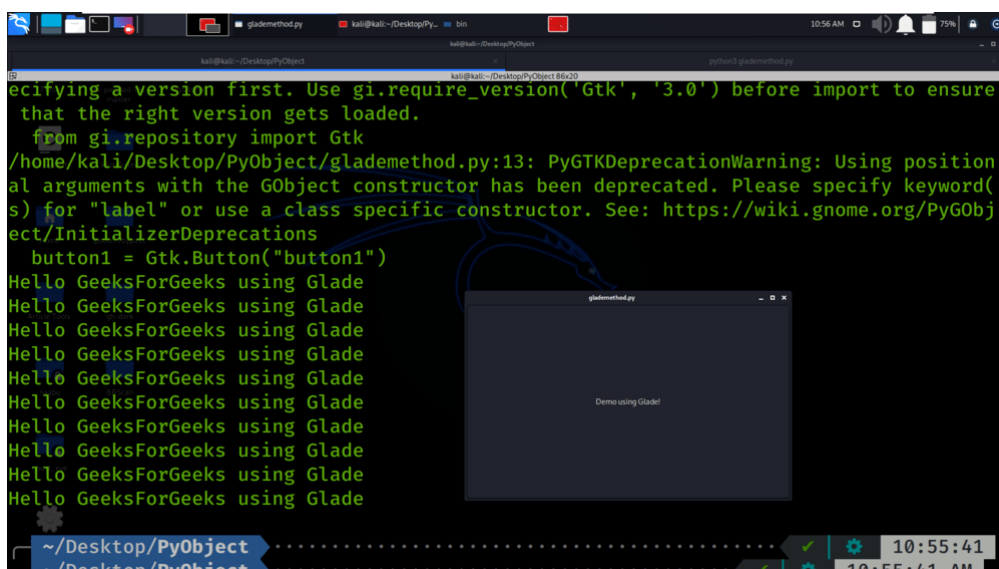


**Step 10:** Run the glademethod.py file by using the following command. You will see the Window opened and the created button will be displayed.

```
python3 glademethod.py
```



**Step 11:** After clicking on the button on the window, the message is been displayed on the terminal.



So, in this way we can create GUI applications using Glade Designer method only by dragging and dropping the components.

Don't miss your chance to ride the wave of the data revolution! Every industry is scaling new heights by tapping into the power of data. Sharpen your skills, become a part of the hottest trend in the 21st century.

Dive into the future of technology - explore the [Complete Machine Learning and Data Science Program](#) by GeeksforGeeks and stay ahead of the curve.

Last Updated : 20 Feb, 2022

👍 1



< Previous

Next >

**How to Conduct a Two Sample T-Test in Python**

**How To Save The Network In XML File Using PyBrain**

Share your thoughts in the comments

Add Your Comment

## Similar Reads

Create More Advance GUI Applications Using PyGobject Tool in Linux

How to Install PyGObject in Linux?

Running GUI Applications on Docker in Linux

Designing GUI applications Using PyQt in Python

How to Run GUI Based Applications inside Docker?

Create Desktop Shortcuts Using The Terminal on Ubuntu

Difference Between Desktop Environment VS Window Manager in Linux

How To Install KDE Plasma Desktop on Linux Mint

How to kill processes on the Linux Desktop with xkill

9 Best Linux Desktop Environments That You Can Use

## Complete Tutorials

Python API Tutorial: Getting Started with APIs

Brain Teasers

SDLC Models | Types, Phases, When to use

Advanced Python Tutorials

Python Automation Tutorial



abhishek...

**Article Tags :** [Geeks-Premier-League-2022](#) , [Picked](#) , [python-modules](#) , [Geeks Premier League](#) , [Linux-Unix](#) , [Python](#)

**Practice Tags :** [python](#)

**Additional Information**



A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305



## Company

About Us  
Legal  
Careers  
In Media  
Contact Us  
Advertise with us

## Explore

Job-A-Thon  
Hiring Challenge  
Hack-A-Thon  
GfG Weekly  
Contest  
Offline Classes  
(Delhi/NCR)

## Languages

Python  
Java  
C++  
PHP  
GoLang  
SQL

## DSA

Data Structures  
Algorithms  
DSA for Beginners  
Basic DSA  
Problems  
DSA Roadmap

## Data Science & ML

Data Science With  
Python  
Data Science For  
Beginner  
Machine Learning  
Tutorial

## HTML & CSS

HTML  
CSS  
Bootstrap  
Tailwind CSS  
SASS  
LESS

GFG Corporate Solution Placement Training Program Apply for Mentor	DSA in JAVA/C++ Master System Design Master CP GeeksforGeeks Videos	R Language Android Tutorial	Top 100 DSA Interview Problems DSA Roadmap by Sandeep Jain All Cheat Sheets	ML Maths Data Visualisation Tutorial Pandas Tutorial NumPy Tutorial NLP Tutorial Deep Learning Tutorial	Web Design
Python	Computer Science	DevOps	Competitive Programming	System Design	JavaScript
Python Programming Examples Django Tutorial Python Projects Python Tkinter Web Scraping OpenCV Python Tutorial Python Interview Question	Operating Systems Computer Network Database Management System Software Engineering Digital Logic Design Engineering Maths	Git AWS Docker Kubernetes Azure GCP DevOps Roadmap	Top DS or Algo for CP Top 50 Tree Top 50 Graph Top 50 Array Top 50 String Top 50 DP Top 15 Websites for CP	What is System Design Monolithic and Distributed SD High Level Design or HLD Low Level Design or LLD Crack System Design Round System Design Interview Questions Grokking Modern System Design	TypeScript ReactJS NextJS AngularJS NodeJS Express.js Lodash Web Browser
NCERT Solutions	School Subjects	Commerce	Management & Finance	UPSC Study Material	SSC/ BANKING
Class 12	Mathematics	Accountancy Business Studies	Management	Polity Notes	SSC CGL Syllabus
Class 11	Physics	Indian Economics	HR Managment	Geography Notes	SBI PO Syllabus
Class 10	Chemistry	Macroeconomics	Income Tax	History Notes	SBI Clerk Syllabus
Class 9	Biology	Microeconomics	Finance	Science and Technology Notes	IBPS PO Syllabus
Class 8	Social Science	Statistics for Economics	Economics	Economy Notes	IBPS Clerk Syllabus
Complete Study Material	English Grammar			Ethics Notes Previous Year Papers	SSC CGL Practice Papers
Colleges	Companies	Preparation	Exams	More	Write & Earn

[Indian Colleges](#)[Admission &](#)[Campus](#)[Experiences](#)[Top Engineering](#)[Colleges](#)[Top BCA Colleges](#)[Top MBA Colleges](#)[Top Architecture](#)[College](#)[Choose College](#)[For Graduation](#)[IT Companies](#)[Software](#)[Development](#)[Companies](#)[Artificial](#)[Intelligence\(AI\)](#)[Companies](#)[CyberSecurity](#)[Companies](#)[Service Based](#)[Companies](#)[Product Based](#)[Companies](#)[PSUs for CS](#)[Engineers](#)

## Corner

[Company Wise](#)[Preparation](#)[Preparation for](#)[SDE](#)[Experienced](#)[Interviews](#)[Internship](#)[Interviews](#)[Competitive](#)[Programming](#)[Aptitude](#)[Preparation](#)[Puzzles](#)[JEE Mains](#)[JEE Advanced](#)[GATE CS](#)[NEET](#)[UGC NET](#)

## Tutorials

[Software](#)[Development](#)[Software Testing](#)[Product](#)[Management](#)[SAP](#)[SEO](#)[Linux](#)[Excel](#)[Write an Article](#)[Improve an](#)[Article](#)[Pick Topics to](#)[Write](#)[Share your](#)[Experiences](#)[Internships](#)