# Python GUI Programming with PAGE

*Create professional-looking GUIs for Python applications efficiently and effectively*

**Gregory Walters**

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

**UK | UAE | INDIA | SINGAPORE**

ISBN 978-93-55518-378

www.bpbonline.com

# Dedicated to

*My son Douglas*

# About the Author

**Gregory Walters** has been programming since 1972. Since 2009, he has been writing articles about programming in Python for Full Circle Magazine. To date, he has done well over 100 Python articles for them as well as book reviews and other things. In addition, he has been using PAGE since about the same time. He has also written a 4 chapter tutorial on using PAGE that has been included in each copy of the Page distribution since early 2019.

Greg has written one book on Python, co-authored one book on GANs in Python and done Technical Review on over 20 books for various publishers.

Greg is currently semi-retired and live in Central Texas, U.S.A. In his spare time, he runs the "Unofficial Page Support Page" on Discord, enjoys cooking, playing music and building his own musical instruments.

# Acknowledgements

There are a few people I want to thank for the continued and ongoing support they have given me during the writing of this book. First and foremost, I would like to thank my son for continuously encouraging me for writing the book, put up with me when things weren't going the way I expected — I could have never completed this book without his support.

I am grateful to Don Rozenberg the author of PAGE. Without his support, guidance and willingness to accept suggestions, I would never have been able to create this book.

I also want to thank Ronnie Tucker, the Editor in chief of Full Circle Magazine, who back in 2009, gave me my first major publication attribution. Without his constant backing and support over the years, I wouldn't be where I am today.

To my dear friend Halvard in Norway, who has for many years sent me kind words, gentle persuasion, and positive reinforcement on a constant basis in all that I've done.

My gratitude also goes to the team at BPB Publication for keeping me on track during the difficult process of pulling this book together.

# Preface

Learning PAGE is designed to take the reader from installation of the PAGE designer program all the way to creating complex Graphical User Interfaces for Python programs. PAGE has been around many years but unfortunately has not gotten much "press" in the media, so it is not very well known.

This book is designed to take the reader from installation of the Page designer through to being able to create complex GUI interfaces. These include front end interfaces for SQLite databases (also extends to other databases) to extending the GUI to use the Canvas widget to include drawing rectangles, ellipses, text and other things. Topics include using standard Tk widgets (buttons, static text, entry, etc.) to the ttk toolkit and expanding the capabilities of each. Also covered is using third party widget libraries and creating a custom widget and including it in real-world programs.

This book is divided into 12 chapters plus the Introduction. They will cover the basics of using PAGE, adding widgets to the form and changing the attributes that those widgets have and the process of generation of the Python modules. After that, the process of modifying the Python files to create the end program.

**Introduction –** will show the user how to install and more importantly how to properly start PAGE.

**Chapter 1: Introduction to PAGE –** will show the various windows that make up PAGE, how to create the user's first program using PAGE. The widgets used for this project will include Radiobuttons, Frames, dynamic and static Labels and more.

**Chapter 2: Going Further –** will expand the learning process to include two projects, one showing how to create a program that has multiple forms and how to create splash screens.

**Chapter 3: Standard Tk Widgets –** will cover the PAGE standard Tk library of widgets, and the attributes that each possess.

**Chapter 4: The Pinger Program –** will show how to create a real world program that uses the ping internet utility. This project shows how to control the Label colors dynamically and basics of using the Tk Text widget.

**Chapter 5: Using Graphics –** covers the use of graphical images in the reader's projects, including which widgets support the use of graphics and how to embed graphics into the project both by way of the designer and in code as well.

**Chapter 6: Menus and Popup Menus –** goes through the use of Menu bars, Button Bar menus and popup (or context) menus.

**Chapter 7: Using ttk Widgets –** similar to Chapter 3 which explored the all the standard Tk widgets, this chapter explores all the ttk group of widget that expand the widget set and provide the ability to create more modern looking programs.

**Chapter 8: Custom Controls –** explores the ability to use third party widgets, not normally supported by PAGE, the enhance the look, feel and usability of the reader's programs.

**Chapter 9: Creating a SQLite Database Front End –** covers the ability to use a PAGE project to act as a front end for advanced Database programs to display, enter and save information back to the database.

**Chapter 10: Creating Custom Profiles –** covers the Preferences editor to allow the user to control the look and feel of PAGE itself and the projects it creates.

**Chapter 11: Using the Canvas Widget –** shows how the Tk Canvas widget, which is one of the most complex and powerful widgets in the Tk toolkit, can be controlled to create graphs, text, lines, ovals and more.

**Chapter 12: Conclusion –** wraps up the book touching on getting support and further resources for the reader to expand their knowledge of Tkinter programs and PAGE.

## Code Bundle and Coloured Images

Please follow the link to download the
***Code Bundle*** and the ***Coloured Images*** of the book:

## https://rebrand.ly/a766f8

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Python-GUI-Programming-with-PAGE**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

## Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

**errata@bpbonline.com**

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

## Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

## If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

## Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

# Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

**https://discord.bpbonline.com**

# Table of Contents

# Introduction

PAGE is a free and open source that creates Graphical User Interfaces for Python programs. It allows a user to create these GUIs in a **Rapid Application Development (RAD)** mindset. PAGE uses Tcl/Tk to create the GUI and then saves the code in native Python/Tkinter files. It has been created and maintained by Don Rozenberg for many years.

PAGE enjoys over 2,000 downloads per month and is used worldwide. You do not need to understand Tcl/Tk to use PAGE. You do need to have a basic knowledge of Python, however.

I have been writing about and using PAGE since about 2011. Back then, PAGE was about the only thing that allowed Python users a quick and easy way to utilize the Tkinter interface easily, without having to be very knowledgeable about the Tk toolkit. There were many books and websites about WxPython/WxWindows and the Qt toolkit, but those had some rather complicated installation instructions and were not completely free and open source. Some confusing licensing restrictions caused many Python users to shy away from any type of GUI design. Tkinter comes with Python and is the de facto standard GUI package.

The current version of PAGE, as of this writing, is version 7.6.

# Why did I write this book?

For the majority of my programming career, I have been a mentor for other programmers in the various languages that I have used. I have been considered the kind of programmer that thinks "outside the box", finding unique solutions for difficult problems.

When it comes to PAGE, I have been using it for almost 10 years now and have enjoyed a close relationship with *Don Rozenberg*. I have been blessed to be one of the people whom Don relies on to test (and attempt to break) new versions of PAGE. Because of this, I have been able to learn a great deal about how PAGE works and how to avoid certain limitations that are not quite obvious to others.

Most of the projects that I will be presenting in this book are either projects that I have created for other users to address their questions about "How do I …" or projects that I have created for myself, either for my Full Circle articles or my own programs.

# What this book is

This book is designed to teach a PAGE user how to use the program to create some complex GUI frontend programs in Python.

It is designed to take the user from beginner to, by the end of the book, knowing how to use most of the Tkinter widgets at a normal programming level.

I have broken the book into 4 sections:

1. **Beginning:** This section is designed to get you started using PAGE from installation to simple programs using some of the widgets and concepts that you use most often.
2. **PAGE in depth:** This section will discuss the standard widgets from Tk and ttk and some real-world projects that include dynamically controlling widget attributes, menus and general use of graphics.
3. **Advanced Concepts:** This section will show how to use third-party widgets and create your own custom widgets to extend the abilities of PAGE.
4. **Summary:** Going through what has been learned, some tips and tricks, and some thoughts about best practices.

# What this book is NOT

This book is not designed to teach any given coding style. Coding styles are, in my opinion, a deeply personal subject and more times than not, will be dictated by the organization that you work for at the moment. My programming style is slanted to that of one that teaches the basic "how to", not necessarily the "best" or "only" way to do things. It is not going to show how to write everything in a single line of code. It uses a large number of comments, more for the teaching aspect than anything else.

This book is not designed to cover everything that can be done with PAGE or Tkinter. It is designed to cover, at a gross level, all the normal widgets that PAGE supports (which is most of them).

# What you need to know before reading this book

You should have a basic knowledge of Python. You DO NOT need to be a Python expert or a data scientist. You should have a basic knowledge of variables and their use and creating functions. Anything else will be helpful, but I have tried very hard to create and document reasonable projects that will provide you with everything you need to be able to, by the end of the book, create your own advanced GUI front ends for almost any program you want to create in Python.

# Text conventions used

There will be many different types of text that you should know about.

## Code blocks

When I am presenting blocks of code, I will use a Consolas font, like this:

```
try:
    import Tkinter as tk
except ImportError:
    import tkinter as tk
```

If I need to draw attention to a specific line of code within a block, I will make that line bold:

```
def on_btnExit():
    print('secondaryform_support.on_btnExit')
    sys.stdout.flush()
    destroy_window()
```

## Terminal commands

Whenever I need you to type a command in a Linux terminal or Windows command prompt, it will be in a bold consolas font. It will always be preceded by a "$" for Linux Terminals or ">" for Windows command prompts as follows:

```
$ tar zxf /path/to/ActiveTcl-download.tar.gz
```

or

```
> c:\page\winpage.bat
```

## Inline references

Sometimes, I will give filenames or information within a sentence. When this happens, I will again, use a bold monospace font within the sentence. Like this:

"Now, save the file as **`myfirstpage.tcl`** to your working directory."

## Widget attributes

All the projects we will be creating will use widgets or controls and there will be various settings or attributes that will need to be modified. To make it easy for you, I will use a table format to show the attribute name in the left column and the value of the attribute in the right. Something like this…

| alias | btnExit |
|---|---|
| command | on_btnExit |
| text | Exit |

## Notes, tips and warnings

Whenever I need to provide you with some special information, I will place the text within a box.

*Note: When entering the callback function for the* `root.after` *command, be sure NOT to put* `parentheses` *at the end of the function name as you normally would. Be sure to format it as follows:*

```
Timer_id = root.after(0, on_tick)
```

# Requirements

## Windows

Under Windows, you will need, of course, Python 3.8 or greater. I will be using Python 3.8.10. Earlier versions of PAGE required you to have Tcl/Tk installed on your system, but as of PAGE 6.0, this is no longer required. So, the only actual requirement for PAGE is Python 3.8 or greater.

You will also need some sort of an editor to keep up with your Python code. There is a very nice free cross-platform **Integrated Development Environment (IDE)** called Geany. I will be using VS Code, which is also free.

## Linux

While Linux usually has everything you need, as with Windows, you need Python 3.8 or greater.

I also use (as with Windows) a very nice free IDE called Geany and VS Code.

## Raspberry Pi (Linux)

You will need to use the latest Raspberry Pi OS. This should include Python 3.8 or greater, so you should be ready to go.

## OSX

Since I do not have access to any OSX machines, I really cannot speak about this from direct experience, but I understand from Don that the requirements for OSX are the same as Linux.

# Installation

## Windows

Download the latest version of PAGE from **https://sourceforge.net/projects/page/** and run the file. This will install PAGE.

Finally, create a "master" directory on your hard drive to hold your files.

## Linux

Installation under Linux is just as simple as downloading the PAGE package, extracting it and creating an alias for PAGE.

Download the PAGE distribution at **https://sourceforge.net/projects/page/.** The page normally detects your OS, so it should prompt you to download the tar-gzipped latest version.

Unpack the downloaded file into your home directory. Usually, the File manager will allow you to do this by right clicking on the file and selecting "**`Open with Archive Manager`**".

Remove any "**`.pagerc`**" files in your root folder.

If you are running a Linux distribution based on Debian, like Ubuntu or Linux Mint, edit your **`~/.bashrc`** file to include an alias to your PAGE file. Look for a line that says **`"# some more ls aliases"`**. I keep my PAGE folder under the Downloads folder, so I would enter:

```
alias page='python /home/greg/Downloads/Page-7.6/page/page.py'
```

Be sure to save the file. Finally, either close that terminal or within the terminal type:

```
$ source ~/.bashrc
```

This forces the terminal to reload the resource file. Now, whenever you want to start PAGE, all you have to do is change to your project folder and type "page".

## Raspberry Pi (Linux)

Make sure that you have the latest Raspberry Pi OS build. Once you have your Raspberry Pi running, you can follow the given steps:

1. Download the PAGE distribution from **https://sourceforge.net/projects/page/.**

2. Once it is downloaded, untar the downloaded file into a convenient directory.
3. Open a terminal window and navigate to that folder.
4. Finally, edit your ~/.bashrc file to add an alias to create an easy command that will start PAGE from your terminal:

```
$ mousepad ~/.bashrc
```

look for a line that says **"# some more ls aliases"**

5. Below this line, add a blank line and type:

```
alias page='python /home/greg/Downloads/Page-7.6/page/page.py'
```

6. Save the file. Now, you have a choice. You can close and reopen your terminal or you can force the terminal to reload the **bashrc** file.

```
source ~/.bashrc
```

You are done with the installation.

## OSX

Again, from what Don tells me, the steps for installation are pretty much the same as Linux.

I will leave the installation of Geany or whatever IDE you choose to use to you.

As under Windows and Linux, create a "master directory" to hold your development files.

# Starting PAGE

One of the frequently asked questions is about having problems embedding graphics in various widgets. It works fine in the designer, but when the user tries to run the generated program, they receive an error message about missing graphics. Almost every time, this is due to not starting PAGE correctly; in whichever operating system you are using. It is **VERY** important to start PAGE directly from the folder that you plan to save your project into.

This is why I always suggest you create a development folder, whether for learning or actual work. For my own work, I have numerous folders for this. One I call **POC** for **P**roof **O**f **C**oncept programs. It is a kind of a sandbox for ideas that I have no idea if it will actually work or not. If things work out correctly and I like the results, I will move the working folder into my **Development** folder. This is where I keep any projects that I expect to get to at least a Beta state. At that point, I either move or copy to my local Github project folder so I can have source code control.

Using a Windows system for a quick example, I put my development folder directly in the **`C:\`** folder. (This is where PAGE installs all its files.) So, my directory tree would look something like this:

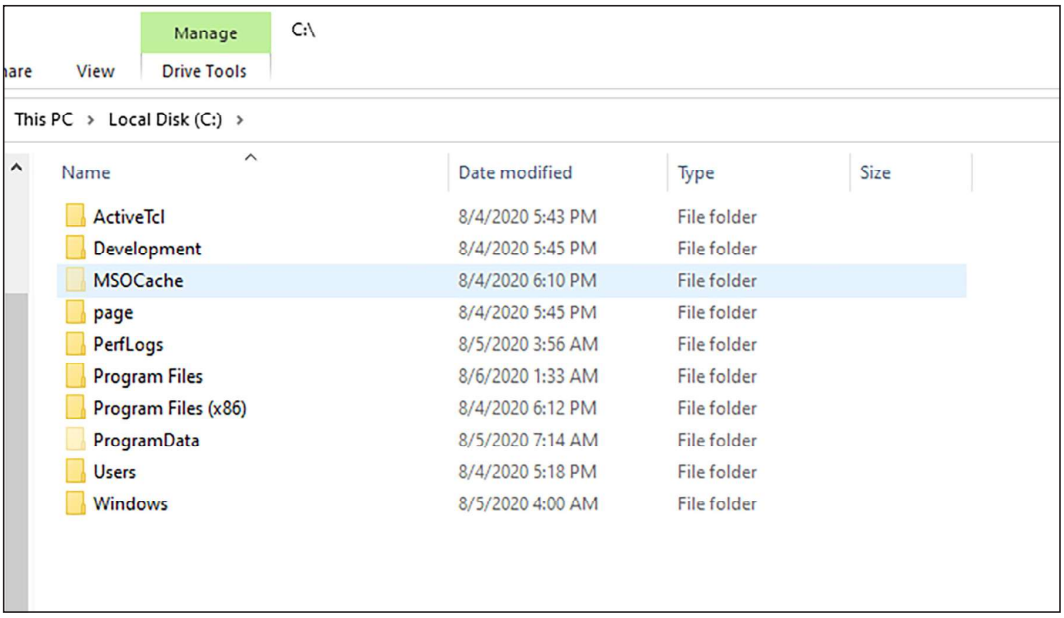


***Figure: Windows** Directory Tree*

Under my Linux machines, I usually keep my Development folder directly under the **`/home/greg`** folder. Under the Development folder are all my various projects in their own folders.

Under Linux, it is simple to get to my Development folder by using the Nemo (under Linux Mint or Nautilus under Ubuntu) GUI file manager to navigate directly to the folder that I want to work in, then right click within that folder to open a new terminal. Once I am in the terminal, to get PAGE running, I simply type:

```
$ page
```

Under Windows, it is not quite that easy, but not too much different. Using the Windows GUI File Manager, navigate to the folder you want to work in. Next, type {Ctrl} + L. This will allow you to type within the address bar, which normally shows the path. Now, simply type "**`cmd`**" and press enter. This will open a new command box. Now, in the new command prompt box, type either:

```
c:\page\winpage.bat
```

or if the PAGE folder has been added to your path, just type **"`winpage`"**.

# Getting support

When you are beginning to work with a new program, there will be times that you will run into issues. It might be something you do not understand or maybe a potential bug. Sometimes, it might just be that you have an idea but do not know how to actually implement it in the program.

PAGE offers several options to get some of the answers. Here are just some of the ways:

- Provided documentation: PAGE comes with extensive documentation that is provided with each copy of the documentation. It is located in the PAGE installation folder under the **docs** folder. In this folder, there is an epub version as well as an html version. There is also a 4-chapter tutorial located there.
- Provided examples: Don has provided numerous example programs to help you along on many different subjects. This is located again in the same folder you installed PAGE under the examples folder. If you need examples for Python 2.x, there is the examples-py2 folder.
- Official support: If you cannot find the answer to your questions in the documentation or in the examples, you can always go to the official PAGE Discussion Forum. There is a good chance that someone has already asked the same question that you have. If not, you can always ask your question there. It might take a day or two to get an answer, but you will get an answer. The forum is located at **https://sourceforge.net/p/page/discussion/**.

# Last thoughts

One last thing about IDE programs. They are not created equal. While there are dozens out there, some have hidden "options" that might make your job harder rather than easier. While this is strictly a personal opinion, I would suggest not using IDLE as your editor of choice. There is a "feature" that starts IDLE with the -i (idle -i) to enter the interactive mode after the program is finished. This leaves the UI on the screen and might cause some concern (Thanks Halvard for pointing this out to me). Again, your choice of an IDE is YOUR choice. I like Geany, as it is a very simple but powerful editor or VS Code which allows you to debug your code by single stepping through selectable areas. Many programmers prefer other editors like Sublime Text while others like IDLE. The bottom line is to use whatever editor you feel productive using.

Now, you should have gotten PAGE installed on your system and created a working folder to hold all the code you will be creating. So, let us get started!

## Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

**https://discord.bpbonline.com**

# Chapter 1
# Introduction to PAGE

## Introduction

It's now time to start learning PAGE. This chapter will concentrate on the basics of using PAGE and creating your own GUI front ends for your programs. Remember, the whole point of PAGE is to allow you to rapidly design and modify graphical interfaces.

The process of creating your GUI is to place widgets, or controls, onto a blank designer form. With PAGE, we can place those widgets by simply clicking on the widget you desire in the Widget Toolbar and clicking on the designer form somewhere close to the position you want it to live. You can move widgets by simply dragging them around the form or using the keyboard arrow keys to "fine-tune" their position.

## What you will learn

Following are the topics that you will learn in this topic:

- Starting PAGE within your Operating System
- The major functions each window provides
- Basic use of the Label widget
- Basic use of the Button widget
- Basic use of the Frame widget

- Basic use of the Radiobutton widget
- Creating a GUI form
- Saving a PAGE project

As I said in the Introduction, one important step to a successful project is the way you start PAGE.

# Starting page

This section will show the best ways to start PAGE for your operating system and a quick overview of all of the windows that make up the PAGE program.

If you haven't already done so, create a folder to hold your source code and assets (images, databases, and so on.).

If you are running Linux, MacOS, or Raspberry Pi, use the GUI File Manager, navigate to your development folder, and use the right mouse button to get the context menu. Select "**`Open in Terminal`**". Once the terminal is open, assuming you set up your terminal resource file, type "**`page`**".

If you are running Windows, use File Explorer to navigate to your project folder. Once there, press *{Ctrl}-L* to edit the address bar and type "**`cmd`**". This will open a new command prompt and type "**`winpage.bat`**" or "**`c:\page\winpage.bat`**".

No matter which operating system you are running, you should be presented with 5 different Windows arranged around your screen. It should look something like the one shown in *Figure 1.1*:

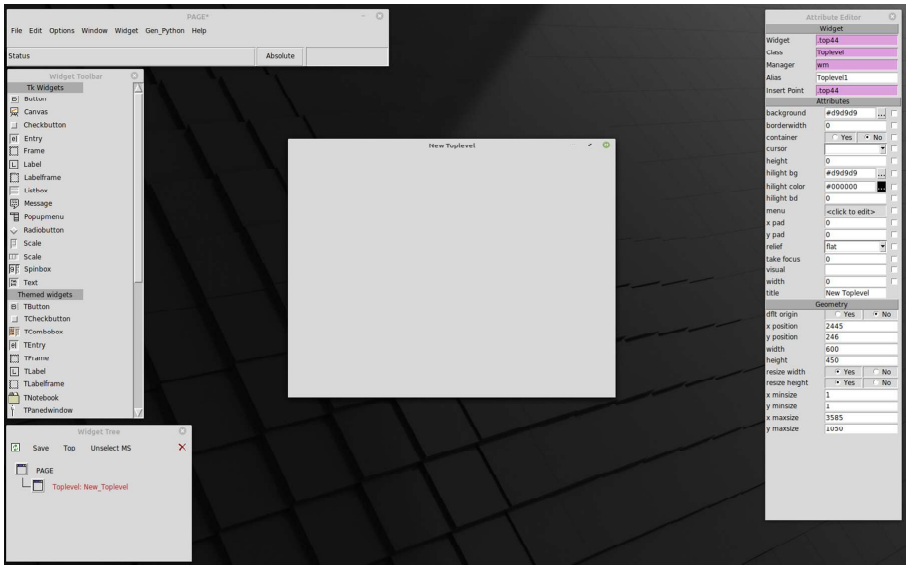


***Figure 1.1**: The Page Windows*

# The PAGE designer forms

With 5 different windows taking up the majority of your screen, it's hard to know where to start the first time you run PAGE. Let's take a moment to take a look at each window to see what it is for.

## The Main Window

The main window as shown in *Figure 1.2* is important for many reasons. It contains the commands to save your project and generate your Python code. There are many other options that you will occasionally use, but for the most part, you will use this window the least out of all the other windows as shown in *Figure 1.2*:

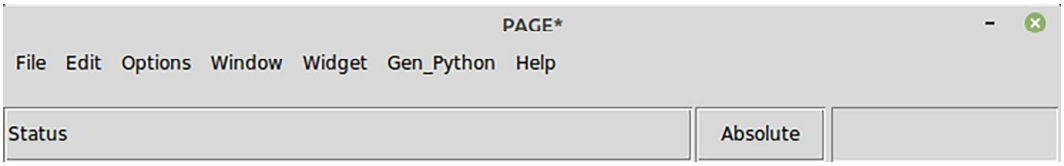


***Figure 1.2**: The Main Window*

At the very top left of the window, you will find the menu. At the bottom left, you will find the status box, which shows things that PAGE wants you to know. Near the right-hand side, again on the bottom, is a button that will say either "Absolute" or "Relative". You can click on it to change the placement mode. The Absolute mode will use an exact X/Y position for the placement of your widgets. In the Relative mode, a more general method for the placement is used. The big difference is that if you select the Relative mode when the designed form is resized, the widgets will resize along with the form and the positions of the widgets will move in relationship with the size. In the Absolute mode, the size and position of all the widgets remain the same, no matter how the user resizes the form.

When I design a GUI, 99% of the time I use the Absolute mode, yet Don, the author of PAGE uses the Relative mode all the time.

We'll explore some of the other menu options as we go through the next few chapters.

## The Attribute Editor

The Attribute Editor shown in *Figure 1.3* is one of the windows that you will spend a great deal of time using. This is where you set the various attributes of each widget. It is divided into three sections: Widget general attributes and Attributes section is where you can set specific information for things like the text displayed on the widget or a command callback function. The final section is the Geometry section

where you can set the width/height and the X/Y position as well as a few other things:

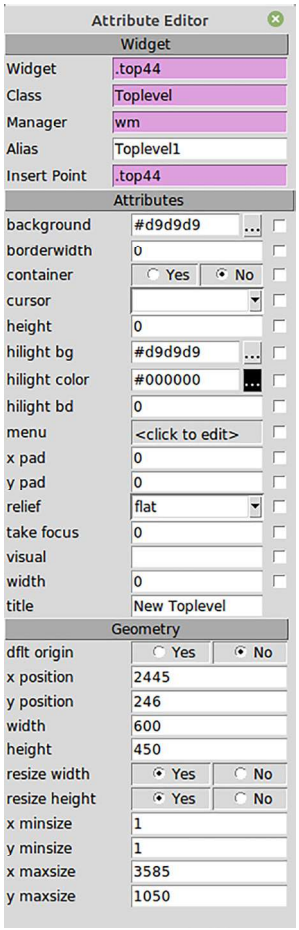


***Figure 1.3**: The Attribute Editor*

This is one of the windows where you will spend a great deal of time using. This is where you set the various attributes of each widget. It is divided into three sections, Widget general attributes, Attributes where you set specific information for things like the text displayed on the widget or a command callback function. The final section is the Geometry section where you set the width/height and the X/Y position as well as a few other things.

If one or more of the sections don't show the proper attributes, simply click on the gray bar for that section and it will expand out so that you can see all the attributes.

The Attributes section changes, depending on the specific widget you have selected. For example, some of the widgets don't have a command callback option or attribute, so it will not show for those widgets.

We'll discuss various widget attributes later in this chapter and in *Chapter 4, Standard Widgets* as well as the various projects as we go forward.

# The Widget Tree

Continuing around the screen in a clockwise manner, we get to the widget tree as shown in *Figure 1.4*:


***Figure 1.4**: The Widget Tree*

Every widget that you add to the form will show up here as well as on the main form itself. This provides a quick and easy way to select a widget without the risk of moving the widget, which can be a concern in very complex forms that have several widgets. We'll look at this more throughout the other chapters.

# The Widget toolbar

The Widget toolbar shown in *Figure 1.5* will be one of the windows that you will spend a lot of time with. This scrollable window holds all the widgets that you can use to create your GUI form:

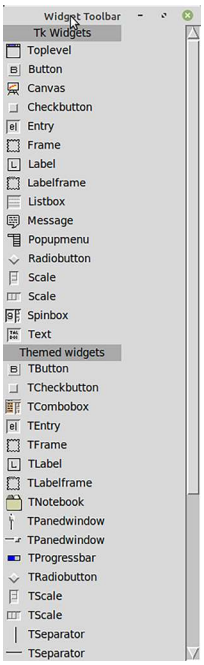


***Figure 1.5:** The Widget Toolbar*

Think of it as an artist's palette of colors. When you want to put a widget on your main form, simply click on the widget to select it (the status area of the main window will acknowledge your selection by changing to something like "Insert Button"), then move the mouse button to the general location you wish to place it on the designer form and click on the form with your left mouse button. That's it. You don't even need to drag and drop it. Click once on the toolbar then click again on the designer form.

I will probably be referring to this window as the "toolbox" from here onwards.

## The Designer form

If the toolbox is the artist's palette, the Designer form is the canvas. This is not to be confused with the canvas widget. It is officially called the **TopLevel** widget (Refer to *Figure 1.6*). The designer form is blank when you start PAGE:


***Figure 1.6**: The Designer Form*

That confuses many people, but that's what it is really called. It is actually a widget, just like a button or static text label. This holds all the other widgets that make up our GUI. At the top of the form is the title. You can modify this using the Attribute Editor or you can leave it with the default and change it in code at any time. We'll see that later on. It also has an alias by default of "Toplevel1". Again, you can change this or leave it with the default.