GUI Programming with Python

Submitted by NanoDano (/users/nanodano) on Sat, 12/05/2015 - 16:17

Overview

Intro

There are many options for GUI programming with Python but the easiest one to use is Tk because it comes packaged with Python and is cross platform. It used to be ugly and had a limited set of widgets, but with version 8.5+ it integrates better with the native system and has more widgets to choose from.

This is aimed at developers who are already familiar with Python but not familiar with GUI programming. I have two goals here. One is to provide examples and demonstrate that GUI programming is not as painful as most developers would expect. The other is to provide a high level view of the widgets that are available for you to use.

All examples are in Python 3. The Windows and Mac OS X distributions come with Tkinter by default, but some Linux distributions may require separate installation. Refer to your distribution documentation for install Tkinter if necessary. Each example is a standalone program that can be run. You can actually copy/paste them right in to the interactive Python3 interpreter to try them out (except for the couple that require a command line argument to pass an image path).

Search

Tags

- Python (/tags /python)

Mailing List

Get informed about new posts and changes to the site.

* indicates required

Email Address *

Name

Subscribe

Related Posts

- Python Tutorial - Functions (/content/python-tutorial-functions)
- Python Tutorial - User Input and Output (/content /python-tutorial-user-input-and-output)
- Python Tutorial - Starting Your Cookbook (/content/python-tutorial-starting-your-cookbook)
- PyCharm Tips (/content /pycharm-tips)
- Working with Spreadsheets in Python (/content /working-spreadsheets-python)
- Text-To-Speech with Python

BACK TO TOP

Important Things to Learn

- There is one main parent window. Widgets and other windows are children of the main window.
- What widgets are available (e.g. Label, Button, Progressbar)
- What options are available on widgets (e.g. border, font, cursor, background, text, command)
- How to pack widgets in to a window

Widgets

Widgets are the building blocks of a GUI program. They are used to display information or get input from the user.

Frame	Used as a container to house other widgets and add borders
Label	Display text or images
Labelframe	A frame that by default displays a border and title
Button	Standard button that calls a function when clicked
Checkbutton	Check box for toggling a value (can have callback on click)
Radiobutton	Standard radio buttons
Entry	Single line editable text entry
Text	Multiple line editable text entry
Message	Multiple line display text that can be styled
Combobox	A single line text entry field that also has a drop down select list. Choose pre-defined items or enter a custom value.
Listbox	Multiple line select list
Scrollbar	Vertical or horizontal
Sizegrip	For adding a triangle in the bottom right corner of a window to make it easier to grab and resize
Progressbar	Progress bars that can be customized in many ways
Scale	Sliding scale to be clicked and dragged with mouse
Spinbox	Like a single line text entry widget with up and down arrows on the side to increment or decrement. Mousewheel up and down to raise and lower values
Separator	Visually separate widgets vertically or horizontally
Notebook	Tabbed contents
Panedwindow	Like a frame that contains a horizontal or vertical set of frames that are resizeable in relation to each other
Canvas	For drawing graphics like lines, circles, arcs, ovals, and rectangles
Toplevel	A window just like the main window with its own title bar and can be moved and resized on its own. If the main window is destroyed, all other top levels are destroyed too
Menu	Menu bar for the top of the window. Typically things like "File", "Edit", and "View".
Optionmenu	Similar to Combobox or HTML select. A dropdown option select menu.

Pop-up Dialog Windows

askquestion	Ask a question with a yes or no response.
askyesno	Ask yes or no.
askyesnocancel	Ask yes, no, or cancel.
askokcancel	Ask ok or cancel.
askretrycancel	Ask retry or cancel
showinfo	Show pop-up with text and info icon.
showwarning	Show pop-up with text and warning icon.
showerror	Show pop-up with text and error icon.
Colorchooser	Visual color picker
Filedialog	Allow user to pick a file from the file system

Espeak (/content/text-speech-python-espeak)
- Installing PyCrypto/Paramiko for Python3 x64 Windows (/content/installing-pycryptoparamiko-python3-x64-windows)
- Fix Pip Install Unable to Find vcvarsall.bat (/content/fix-pip-install-unable-find-vcvarsallbat)

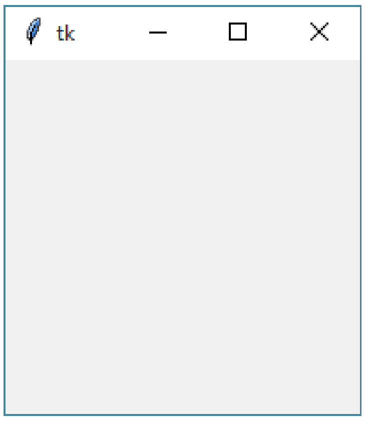
BACK TO TOP

Check TkVersion

```
# check_tk_version.py
import tkinter
print(tkinter.TkVersion)
```

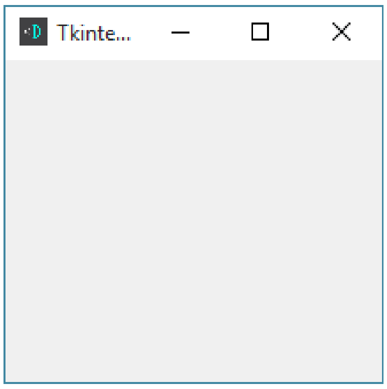
Minimal Program

```
# minimal_tkinter.py
from tkinter import Tk
root = Tk()
root.mainloop()
```



Set Icon and Title of Window

```
# set_icon_and_title.py
import sys
from tkinter import Tk, PhotoImage
root = Tk()

# Image path provided as first command line arg. PNG format
root.iconphoto(root, PhotoImage(file=sys.argv[1]))
root.title("Tkinter Test")

root.mainloop()
```



Get Screen Size

BACK TO TOP

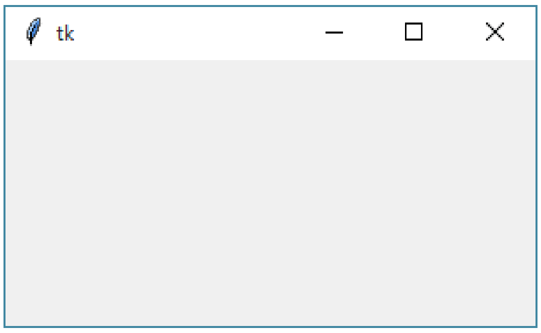
```
# get_screen_size.py
from tkinter import Tk
root = Tk()

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

print("Screen width:", screen_width)
print("Screen height:", screen_height)
```

Make Window Full Screen

```
# make_fullscreen_window.py
from tkinter import Tk
root = Tk()

root.attributes('-fullscreen', True)

root.mainloop()
```

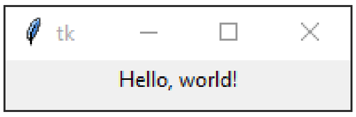
Set Size and Location of Window

```
# set_window_geometry.py
from tkinter import Tk
root = Tk()

# Make window 300x150 and place at position (50,50)
root.geometry("300x150+50+50")

root.mainloop()
```



Displaying Text

```
# display_text.py
from tkinter import Tk, Label
root = Tk()

# Create a label as a child of root window
my_text = Label(root, text='Hello, world!')
my_text.pack()

root.mainloop()
```

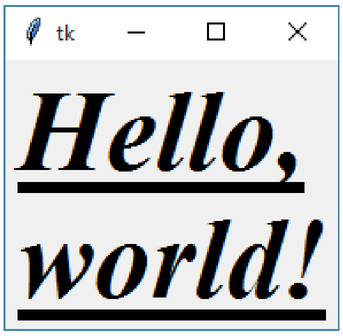


Hello, world!

BACK TO TOP

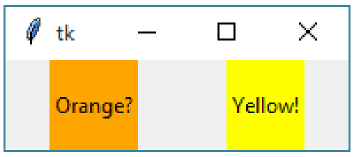
Change Fonts

```
# change_font.py
from tkinter import Tk, Message
root = Tk()

msg = Message(root, text='Hello, world!')

# Font is a tuple of (font_family, size_in_points, style_modifier_string)
msg.config(font=('times', 48, 'italic bold underline'))

msg.pack()

root.mainloop()
```



Packing Options

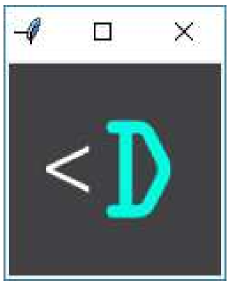
```
# pack_options.py
from tkinter import Tk, Label, Y, RIGHT
root = Tk()

label1 = Label(root, text='Yellow!', background='yellow')
label2 = Label(root, text='Orange?', background='orange')

# Some of the packing options:
# - fill: tells the widget to expand to take up any extra space (X, Y, or BOTH)
# - padx/pady: outter padding
# - ipadx/ipady: inner padding
# - side: which side to stack from. Default is TOP (to bottom)

label1.pack(fill=Y, padx=25, ipady=15, side=RIGHT)  # Pack from right to left
label2.pack(fill=Y, padx=25, ipady=15, side=RIGHT)

root.mainloop()
```



Displaying an Image

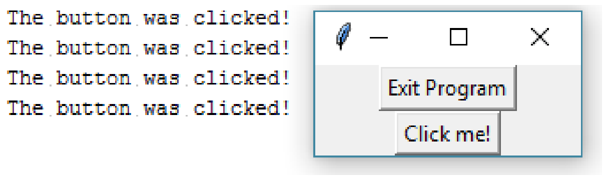
BACK TO TOP

```
# display_image.py
import sys
from tkinter import Tk, Label, PhotoImage
root = Tk()

# First command line arg is image path. PNG format
img = PhotoImage(file=sys.argv[1])
my_image = Label(root, image=img)
my_image.pack()

root.mainloop()
```



Buttons with Callbacks

```
# button_callback.py
from tkinter import Tk, Button
root = Tk()

# Create a button that will destroy the main window when clicked
exit_button = Button(root, text='Exit Program', command=root.destroy)
exit_button.pack()

# Create a button with a custom callback
def my_callback():
    print("The button was clicked!")  # Prints to console not the GUI

print_button = Button(root, text='Click me!', command=my_callback)
print_button.pack()

root.mainloop()
```

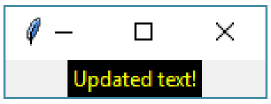


```
The button was clicked!
The button was clicked!
The button was clicked!
The button was clicked!
```

Modify Existing Widget

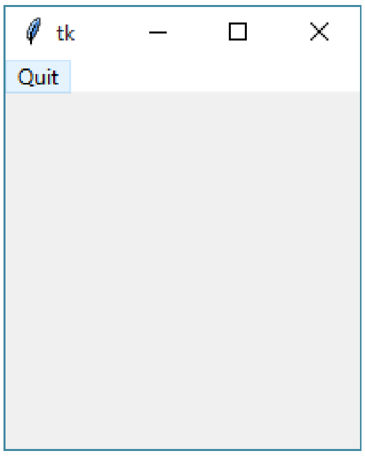
BACK TO TOP

```
# modify_widget.py
from tkinter import Tk, Label
root = Tk()

# Create a plain label that says "Hello, world!"
label = Label(root, text='Hello, world!')
label.pack()

# Modify a widget with config(). Widgets can be modified after packing.
label.config(foreground='yellow', background='black', text='Updated text!')

root.mainloop()
```



Menu Bar

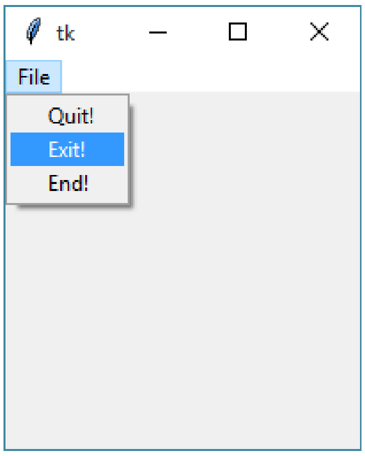
```
# menu_bar.py
from tkinter import Tk, Menu
root = Tk()

# Create a main menu and add a command to it
main_menu = Menu(root, tearoff=0)
main_menu.add_command(label="Quit", command=root.destroy)

root.config(menu=main_menu)
root.mainloop()
```



Menu Bar with Dropdown

BACK TO TOP

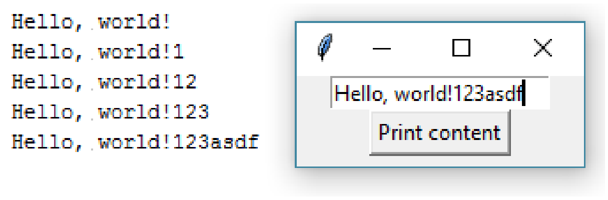
```
# menu_bar_dropdown.py
from tkinter import Tk, Menu
root = Tk()

# Create main menu bar
menu_bar = Menu(root)

# Create the submenu (tearoff is if menu can pop out)
file_menu = Menu(menu_bar, tearoff=0)

# Add commands to submenu
file_menu.add_command(label="Quit!", command=root.destroy)
file_menu.add_command(label="Exit!", command=root.destroy)
file_menu.add_command(label="End!", command=root.destroy)

# Add the "File" drop down sub-menu in the main menu bar
menu_bar.add_cascade(label="File", menu=file_menu)

root.config(menu=menu_bar)
root.mainloop()
```



Input and Output with Entry Widget

BACK TO TOP

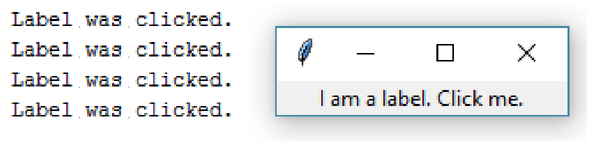
```
# entry_widget.py
from tkinter import Tk, Entry, Button, INSERT
root = Tk()

# Create single line text entry box
entry = Entry(root)
entry.pack()

# Specifying character position in entry
# - END: After last character of entry widget
# - ANCHOR: The beginning of the current selection
# - INSERT: Current text cursor position
# - "@x": Mouse coordinates

# Insert some default text
entry.insert(INSERT, 'Hello, world!')

# Print the contents of entry widget to console
def print_content():
    print(entry.get())

# Create a button that will print the contents of the entry
button = Button(root, text='Print content', command=print_content)
button.pack()

root.mainloop()
```



```
Hello, world!
Hello, world!1
Hello, world!12
Hello, world!123
Hello, world!123asdf
```

Bind Mouse Click to Label

```
# label_click_event.py
from tkinter import Tk, Label
root = Tk()

# Labels do not have a command option like buttons
# but you can manually attach the click event to a callback
label = Label(root, text='I am a label. Click me.')
label.pack()

def my_callback():
    print('Label was clicked.')

# Bind mouse button 1 click on label
label.bind("<Button-1>", lambda e:my_callback())

# A lambda with parameters looks like this:
# lambda event, a=12, b=34:my_callback(a, b))

root.mainloop()
```

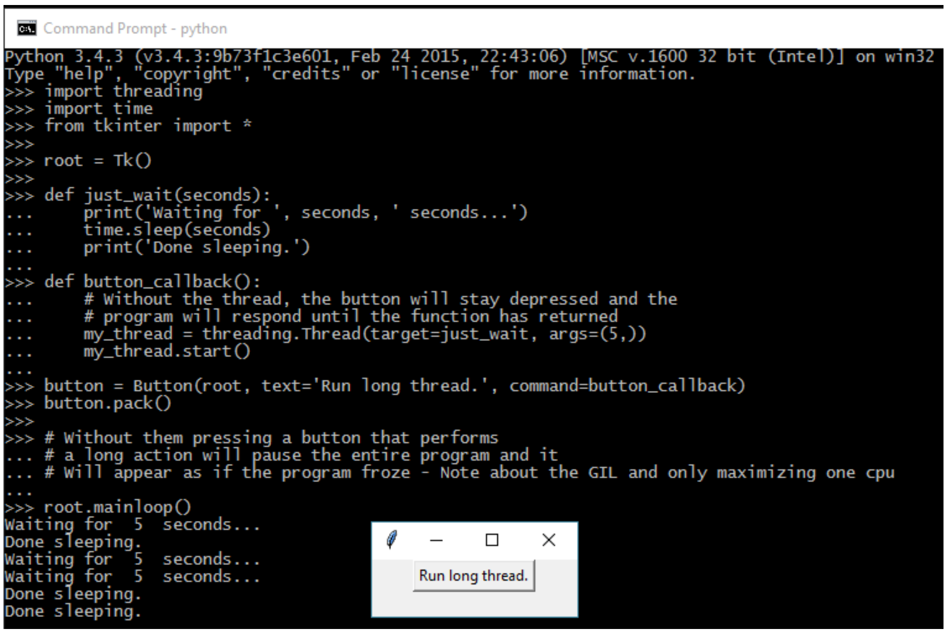


```
Label was clicked.
Label was clicked.
Label was clicked.
Label was clicked.
```

BACK TO TOP

Use Threads to Perform Background Operations

```
# threading_example.py
import threading
import time
from tkinter import Tk, Button

root = Tk()

def just_wait(seconds):
    print('Waiting for ', seconds, ' seconds...')
    time.sleep(seconds)
    print('Done sleeping.')

def button_callback():
    # Without the thread, the button will stay depressed and the
    # program will respond until the function has returned
    my_thread = threading.Thread(target=just_wait, args=(5,))
    my_thread.start()

button = Button(root, text='Run long thread.', command=button_callback)
button.pack()

# Without them pressing a button that performs
# a long action will pause the entire program and it
# Will appear as if the program froze - Note about the GIL and only maximizing on
e cpu

root.mainloop()
```

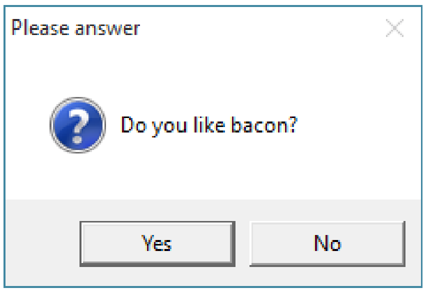


```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import threading
>>> import time
>>> from tkinter import *
>>>
>>> root = Tk()
>>>
>>> def just_wait(seconds):
...     print('Waiting for ', seconds, ' seconds...')
...     time.sleep(seconds)
...     print('Done sleeping.')
...
>>> def button_callback():
...     # Without the thread, the button will stay depressed and the
...     # program will respond until the function has returned
...     my_thread = threading.Thread(target=just_wait, args=(5,))
...     my_thread.start()
...
>>> button = Button(root, text='Run long thread.', command=button_callback)
>>> button.pack()
>>>
>>> # Without them pressing a button that performs
... # a long action will pause the entire program and it
... # Will appear as if the program froze - Note about the GIL and only maximizing one cpu
...
>>> root.mainloop()
Waiting for  5  seconds...
Done sleeping.
Waiting for  5  seconds...
Waiting for  5  seconds...
Done sleeping.
Done sleeping.
```

Pop-up Dialog

BACK TO TOP

```
# ask_yes_no.py
from tkinter import messagebox

dialog_title = 'Please answer'
dialog_text = 'Do you like bacon?'
answer = messagebox.askquestion(dialog_title, dialog_text)

if answer == 'yes':
    print('I like bacon too!')
else:  # 'no'
    print('You must have clicked the wrong button by accident.')
```



Multiple Top Level Windows

```
# create_destroy_toplevel.py
from tkinter import Tk, Toplevel, Button
root = Tk()

# Create new top level window. Opens immediately
second_window = Toplevel()
second_window.title('Second window')

# Destroy window
def destroy_second_window():
    second_window.destroy()

close_button = Button(
    root,
    text='Close second window',
    command=destroy_second_window
)
close_button.pack()

root.mainloop()
```

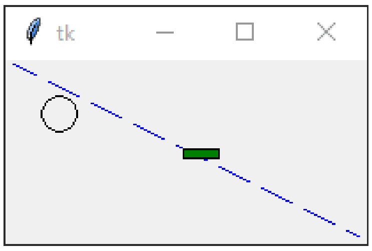
Focus Different Toplevel Windows

```
# focus_toplevel.py
from tkinter import Tk, Toplevel
root = Tk()

second_window = Toplevel()
second_window.title('Second window')

# Focus on window
second_window.focus_force()

root.mainloop()
```

BACK TO TOP

Drawing with the Canvas

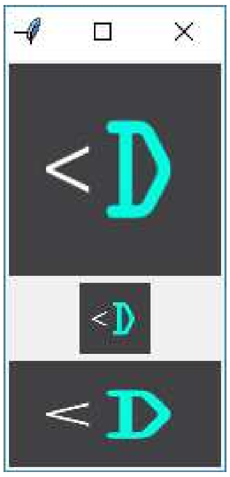
```
# canvas_draw.py
from tkinter import Tk, Canvas
root = Tk()

canv = Canvas(root, width=200, height=100)
canv.pack()

# Draw blue line from top left to bottom right with wide dashes
canv.create_line(0, 0, 200, 100, fill='blue', dash=(5, 15))

# Draw green rectangle at (100,50) to (120,55)
canv.create_rectangle(100, 50, 120, 55, fill='green')

# Draw oval(circle) from (20,20) to (40,40)
canv.create_oval(20, 20, 40, 40)

root.mainloop()
```



Shrink an Image

```
# shrink_image.py
import sys
from tkinter import Tk, PhotoImage, Label
root = Tk()

# First command line arg is image path. PNG format
base_img = PhotoImage(file=sys.argv[1])

# Take only every 3rd pixel from x and y resulting in an image of 1/3 size
img1 = base_img.subsample(3, 3)

# Take every pixel from x but only every other from y. Shrinks y by half
img2 = base_img.subsample(1, 2)

base_img_label = Label(root, image=base_img)
img1_label = Label(root, image=img1)
img2_label = Label(root, image=img2)

base_img_label.pack()
img1_label.pack()
img2_label.pack()

root.mainloop()
```

BACK TO TOP



Get and Set Pixels of an Image

```
# get_set_image_pixels.py
import sys
from tkinter import Tk, PhotoImage, Label
root = Tk()

# First command line arg is image path. PNG format
img = PhotoImage(file=sys.argv[1])

red = 255
green = 0
blue = 0
x = 0
y = 0

# Sets the top left pixel of the image to red (difficult to see)
img.put("#%02x%02x%02x" % (red, green, blue), (y, x))

# Get the (red, green, blue) tuple of pixel at 0,0
top_left_pixel = img.get(0, 0)
print(top_left_pixel)

# Display the image
img_label = Label(root, image=img)
img_label.pack()

root.mainloop()
```

Further Reading

- http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html (http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html)
- http://effbot.org/tkinterbook/ (http://effbot.org/tkinterbook/)
- http://www.tkdocs.com/ (http://www.tkdocs.com/)

BACK TO TOP

11 Comments **Dev Dungeon** 1 **Login**

Recommend 10 Share Sort by Best

Join the discussion…

Helder • a year ago

Very nice intro tutorial! Thanks!

2 • Reply • Share ›

Martín Herrera • a year ago

Great post! Are there any limitations with Python 2.7 version? Talking about Tkinter GUI of course...

1 • Reply • Share ›

devdungeon Mod → Martín Herrera • a year ago

Not that I can think of. Some of the modules were renamed but they both rely on the underlying tk library so they should be equal.

• Reply • Share ›

Alexander Kuptsov • a year ago

Does anyone use it still?

1 • Reply • Share ›

devdungeon Mod → Alexander Kuptsov • a year ago

Sure. I do.

1 • Reply • Share ›

Alexander Kuptsov → devdungeon • a year ago

Cool) I'm asking because I was thinking of learning a GUI under Python but I read that Tkinter is a waste of time and PyQt is better. I really don't know if that's true but PyQt seems kinda hard to install for a noob like me. Anyway, thanks for the reply :)

1 • Reply • Share ›

devdungeon Mod → Alexander Kuptsov • a year ago

That is the same reason I choose tkinter over qt. It is a hassle and difficult if you work on many different platforms. Qt has an open source license but requires a lot of money if you want to use it commercially.

Although if you are making a commercial product, Qt is superior with more widgets and better look. For most of my use cases, Tkinter has the widgets I need and it is much easier to load up on Windows/Mac/Linux versus trying to get Qt and PyQt with the right version installed on each machine.

BACK TO TOP

About DevDungeon

DevDungeon.com is a virtual hackerspace community with resources for developers and hackers.

Other

- RSS Feed (/rss.xml)
- DevDungeon GitHub Projects (http://www.github.com/DevDungeon)
- @thedevdungeon (http://www.twitter.com/thedevdungeon)
- Privacy Policy (/content/privacy-policy)
- XML Sitemap (/sitemap.xml)
- Brendan Polk Design (http://www.brendanpolk.com)

Contact

- Email: nanodano@devdungeon.com (mailto:nanodano@devdungeon.com)
- XMPP: nanodano@xmpp.devdungeon.com (xmpp:nanodano@xmpp.devdungeon.com)

NanoDano

★★★★★

Expertise

Python (https://www.codementor.io/python-experts)
PHP (https://www.codementor.io/php-experts)
Web application security (https://www.codementor.io/web-application-security-experts)
...

BOOK 1:1 SESSION (HTTPS://WWW.CODEMENTOR.IO/NANODANO?UTM_CAMPAIGN=PROFILE UTM_SOURCE=BADGE-NANODANO& UTM_MEDIUM=LIGHT)

BACK TO TOP