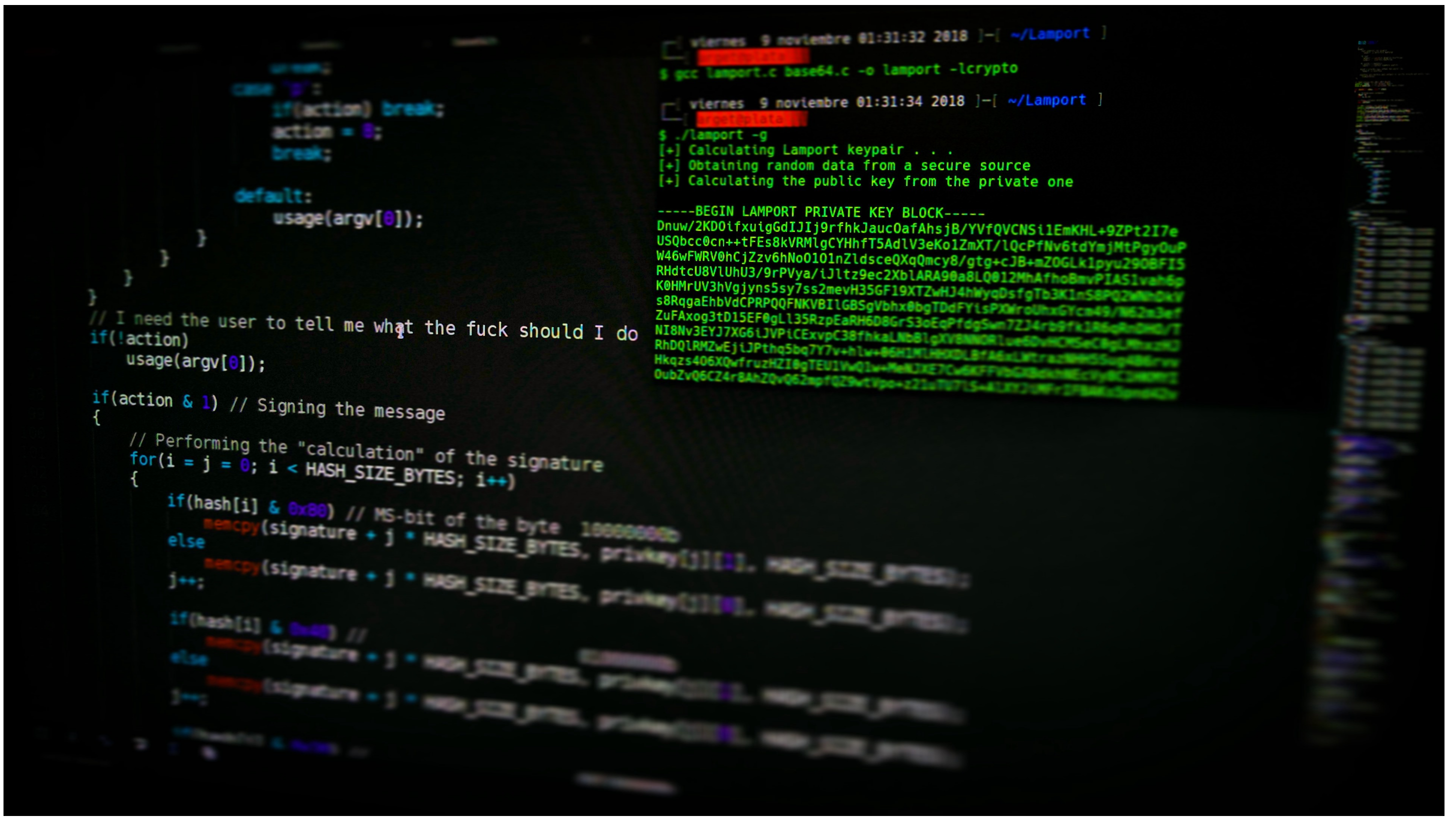
Learning to Use Progress Bars in Python

Introduction to 4 different libraries (Command Line & UI)

Costas Andreou
Dec 27 · 4 min read ★



```
    default:
        usage(argv[0]);
      }
    }
  }
}
// I need the user to tell me what the fuck should I do
if(!action)
    usage(argv[0]);

if(action & 1) // Signing the message
{
    // Performing the "calculation" of the signature
    for(i = j = 0; i < HASH_SIZE_BYTES; i++)
    {
```

```
viernes  9 noviembre 01:31:34 2018 ]─[ ~/Lamport ]
$ ./lamport -g
[+] Calculating Lamport keypair . . .
[+] Obtaining random data from a secure source
[+] Calculating the public key from the private one

-----BEGIN LAMPORT PRIVATE KEY BLOCK-----
```

Photo by Arget on Unsplash

Types of bars from ProgressBar (gif from the library's page)

Progress Bars are Awesome

Progress Bars are a visual representation of how long is left for a process to complete. They save you having to worry whether the process has hang or try to predict how your code has been getting on. You can visually see in real time how well the script is progressing!

If you have never thought of or worked with progress bars before, it's easy to assume that they would add unnecessary complexity to your code and would be hard to maintain. It couldn't be further from the truth. In just a couple lines of code, we will see how to add progress bars to both our command line scripts, but also our PySimpleGUI UIs.

. . .

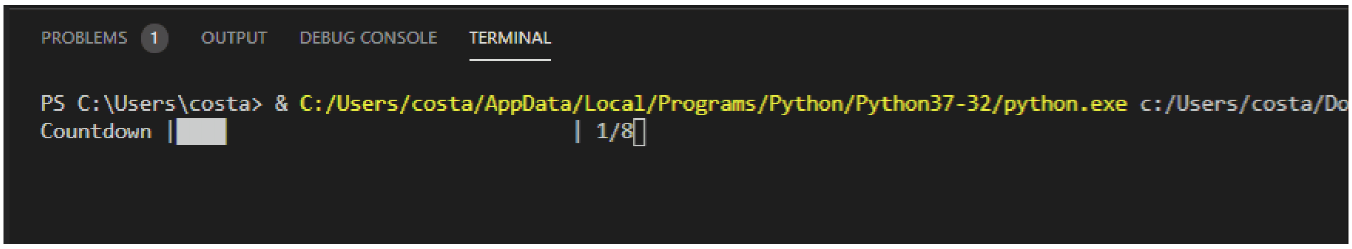
Using Progress

The first python library to review is *Progress*.

All you need to do is define the number of iterations you expect to do, the type of bar and let the bar know at every iteration.

```
import time
from progress.bar import IncrementalBar

mylist = [1,2,3,4,5,6,7,8]

bar = IncrementalBar('Countdown', max = len(mylist))

for item in mylist:
    bar.next()
    time.sleep(1)

bar.finish()
```

Returning:



```
PS C:\Users\costa> & C:/Users/costa/AppData/Local/Programs/Python/Python37-32/python.exe c:/Users/costa/Do
Countdown |███                              | 1/8
```

Incremental Bar of Progressbar

If you don’t like the format of the progress bar, there are many for you to choose from:

Types of bars from ProgressBar (gif from the library’s page)

Don’t forget to check out their documentation.

. . .

Using tqdm

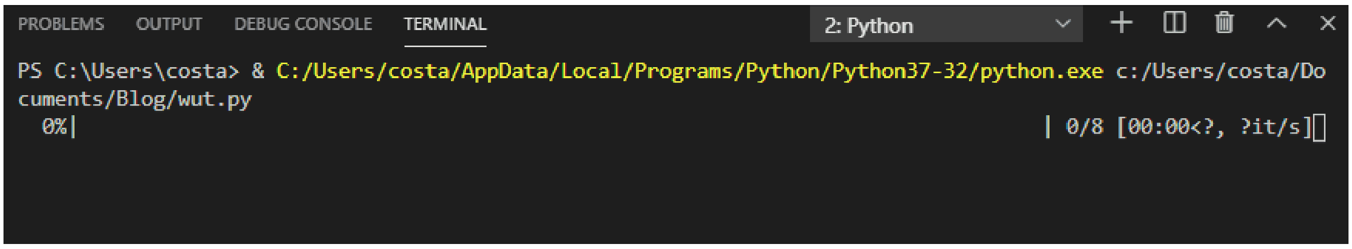
Next up in our review, is the *tqdm* library.

> A Fast, Extensible Progress Bar for Python and CLI

Much like the previous library, we’ve seen, with a couple of lines we can introduce a progres bar. Only a slight difference in terms of the set up:

```
import time
from tqdm import tqdm

mylist = [1,2,3,4,5,6,7,8]

for i in tqdm(mylist):
    time.sleep(1)
```

Giving us:



```
PS C:\Users\costa> & C:/Users/costa/AppData/Local/Programs/Python/Python37-32/python.exe c:/Users/costa/Do
cuments/Blog/wut.py
  0%|                                                                      | 0/8 [00:00<?, ?it/s]
```

Same as before, the bar does come with a few options. Do make sure to check the documentation.

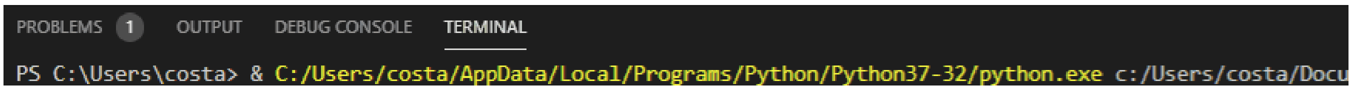
. . .

Using Alive Progress

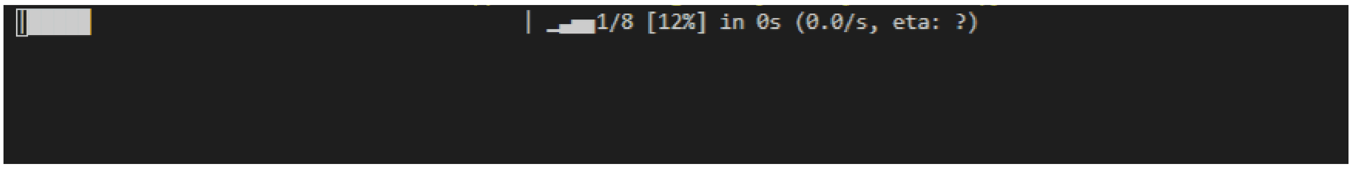
This library, as the name suggests, attempts to bring the progress bar alive. It has a few more animations than the previous progress bars we've seen. In terms of code, it's pretty similar however:

```
from alive_progress import alive_bar
import time

mylist = [1,2,3,4,5,6,7,8]

with alive_bar(len(mylist)) as bar:
    for i in mylist:
        bar()
        time.sleep(1)
```

With the bar looking as expected:



```
PS C:\Users\costa> & C:/Users/costa/AppData/Local/Programs/Python/Python37-32/python.exe c:/Users/costa/Docu
```



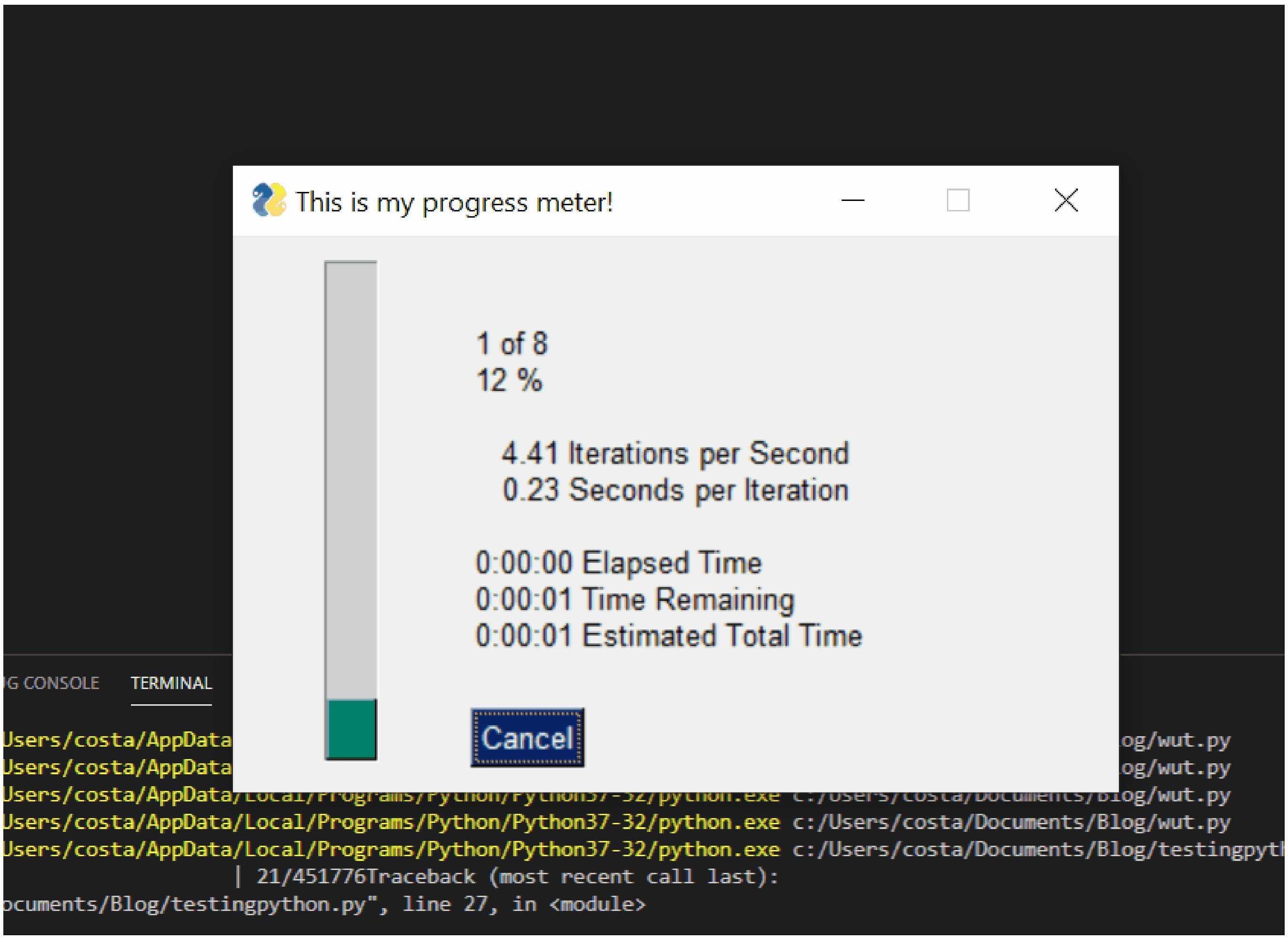
```
|             | ▁▂▃1/8 [12%] in 0s (0.0/s, eta: ?)
```

Once again, don't forget to explore the library's different features.

. . .

A Graphical Progress Bar using PySimpleGUI

In the same spirit that we've seen so far, we can add a single line of code to get a graphical progress bar to our command line script.



```
1 of 8
12 %

  4.41 Iterations per Second
  0.23 Seconds per Iteration

0:00:00 Elapsed Time
0:00:01 Time Remaining
0:00:01 Estimated Total Time
```

```
Users/costa/AppData                                                          og/wut.py
Users/costa/AppData                                                          og/wut.py
Users/costa/AppData/Local/Programs/Python/Python37-32/python.exe c:/Users/costa/Documents/Blog/wut.py
Users/costa/AppData/Local/Programs/Python/Python37-32/python.exe c:/Users/costa/Documents/Blog/wut.py
Users/costa/AppData/Local/Programs/Python/Python37-32/python.exe c:/Users/costa/Documents/Blog/testingpyt
                     | 21/451776Traceback (most recent call last):
ocuments/Blog/testingpython.py", line 27, in <module>
```

To achieve the above, all we need is:

```
import PySimpleGUI as sg
import time

mylist = [1,2,3,4,5,6,7,8]
```

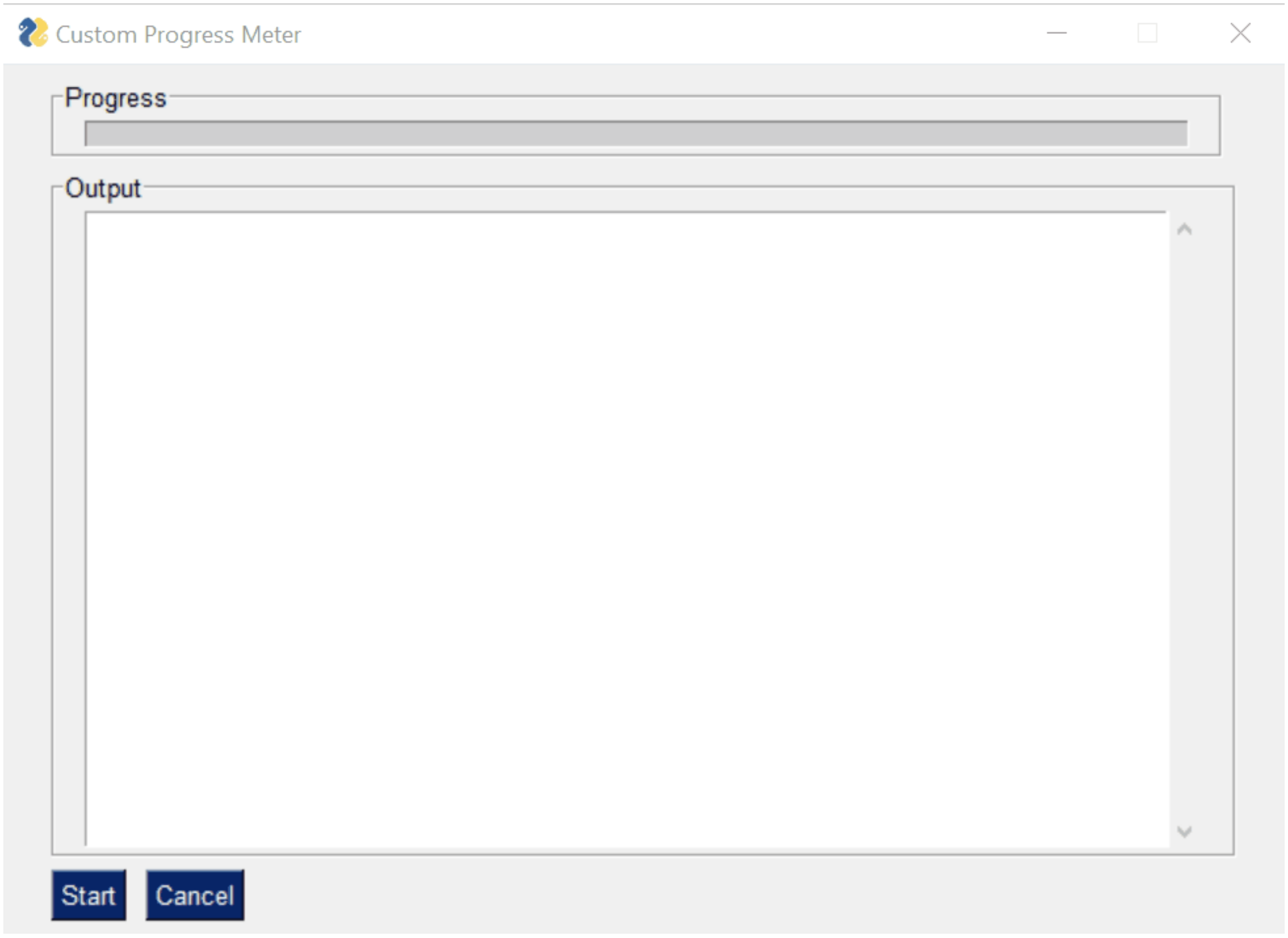
```
for i, item in enumerate(mylist):
    sg.one_line_progress_meter('This is my progress meter!', i+1, 
len(mylist), '-key-')
    time.sleep(1)
```

Thanks Mike for pointing this out!

Progress Bar in our PySimpleGUI Application

If you have been following my latest blogs, we explored how to quickly spin up Python UIs and then we built a comparison tool with a UI. To continue our learning journey, today we will explore how to integrate a progress bar.

The UI:



The code:

```
import PySimpleGUI as sg
import time

mylist = [1,2,3,4,5,6,7,8]
```

```
progressbar = [
    [sg.ProgressBar(len(mylist), orientation='h', size=(51, 10),
key='progressbar')]
]
outputwin = [
    [sg.Output(size=(78,20))]
]

layout = [
    [sg.Frame('Progress',layout= progressbar)],
    [sg.Frame('Output', layout = outputwin)],
    [sg.Submit('Start'),sg.Cancel()]
]

window = sg.Window('Custom Progress Meter', layout)
progress_bar = window['progressbar']

while True:
    event, values = window.read(timeout=10)
    if event == 'Cancel'  or event is None:
        break
    elif event == 'Start':
        for i,item in enumerate(mylist):
            print(item)
            time.sleep(1)
            progress_bar.UpdateBar(i + 1)

window.close()
```

. . .

Conclusion

This is it guys! With only a few lines of code you can implement progress bars in your python scripts! It's nothing too complicated and you no longer have to guess as to how your script is getting on!

Hope you found this useful!

)

Python Programming Data Science Software Engineering Start Up