

Overview of TQDM: Python Progress Bar Library (Updated 2024)

[BEGINNER](#)[LIBRARIES](#)[PANDAS](#)[PYTHON](#)[TECHNIQUE](#)

Hours of hard work will go in vain if our program becomes unresponsive during execution. We often come across large datasets in [machine learning](#) or longer loops that take a long while to complete, such as in Data Scrapping. While these commands are executing and massive loops are being processed behind the screen, it seems like an eternity of waiting time until the process is completed. Thus, One way of using tqdm is by creating a Progress Bar would solve this problem. Progress Bars would help us to look at the progress of our execution and to manage our anxiety levels. This tutorial will teach you how to implement Progress Bars in Python programming language. We will see how to do it in [Jupyter Notebook](#) and Command Line Interface (cli).

Learning Objectives

- Learn how to code a progress bar in Python.
- Understand how you can track progress in Python.
- Know in detail about tqdm () in Python.

This article was published as a part of the [Data Science Blogathon](#)

Quiz Time

Get ready for a journey through TQDM: Python Progress Bar Library. Test your knowledge about TQDM, a Python library for implementing progress bars and tracking task progress.

[Start Quiz](#)

Table of contents

- [What Is tqdm in Python?](#)
- [Using tqdm\(\).](#)
 - [Step 1: Install the Requirements](#)
 - [Step 2: Import the Libraries](#)
 - [Step 3: Using tqdm\(\).](#)
- [Using tqdm_notebook\(\).](#)
 - [Step 1: Import the Libraries](#)
 - [Step 2: Using tqdm_notebook\(\).](#)
- [Using tqdm_notebook\(\) in Pandas](#)
- [Comparison: Using tqdm_notebook\(\) and tqdm\(\) on Nested Loops](#)

- [How to make a terminal progress bar using tqdmChat?](#)
- [Conclusion](#)
- [Frequently Asked Questions](#)

What Is tqdm in Python?

Tqdm is a popular Python [library](#) that provides a simple and convenient way to add progress bars to loops and iterable objects. It gets its name from the Arabic name ***taqaddum***, which means 'progress.'

Using tqdm, you can wrap your loops or iterators with a progress bar, allowing you to track the progress of your code execution. It provides an intuitive and visually appealing progress bar that shows the percentage of completion, estimated time remaining, and other relevant information.

tqdm is easy to use. You simply import the library, wrap your iterable with the `tqdm()` function, and iterate over it. The progress bar will automatically update with each iteration, providing real-time feedback on the progress of your code.

Additionally, tqdm offers various customization options, such as setting the bar style, changing the progress update interval, and displaying additional information. It also supports nested progress bars, allowing you to track the progress of multiple loops or processes simultaneously.

Overall, tqdm is a valuable tool for enhancing the user experience when running long-running processes or iterating over large datasets by providing clear and informative progress visualization.

It can be easily implemented in our loops, functions, or even Pandas. Progress bars are pretty useful in Python because:

- One can see if the Kernal is still working
- Progress Bars are visually appealing to the eyes
- It gives Code Execution Time and Estimated Time for the code to complete, which would help while working on huge datasets



Image Source: dlpng.com

Further down, we will look at how to use `tqdm()`, `tqdm_notebook()`, and their application In Pandas in Jupyter Notebook.

Using tqdm()

tqdm library gives a console kind of progress bar for our processes. Using the library is a straightforward process explained in the following steps:

Step 1: Install the Requirements

First, we must install our required dependency libraries – tqdm and time. Open a New Jupyter Notebook and execute the following commands. You can also run these commands in the command line interface (cli) without using '!'

```
!pip install tqdm
```

```
!pip install time
```

This will complete the installation of tqdm, which you can start using after restarting the Kernel. You can find the official github repository [here](#).

Step 2: Import the Libraries

Import the newly installed tqdm and time libraries

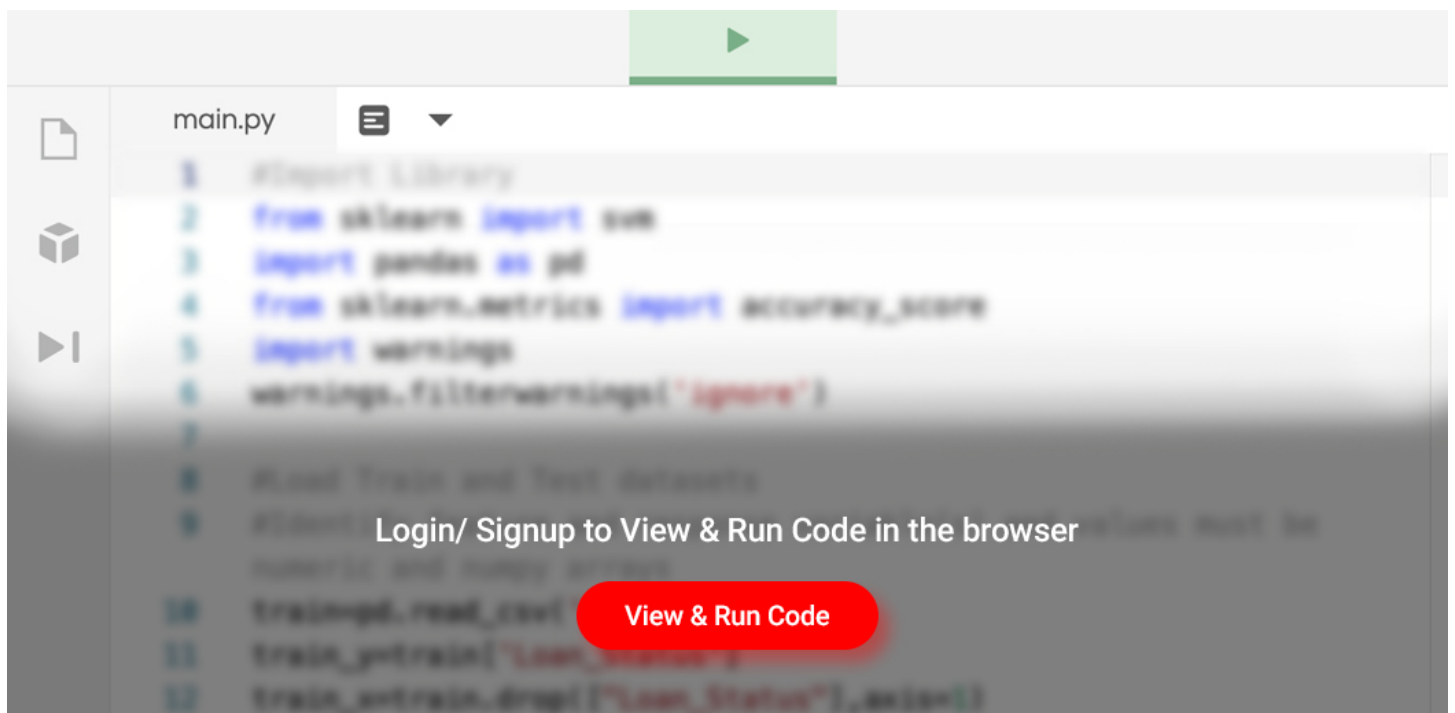
```
from tqdm import tqdm
```

```
import time
```

Step 3: Using tqdm()

tqdm will display the progress bar widgets that will help us understand the ETA (Estimated Time to Completion)

Python Code:



Now we will use the function `experiment()` that uses `tqdm()` on a simple program with a *for loop*.

```
for i in tq(range(20)): time.sleep(0.5)
```

Here **i** is the variable that takes a value of the number 0 to 19 during each iteration. The system will sleep for 0.5 seconds during each iteration before moving to the next iteration.

The complete code would look like this:

On **Completion** of Code Execution, we get:

```
from tqdm import tqdm
import time
for i in tq(range(20)):
    time.sleep(0.5)
100%|████████████████████████████████████████████████████████████████████████████████| 20/20 [00:10<00:00, 1.97it/s]
```

We can also give attributes to `tqdm()`, such as `desc`, which takes a string and will get added as a prefix before the progress bar. Thus,

```
from tqdm import tqdm import time for i in tqdm(range(20), desc = 'tqdm() Progress Bar'): time.sleep(0.5)
```

On **Completion** of Code Execution, we get:

```
from tqdm import tqdm
import time
for i in tqdm(range(20), desc = 'tqdm() Progress Bar'):
    time.sleep(0.5)
tqdm() Progress Bar: 100%|████████████████████████████████████████████████████████████████████████████████| 20/20 [00:10<00:00, 1.96it/s]
```

Apart from the progress bar, `tqdm` gives additional information such as the number of iterations completed out of the total number of iterations, Total Elapsed Time, Estimated Time to Complete the whole loop, and the speed of the loop in iterations per second (or it/s).

Using `tqdm_notebook()`

Unlike the `tqdm()`, **`tqdm_notebook()`** gives a colored version of progress bars. It has 3 sets of colours by default.

A moving **Blue Bar** shows a process undergoing, a stable **Green Bar** shows that the process is completed, and a **Red Bar** shows that the process has been stopped. Interestingly, like the `tqdm()`, `tqdm_notebook()` too has a straightforward way of implementation.

Step 1: Import the Libraries

```
from tqdm.notebook import tqdm_notebook import time
```

Step 2: Using `tqdm_notebook()`

```
for i in tqdm_notebook(range(10)): time.sleep(0.5)
```

Here `i` is the variable that takes a value of the number 0 to 19 during each iteration. The system will sleep for 0.5 seconds during each iteration before moving to the next iteration.

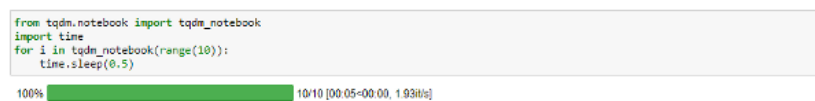
The complete code would look like this:

```
from tqdm.notebook import tqdm_notebook import time for i in tqdm_notebook(range(10)): time.sleep(0.5)
```

On **Executing** the Code, we get:



On **Completion** of Code execution, we get:



If the Code execution is **Terminated**:

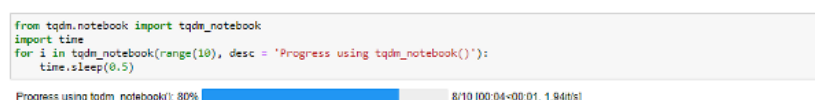


Notice the color of the Progress Bar.

We can give additional arguments into **tqdm_notebook()**, such as **desc**, which adds a prefix to the Progress Bar. The Code would look like this:

```
from tqdm.notebook import tqdm_notebook import time for i in tqdm_notebook(range(10), desc = 'Progress using tqdm_notebook()'): time.sleep(0.5)
```

On Executing the Code, we get:



Using **tqdm_notebook()** in Pandas


Both **tqdm()** and **tqdm_notebook()** can be used in Pandas. One way is to use them with **for loops** with Pandas Series, which works the same as with the loops we have seen earlier. Another way is to use them in Pandas **.apply()** method. To use **tqdm()** or **tqdm_notebook()** for **.apply()**, the **.apply()** needs to be replaced by **.progress_apply()**.

For example, let's take a dataset from [Kaggle](#):

```
import pandas as pd
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
df.head()
tqdm_notebook.pandas()
df['Churn'] = df['Churn'].progress_apply(lambda x: 1 if x == 'Yes' else 0)
```

On **Completion** of Code Execution, we get:

```
import pandas as pd
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
df.head()
tqdm_notebook.pandas()
df['Churn'] = df['Churn'].progress_apply(lambda x: 1 if x == 'Yes' else 0)
```



The **tqdm_notebook.pandas()** is responsible for displaying the progress bar. The progress bar shows the apply function being applied on all values of the **Churn** column. It is noteworthy that the **progress.apply()** works the same as the **.apply()** method of Pandas.

Comparison: Using tqdm_notebook() and tqdm() on Nested Loops

First, Let's see an example of using tqdm() on nested loops:

```
from tqdm import tqdm
```

```
for i in tqdm(range(2), desc = 'Loop 1'):
    for j in tqdm(range(20,25), desc = 'Loop 2'):
        time.sleep(0.5)
```

On **Completion** of Code Execution, we get:

```
from tqdm import tqdm
for i in tqdm(range(2), desc = 'Loop 1'):
    for j in tqdm(range(20,25), desc = 'Loop 2'):
        time.sleep(0.5)
```

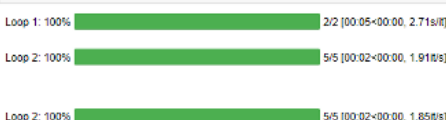


Now, let's see tqdm_notebook() on nested loops:

```
from tqdm.notebook import tqdm_notebook
for i in tqdm_notebook(range(2), desc = 'Loop 1'):
    for j in tqdm_notebook(range(20,25), desc = 'Loop 2'):
        time.sleep(0.5)
```

On **Completion** of Code Execution, we get:

```
from tqdm.notebook import tqdm_notebook
for i in tqdm_notebook(range(2), desc = 'Loop 1'):
    for j in tqdm_notebook(range(20,25), desc = 'Loop 2'):
        time.sleep(0.5)
```



It is evident that **tqdm()** has more progress bars. Each iteration of Loop 1 shows a separate progress bar; each iteration of Loop 1 shows individual progress bars for each iteration of Loop 2.

Thus, the first progress bar of Loop 1 shows 5 progress bars of Loop 2, then again, for the second progress bar of Loop 1, it shows 5 separate progress bars of Loop 2, and so on.

On the other hand, Progress Bars in **tqdm_notebook()** are very intuitive. It shows only one progress bar for Loop 1, and for each iteration of Loop 1, one progress bar for Loop 2. It might sound difficult to read, but it becomes understandable once you run the code.

How to make a terminal progress bar using tqdmChat?

1. First, you need to install the tqdm library if you haven't already. You can do this by running `pip install tqdm` in your terminal or command prompt.
2. Next, you write a Python script with the following steps:
 - Import the tqdm library and the time module.
 - Decide how many steps your task will have and store this number in a variable (for example, `total_iterations = 100`).
 - Create a progress bar using tqdm with `total` set to the total number of steps. You can also give it a description to display (like "Processing").
 - Inside a loop that goes through each step of your task, do the work for that step (you can simulate this with `time.sleep(0.1)` to make it take some time).
 - After each step, update the progress bar using `progress_bar.update(1)`.
 - When the loop finishes, close the progress bar with `progress_bar.close()`.
 - Optionally, print a message to indicate that the task is complete.
3. Run your Python script, and you should see a progress bar in your terminal that updates as your task progresses.

You can create a terminal progress bar using the tqdm library in Python. Here's a simple example

```
from tqdm import tqdm import time # Define the total number of iterations for the progress bar
total_iterations = 100 # Create a tqdm progress bar progress_bar = tqdm(total=total_iterations,
desc="Processing") # Simulate some task that iterates for i in range(total_iterations): # Do some processing
here time.sleep(0.1) # Simulating a delay # Update the progress bar progress_bar.update(1) # Close the
progress bar progress_bar.close() print("Task completed!")
```

Conclusion

Thus, [tracking](#) the progresses helps a lot of time and saves us from the state of confusion. Apart from the **desc** attribute of both `tqdm()` and `tqdm_notebook()`, it takes more attributes that depend upon the end-user and its need. You can also track progress using **progressbar**, **progressbar2**, and **Alive progress**.

An advantage of using `tqdm()` over the others is that it has very detailed [documentation](#), which would help one to refer anytime. There is numerous customization possible for the progress bars, and more might come in future updates.

Key Takeaways

- tqdm is one of the most common libraries used by data scientists.
- It is a popular python library that tracks the time taken to complete a task.

Frequently Asked Questions

Q1. How do you code a progress bar in python?

A. Progress bars can be implemented in python using tqdm() or tkinter(). For example,

```
from tqdm import tqdm
temp=[i for i in tqdm(range(10000))]
```

Q2. What is tqdm in python?

A. tqdm is a python library that displays the progress bar. Simply put, it helps us understand how much of the task is being completed.

Q3. What is the difference between tqdm and tkinter?

A. tqdm is a python library that tracks the time taken to complete the task. tkinter is another python library that is mainly for GUI and animations.

Q4. What is the range of tqdm in Python?

A. The range of tqdm in Python refers to the number of iterations or elements in the iterable that the progress bar will track. It can be specified using Python's range() function or any other iterable object.

Q5. What is tqdm in deep learning?

A. In deep learning, tqdm monitors the progress of training a neural network. It provides a real-time progress bar that updates with each iteration, allowing users to monitor the training process and detect any issues that may arise.

Q6. How to install tqdm in Python?

A. To install tqdm in Python, one can use the pip package manager by running the command "pip install tqdm" in the terminal or command prompt. Alternatively, one can install tqdm using Anaconda or by downloading the source code and installing it manually.

The media shown in this article are not owned by Analytics Vidhya and are used at the Author's discretion.

Article Url - <https://www.analyticsvidhya.com/blog/2021/05/how-to-use-progress-bars-in-python/>



Rahul Shah

IT Engineering Graduate currently pursuing Post Graduate Diploma in Data Science.