Open in app Get started

Published in Towards Data Science

You have **2** free member-only stories left this month.
Sign up for Medium and get an extra one

Dario Radečić Follow
May 7, 2021 · 6 min read · · Listen

Save

How to Schedule Python Scripts With Cron — The Only Guide You'll Ever Need

Automate your Python script execution — works on Linux and macOS

After reading, you'll know how to automate the execution of two Python scripts for fetching, parsing, and saving data from the web. Let's get started!

The article is structured as follows:

What is Cron?

Think of Cron as one of the easiest ways to schedule tasks in Linux and macOS environments. The word "Cron" comes from the Greek word "Chronos" (time), and the word "Crontab" stands for "Cron table" or time table. You'll learn in a bit what this table refers to.

You should use Cron any time you want to automate something, like an OS job or a Python script. Needless to say, but an automated Python script can do basically anything.

On Linux and macOS, the Crontab consists of six fields. The first five are reserved for the date and time of scheduled execution (minute, day of month, month of year, day of week), and the last field is reserved for a command to be executed.

You're free to use asterisks, values, and ranges for the date-time fields, but more on that in a bit.

Writing the Scripts

Today we'll schedule two simple scripts. Their job is to fetch the data from the following dummy websites:

- https://jsonplaceholder.typicode.com/users
- https://jsonplaceholder.typicode.com/posts

These sites contain dummy data on various subjects, but we'll focus only on the users and posts. The only reason we're handling two sites is to show how to schedule cron jobs at different time intervals.

Both scripts will fetch the data via `requests.get()` API call, parse the JSON data, process it (keep only certain attributes), convert it to a Pandas DataFrame, and save it to a CSV file. Sounds like a lot, but it's only a few lines of code.

Let's start with the `get_users.py` file:

```python
import os
import json
import requests
import pandas as pd
from datetime import datetime


def fetch(url: str) -> list:
    res = requests.get(url)
    return json.loads(res.content)


def process(users: list) -> pd.DataFrame:
    processed = []
    for user in users:
        processed.append({
            'ID': user['id'],
            'Name': user['name'],
            'Username': user['username'],
            'Email': user['email'],
            'Phone': user['phone'],
def save(users: pd.DataFrame, path: str) -> None:
    users.to_csv(path, index=False)


if __name__ == '__main__':
    users = fetch(url='https://jsonplaceholder.typicode.com/users')
    users = process(users=users)
    curr_timestamp = int(datetime.timestamp(datetime.now()))
    path = os.path.abspath(f'Documents/cron-tutorial/output/users_{curr_timestamp}.csv')
    save(users=users, path=path)
```

The `get_posts.py` file will be more or less identical — except for the processing part:

```python
import os
import json
import requests
import pandas as pd
from datetime import datetime


def fetch(url: str) -> list:
    res = requests.get(url)
    return json.loads(res.content)


def process(posts: list) -> pd.DataFrame:
    processed = []
    for post in posts:
        processed.append({
            'ID': post['id'],
            'UserID': post['userId'],
            'Title': post['title']
        })
    return pd.DataFrame(processed)


def save(posts: pd.DataFrame, path: str) -> None:
    posts.to_csv(path, index=False)


if __name__ == '__main__':
```

Finally, make sure to create the `output` just where your scripts are. That's all we need. If you were to run the scripts, the CSVs would save in the `output` directory with the `<name>_<timeastamp>.csv` name structure. The timestamps are here to ensure files don't get overridden.

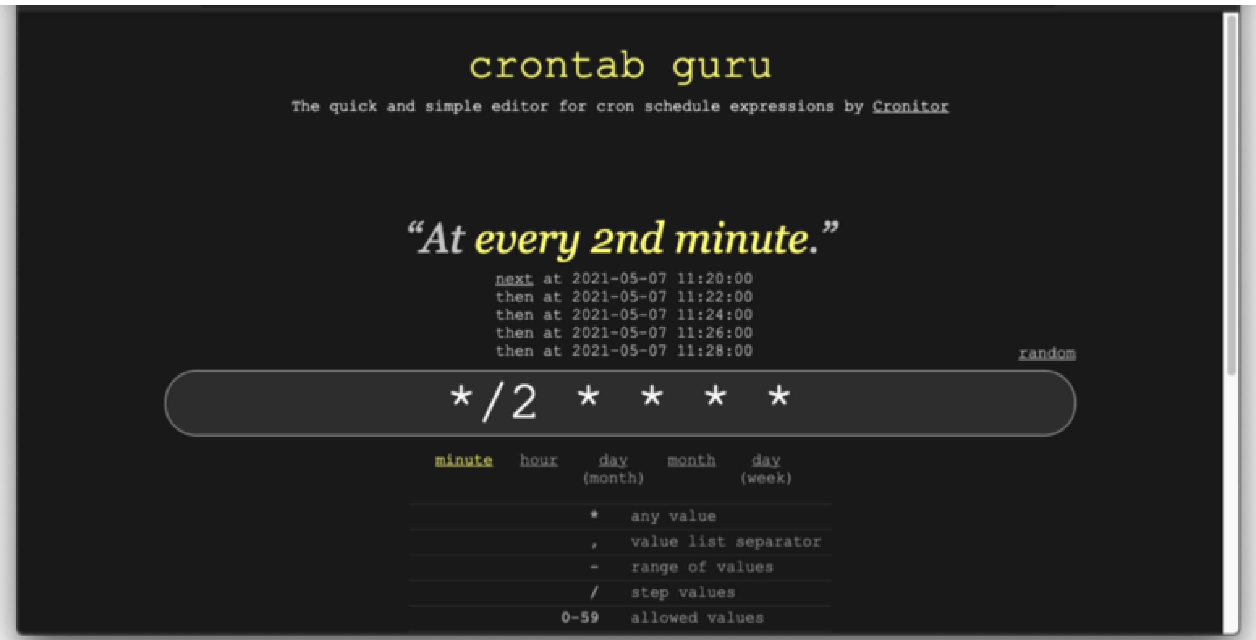
Let's see how to schedule the tasks next.

Editing the Crontab File

It doesn't matter if you're on Linux or macOS — this section will be identical. There are some permission gotchas for macOS, but we'll get to that later.

As discussed before, you must follow a specific syntax to schedule cron jobs. The good news is, you can use Crontab.guru website to work on your scheduling.

We want `get_users.py` to run on every even minute (e.g., 0, 2, 4) and `get_posts.py` to run on every odd minute (e.g., 1, 3, 5). Here's the correct pattern to run a job on every even minute:



crontab guru

The quick and simple editor for cron schedule expressions by Cronitor

"At every 2nd minute."

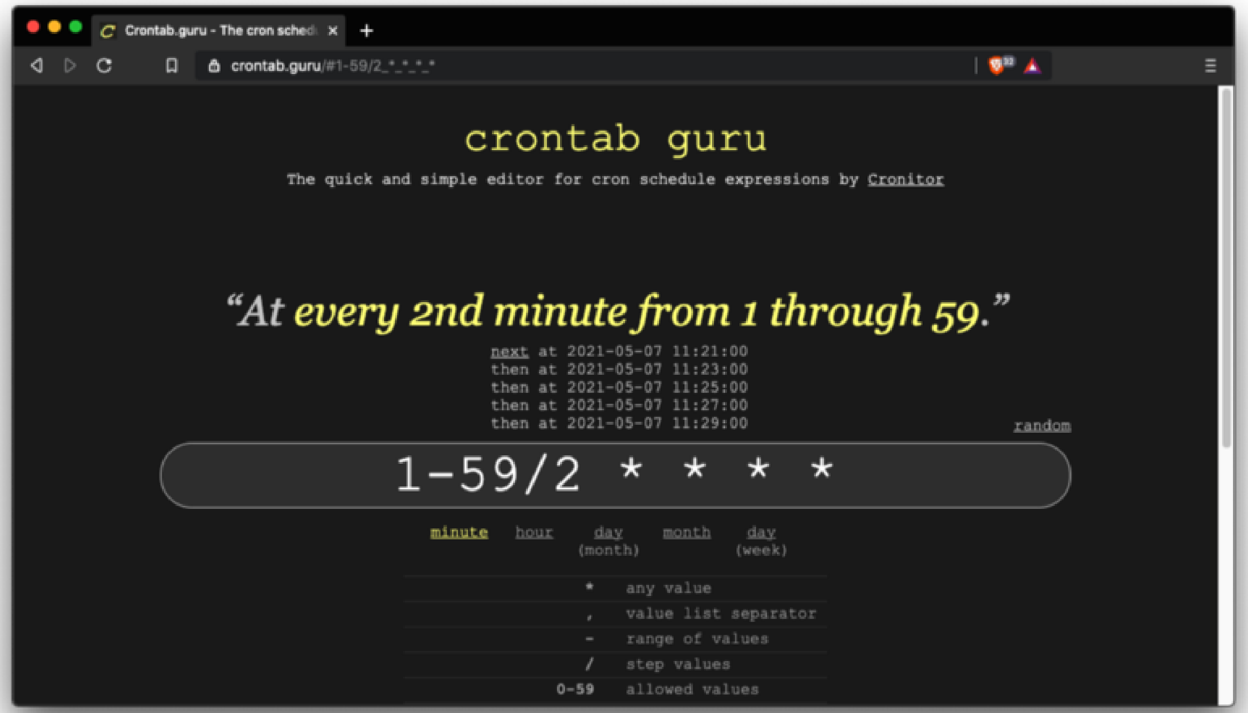
```
next at 2021-05-07 11:20:00
then at 2021-05-07 11:22:00
then at 2021-05-07 11:24:00
then at 2021-05-07 11:26:00
then at 2021-05-07 11:28:00
```

```
*/2 * * * *
```

*	any value
,	value list separator
-	range of values
/	step values
0-59	allowed values

Image 1 — Cron job pattern for every even minute execution (image by author)

And here's for the odd minutes:



crontab guru

The quick and simple editor for cron schedule expressions by Cronitor

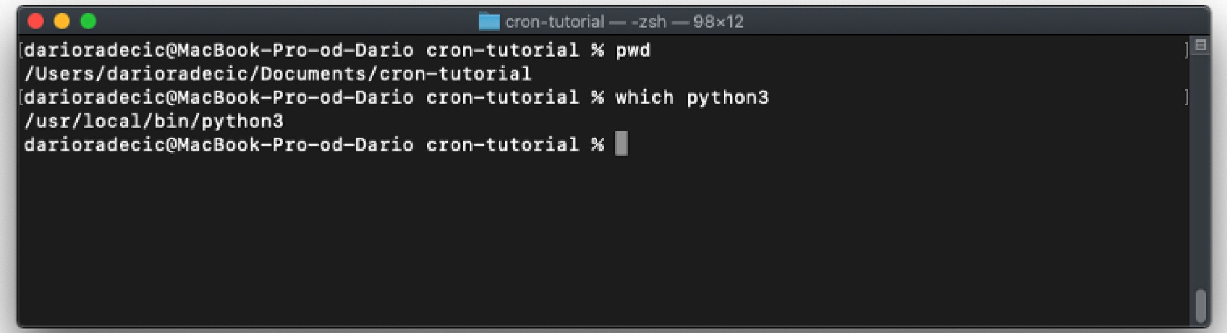
"At every 2nd minute from 1 through 59."

```
next at 2021-05-07 11:21:00
then at 2021-05-07 11:23:00
then at 2021-05-07 11:25:00
then at 2021-05-07 11:27:00
then at 2021-05-07 11:29:00
```

```
1-59/2 * * * *
```

*	any value
,	value list separator
-	range of values
/	step values
0-59	allowed values

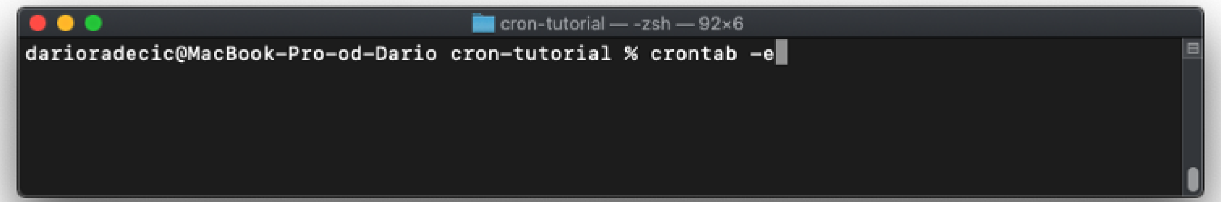
Image 2 — Cron job pattern for every odd minute execution (image by author) script folder and Python:



```
darioradecic@MacBook-Pro-od-Dario cron-tutorial % pwd
/Users/darioradecic/Documents/cron-tutorial
darioradecic@MacBook-Pro-od-Dario cron-tutorial % which python3
/usr/local/bin/python3
darioradecic@MacBook-Pro-od-Dario cron-tutorial %
```

Image 3 — Obtaining absolute paths (image by author)

Once you have these, enter the `crontab -e` command to edit a cron file, or to make one if it doesn't exist:

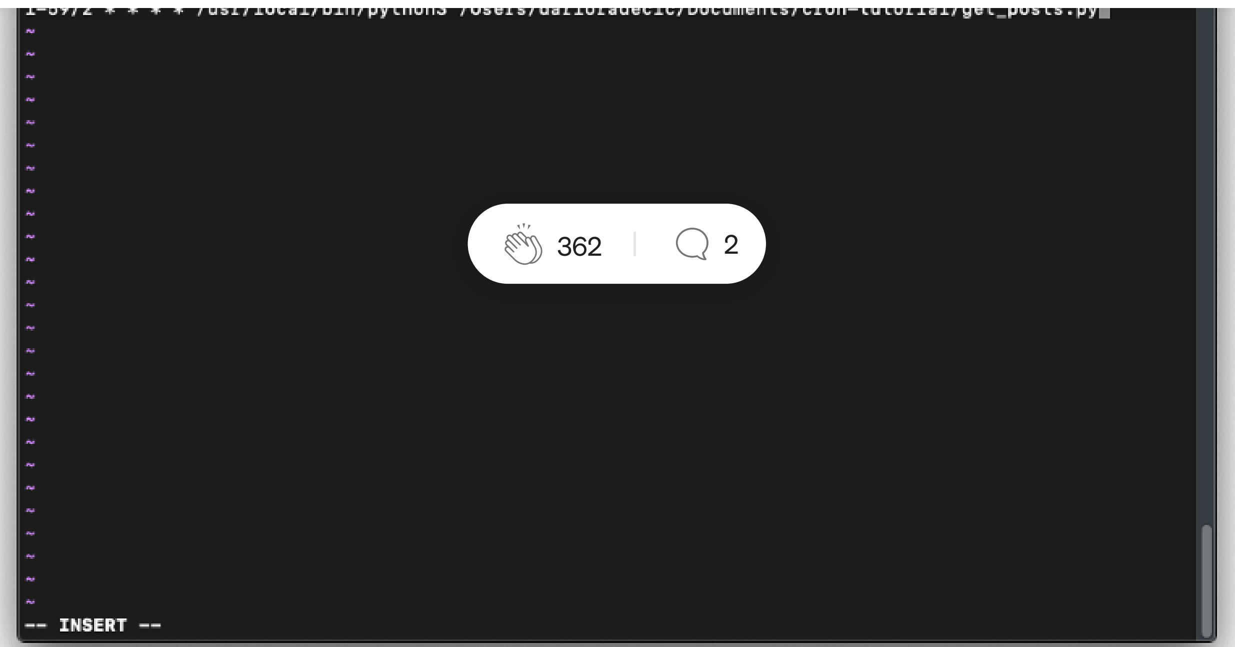


```
darioradecic@MacBook-Pro-od-Dario cron-tutorial % crontab -e
```

Image 4 — Accessing crontab file (image by author)

It will open a VIM editor — from there, click on the `I` key on your keyboard to enter insertion mode. You'll have to specify the scheduling pattern, full path to the Python executable, and full path to the script to make scheduling work. Use the following image as a reference:

Open in app
Get started

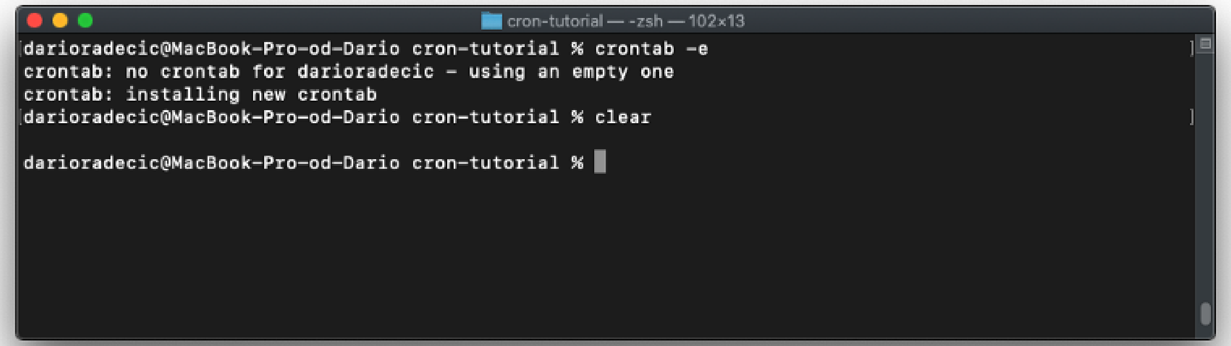


```
1-59/2 * * * * /usr/local/bin/python3 /Users/darioradecic/Documents/cron-tutorial/get_posts.py
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

362 2

Image 5 — Editing a crontab file (image by author)

Once done, press the `ESC` key to exit the insert mode, immediately followed by `:wq` and `ENTER` keys. This will save the crontab files, and you'll be back in the Terminal window instantly:

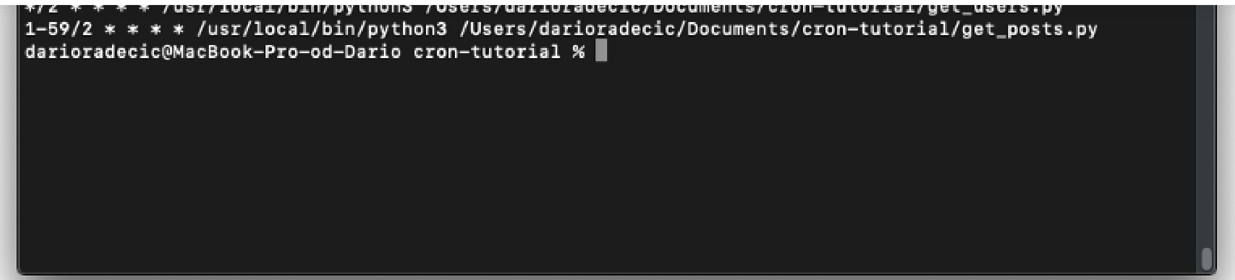


```
darioradecic@MacBook-Pro-od-Dario cron-tutorial % crontab -e
crontab: no crontab for darioradecic - using an empty one
crontab: installing new crontab
darioradecic@MacBook-Pro-od-Dario cron-tutorial % clear

darioradecic@MacBook-Pro-od-Dario cron-tutorial % 
```

Image 6 — Successful crontab file creation (image by author)

To verify the file was successfully saved, you can use the `crontab -l` command — it will list all scheduled jobs:



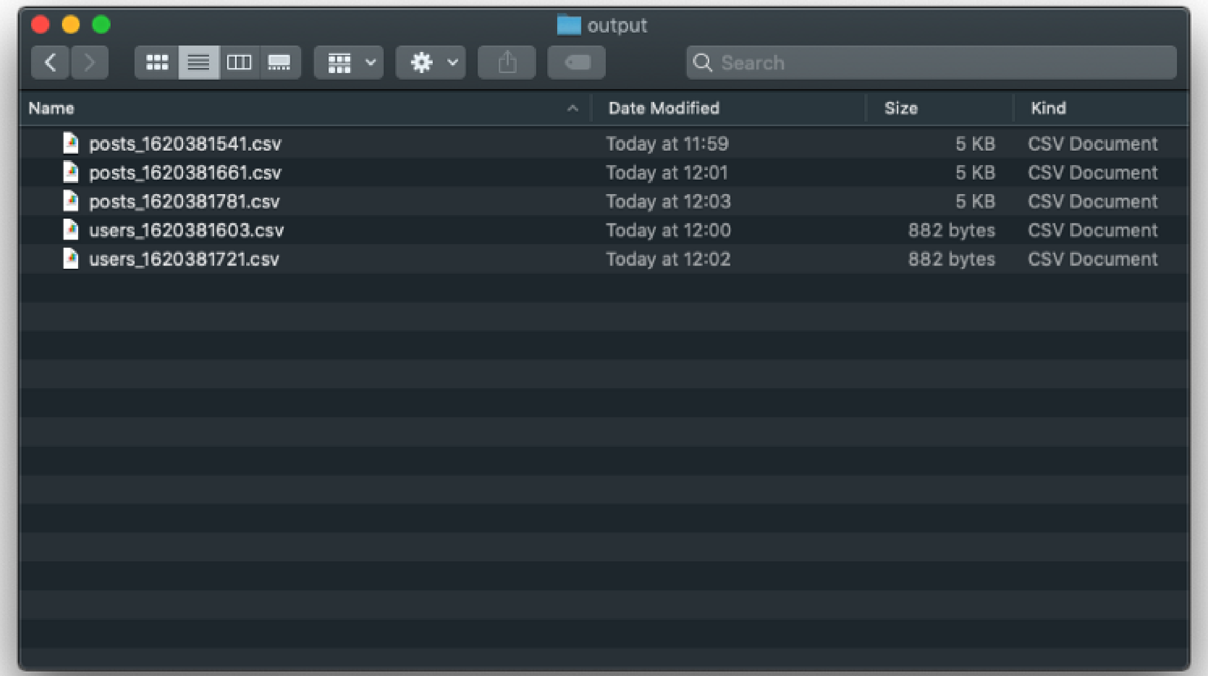
```
*/2 * * * * /usr/local/bin/python3 /Users/darioradecic/Documents/cron-tutorial/get_users.py
1-59/2 * * * * /usr/local/bin/python3 /Users/darioradecic/Documents/cron-tutorial/get_posts.py
darioradecic@MacBook-Pro-od-Dario cron-tutorial %
```

Image 7 — Listing all scheduled jobs (image by author)

And that's all you have to do. Let's check if scheduling works in the next section.

Testing

There's nothing left for you to do but to sit and watch your scripts executed every minute. Here's how my `output` folder looks like after a couple of minutes:



Name	Date Modified	Size	Kind
posts_1620381541.csv	Today at 11:59	5 KB	CSV Document
posts_1620381661.csv	Today at 12:01	5 KB	CSV Document
posts_1620381781.csv	Today at 12:03	5 KB	CSV Document
users_1620381603.csv	Today at 12:00	882 bytes	CSV Document
users_1620381721.csv	Today at 12:02	882 bytes	CSV Document

Image 8 — Output folder (image by author)

Open in app Get started

MacOS Gotchas

Linux users shouldn't have any issues, but the story is different on macOS. By default, macOS doesn't give full disc access to Terminal and Cron, so you'll have to do that manually.

Just to be clear — you won't get any errors, but the `output` folder will remain empty.

If that's the case, please follow this article — it will show you how to grant full disc access to both Terminal and Cron.

Conclusion

And there you have it — how to easily schedule Python scripts with Cron on Linux and macOS.

The possibilities are endless — from scheduled web scraping to automated execution of ETL pipelines. The code can change, but the principle remains identical.

Have fun!

Loved the article? Become a Medium member to continue learning without limits. I'll receive a portion of your membership fee if you use the following link, with no extra cost to you.

Join Medium with my referral link - Dario Radečić

As a Medium member, a portion of your membership fee goes to

Open in app Get started

Learn more

- 3 Programming Books Every Data Scientist Must Read
- How to Make Python Statically Typed — The Essential Guide
- Object Orientated Programming with Python — Everything You Need to Know
- Python Dictionaries: Everything You Need to Know
- Introducing f-Strings — The Best Options for String Formatting in Python

Stay connected

- Follow me on Medium for more stories like this
- Sign up for my newsletter
- Connect on LinkedIn
- Check out my website

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

By signing up, you will create a Medium account if you don't already have one. Review