

How to Make an Instagram Bot With Python and InstaPy

by [Jahongir Rahmonov](#) ⌚ Apr 06, 2020 💬 [53 Comments](#)

🔑 [intermediate](#) [projects](#)

[Mark as Completed](#) [🔖](#)

[🐦 Tweet](#) [f Share](#) [✉ Email](#)

Table of Contents

- [How Instagram Bots Work](#)
- [How to Automate a Browser](#)
- [How to Use the Page Object Pattern](#)
- [How to Build an Instagram Bot With InstaPy](#)
 - [Essential Features](#)
 - [Additional Features in InstaPy](#)
- [Conclusion](#)



**Master Real-World Python Skills
With a Community of Experts**

Level Up With Unlimited Access to Our Vast Library
of Python Tutorials and Video Lessons

[Watch Now »](#)

[🔧 Remove ads](#)

What do [SocialCaptain](#), [Kicksta](#), [Instavast](#), and [many other companies](#) have in common? They all help you reach a greater audience, gain more followers, and get more likes on Instagram while you hardly lift a finger. They do it all through automation, and people pay them a good deal of money for it. But you can do the same thing—for free—using InstaPy!

In this tutorial, you'll learn how to build a bot with [Python](#) and [InstaPy](#), a library by [Tim Großmann](#) which **automates** your Instagram activities so that you gain more followers and likes with minimal manual input. Along the way, you'll learn about browser automation with [Selenium](#) and the **Page Object Pattern**, which together serve as the basis for InstaPy.

In this tutorial, you'll learn:

- How **Instagram bots** work

[Help](#)

- How to automate a browser with [Selenium](#)
- How to use the **Page Object Pattern** for better readability and testability
- How to build an Instagram bot with **InstaPy**

You'll begin by learning how Instagram bots work before you build one.

Important: Make sure you check [Instagram's Terms of Use](#) before implementing any kind of automation or scraping techniques.

Free Bonus: 5 Thoughts On Python Mastery, a free course for Python developers that shows you the roadmap and the mindset you'll need to take your Python skills to the next level.

How Instagram Bots Work

How can an automation script gain you more followers and likes? Before answering this question, think about how an actual person gains more followers and likes.

They do it by being consistently active on the platform. They post often, follow other people, and like and leave comments on other people's posts. Bots work exactly the same way: They follow, like, and comment on a consistent basis according to the criteria you set.

The better the criteria you set, the better your results will be. You want to make sure you're targeting the right groups because the people your bot interacts with on Instagram will be more likely to interact with *your* content.

For example, if you're selling women's clothing on Instagram, then you can instruct your bot to like, comment on, and follow mostly women or profiles whose posts include hashtags such as #beauty, #fashion, or #clothes. This makes it more likely that your target audience will notice your profile, follow you back, and start interacting with your posts.


How does it work on the technical side, though? You can't use the [Instagram Developer API](#) since it is fairly limited for this purpose. Enter **browser automation**. It works in the following way:

1. You serve it your credentials.
2. You set the criteria for who to follow, what comments to leave, and which type of posts to like.
3. Your bot opens a browser, types in `https://instagram.com` on the address bar, logs in with your credentials, and starts doing the things you instructed it to do.

Next, you'll build the initial version of your Instagram bot, which will automatically log in to your profile. Note that you won't use InstaPy just yet.



 [The Real Python Podcast »](#)

 [Remove ads](#)

How to Automate a Browser

For this version of your Instagram bot, you'll be using [Selenium](#), which is the tool that InstaPy uses under the hood.

First, [install Selenium](#). During installation, make sure you also install the [Firefox WebDriver](#) since the latest version of [InstaPy dropped support for Chrome](#). This also means that you need the [Firefox browser](#) installed on your computer.

Now, create a Python file and write the following code in it:

Python

```
1 from time import sleep
2 from selenium import webdriver
3
4 browser = webdriver.Firefox()
5
6 browser.get('https://www.instagram.com/')
7
8 sleep(5)
9
10 browser.close()
```

Run the code and you'll see that a Firefox browser opens and directs you to the Instagram login page. Here's a line-by-line breakdown of the code:

- **Lines 1 and 2** import `sleep` and `webdriver`.
- **Line 4** initializes the Firefox driver and sets it to `browser`.
- **Line 6** types `https://www.instagram.com/` on the address bar and hits .
- **Line 8** waits for five seconds so you can see the result. Otherwise, it would close the browser instantly.
- **Line 10** closes the browser.

This is the Selenium version of Hello, world. Now you're ready to add the code that logs in to your Instagram profile. But first, think about how you would log in to your profile manually. You would do the following:

1. Go to `https://www.instagram.com/`.
2. Click the login link.
3. Enter your credentials.
4. Hit the login button.

The first step is already done by the code above. Now change it so that it clicks on the login link on the Instagram home page:

Python

```
1 from time import sleep
2 from selenium import webdriver
3
4 browser = webdriver.Firefox()
5 browser.implicitly_wait(5)
6
7 browser.get('https://www.instagram.com/')
8
9 login_link = browser.find_element_by_xpath("//a[text()='Log in']")
10 login_link.click()
11
12 sleep(5)
13
14 browser.close()
```

Note the highlighted lines:

- **Line 5** sets five seconds of waiting time. If Selenium can't find an element, then it waits for five seconds to allow everything to load and tries again.
- **Line 9** finds the element `<a>` whose text is equal to `Log in`. It does this using [XPath](#), but there are a [few other methods](#) you could use.
- **Line 10** clicks on the found element `<a>` for the login link.

Run the script and you'll see your script in action. It will open the browser, go to Instagram, and click on the login link to go to the login page.

On the login page, there are three important elements:

1. The username input
2. The password input

3. The login button

Next, change the script so that it finds those elements, enters your credentials, and clicks on the login button:

Python

```

1  from time import sleep
2  from selenium import webdriver
3
4  browser = webdriver.Firefox()
5  browser.implicitly_wait(5)
6
7  browser.get('https://www.instagram.com/')
8
9  login_link = browser.find_element_by_xpath("//a[text()='Log in']")
10 login_link.click()
11
12 sleep(2)
13
14 username_input = browser.find_element_by_css_selector("input[name='username']")
15 password_input = browser.find_element_by_css_selector("input[name='password']")
16
17 username_input.send_keys("<your username>")
18 password_input.send_keys("<your password>")
19
20 login_button = browser.find_element_by_xpath("//button[@type='submit']")
21 login_button.click()
22
23 sleep(5)
24
25 browser.close()
```

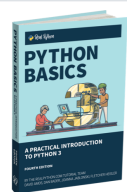
Here's a breakdown of the changes:

1. **Line 12** sleeps for two seconds to allow the page to load.
2. **Lines 14 and 15** find username and password inputs by CSS. You could use [any other method](#) that you prefer.
3. **Lines 17 and 18** type your username and password in their respective inputs. Don't forget to fill in <your username> and <your password>!
4. **Line 20** finds the login button by XPath.
5. **Line 21** clicks on the login button.

Run the script and you'll be automatically logged in to to your Instagram profile.

You're off to a good start with your Instagram bot. If you were to continue writing this script, then the rest would look very similar. You would find the posts that you like by scrolling down your feed, find the like button by CSS, click on it, find the comments section, leave a comment, and continue.

The good news is that all of those steps can be handled by InstaPy. But before you jump into using InstaPy, there is one other thing that you should know about to better understand how InstaPy works: the **Page Object Pattern**.



Your **Practical Introduction to Python 3** »

[Remove ads](#)

How to Use the Page Object Pattern

Now that you've written the login code, how would you write a test for it? It would look something like the following:

Python

```
def test_login_page(browser):
    browser.get('https://www.instagram.com/accounts/login/')
    username_input = browser.find_element_by_css_selector("input[name='username']")
    password_input = browser.find_element_by_css_selector("input[name='password']")
    username_input.send_keys("<your username>")
    password_input.send_keys("<your password>")
    login_button = browser.find_element_by_xpath("//button[@type='submit']")
    login_button.click()

    errors = browser.find_elements_by_css_selector('#error_message')
    assert len(errors) == 0
```

Can you see what's wrong with this code? It doesn't follow the [DRY principle](#). That is, the code is duplicated in both the application and the test code.

Duplicating code is especially bad in this context because Selenium code is dependent on UI elements, and UI elements tend to change. When they do change, you want to update your code in one place. That's where the [Page Object Pattern](#) comes in.

With this pattern, you create **page object classes** for the most important pages or fragments that provide interfaces that are straightforward to program to and that hide the underlying widgetry in the window. With this in mind, you can rewrite the code above and create a HomePage class and a LoginPage class:

Python

```
from time import sleep

class LoginPage:
    def __init__(self, browser):
        self.browser = browser

    def login(self, username, password):
        username_input = self.browser.find_element_by_css_selector("input[name='username']")
        password_input = self.browser.find_element_by_css_selector("input[name='password']")
        username_input.send_keys(username)
        password_input.send_keys(password)
        login_button = browser.find_element_by_xpath("//button[@type='submit']")
        login_button.click()
        sleep(5)

class HomePage:
    def __init__(self, browser):
        self.browser = browser
        self.browser.get('https://www.instagram.com/')

    def go_to_login_page(self):
        self.browser.find_element_by_xpath("//a[text()='Log in']").click()
        sleep(2)
        return LoginPage(self.browser)
```

The code is the same except that the home page and the login page are represented as classes. The classes encapsulate the mechanics required to find and manipulate the data in the UI. That is, there are methods and [accessors](#) that allow the software to do anything a human can.

One other thing to note is that when you navigate to another page using a page object, it returns a page object for the new page. Note the returned value of `go_to_log_in_page()`. If you had another class called `FeedPage`, then `login()` of the `LoginPage` class would return an instance of that: `return FeedPage()`.

Here's how you can put the Page Object Pattern to use:

Python

```
from selenium import webdriver

browser = webdriver.Firefox()
browser.implicitly_wait(5)

home_page = HomePage(browser)
login_page = home_page.go_to_login_page()
login_page.login("<your username>", "<your password>")

browser.close()
```

It looks much better, and the test above can now be rewritten to look like this:

Python

```
def test_login_page(browser):
    home_page = HomePage(browser)
    login_page = home_page.go_to_login_page()
    login_page.login("<your username>", "<your password>")

    errors = browser.find_elements_by_css_selector('#error_message')
    assert len(errors) == 0
```

With these changes, you won't have to touch your tests if something changes in the UI.

For more information on the Page Object Pattern, refer to the [official documentation](#) and to [Martin Fowler's article](#).

Now that you're familiar with both Selenium and the Page Object Pattern, you'll feel right at home with InstaPy. You'll build a basic bot with it next.

Note: Both Selenium and the Page Object Pattern are widely used for other websites, not just for Instagram.

How to Build an Instagram Bot With InstaPy

In this section, you'll use InstaPy to build an Instagram bot that will automatically like, follow, and comment on different posts. First, you'll need to install InstaPy:

Shell

```
$ python3 -m pip install instapy
```

This will install instapy in your system.

Note: The best practice is to use [virtual environments](#) for every project so that the dependencies are isolated.

**A Peer-to-Peer Learning Community for
Python Enthusiasts...Just Like You**

pythonistacafe.com

 PYTHONISTACAFE



 [Remove ads](#)

Essential Features

Now you can rewrite the code above with InstaPy so that you can compare the two options. First, create another Python file and put the following code in it:

Python

```
from instapy import InstaPy

InstaPy(username="<your_username>", password="<your_password>").login()
```

Replace the username and password with yours, run the script, and voilà! With just **one line of code**, you achieved **the same result**.

Even though your results are the same, you can see that the behavior isn't exactly the same. In addition to simply logging in to your profile, InstaPy does some other things, such as checking your internet connection and the status of the Instagram servers. This can be observed directly on the browser or in the logs:

Shell

```
INFO [2019-12-17 22:03:19] [username] -- Connection Checklist [1/3] (Internet Connection Status)
INFO [2019-12-17 22:03:20] [username] - Internet Connection Status: ok
INFO [2019-12-17 22:03:20] [username] - Current IP is "17.283.46.379" and it's from "Germany/DE"
INFO [2019-12-17 22:03:20] [username] -- Connection Checklist [2/3] (Instagram Server Status)
INFO [2019-12-17 22:03:26] [username] - Instagram WebSite Status: Currently Up
```

Pretty good for one line of code, isn't it? Now it's time to make the script do more interesting things than just logging in.

For the purpose of this example, assume that your profile is all about cars, and that your bot is intended to interact with the profiles of people who are also interested in cars.

First, you can like some posts that are tagged #bmw or #mercedes using `like_by_tags()`:

Python

```
1 from instapy import InstaPy
2
3 session = InstaPy(username="<your_username>", password="<your_password>")
4 session.login()
5 session.like_by_tags(["bmw", "mercedes"], amount=5)
```

Here, you gave the method a list of tags to like and the number of posts to like for each given tag. In this case, you instructed it to like ten posts, five for each of the two tags. But take a look at what happens after you run the script:

Shell

```
INFO [2019-12-17 22:15:58] [username] Tag [1/2]
INFO [2019-12-17 22:15:58] [username] --> b'bmw'
INFO [2019-12-17 22:16:07] [username] desired amount: 14 | top posts [disabled]: 9 | possible p
INFO [2019-12-17 22:16:13] [username] Like# [1/14]
INFO [2019-12-17 22:16:13] [username] https://www.instagram.com/p/B6MCcGcC3tU/
INFO [2019-12-17 22:16:15] [username] Image from: b'mattyproduction'
INFO [2019-12-17 22:16:15] [username] Link: b'https://www.instagram.com/p/B6MCcGcC3tU/'
INFO [2019-12-17 22:16:15] [username] Description: b'Mal etwas anderes \xf0\x9f\x91\x80\xe2\x98\xba
INFO [2019-12-17 22:16:15] [username] Location: b'K\xc3\xb6ln, Germany'
INFO [2019-12-17 22:16:51] [username] --> Image Liked!
INFO [2019-12-17 22:16:56] [username] --> Not commented
INFO [2019-12-17 22:16:57] [username] --> Not following
INFO [2019-12-17 22:16:58] [username] Like# [2/14]
INFO [2019-12-17 22:16:58] [username] https://www.instagram.com/p/B6MDK1wJ-Kb/
INFO [2019-12-17 22:17:01] [username] Image from: b'davs0'
INFO [2019-12-17 22:17:01] [username] Link: b'https://www.instagram.com/p/B6MDK1wJ-Kb/'
INFO [2019-12-17 22:17:01] [username] Description: b'Someone said cloud? \xf0\x9f\xa4\x94\xf0\x9f\x
INFO [2019-12-17 22:17:34] [username] --> Image Liked!
INFO [2019-12-17 22:17:37] [username] --> Not commented
INFO [2019-12-17 22:17:38] [username] --> Not following
```

By default, InstaPy will like the first nine top posts in addition to your amount value. In this case, that brings the total number of likes per tag to fourteen (nine top posts plus the five you specified in amount).

Also note that InstaPy logs every action it takes. As you can see above, it mentions which post it liked as well as its link, description, location, and whether the bot commented on the post or followed the author.

You may have noticed that there are delays after almost every action. That's by design. It prevents your profile from getting banned on Instagram.

Now, you probably don't want your bot liking inappropriate posts. To prevent that from happening, you can use `set_dont_like()`:

Python

```
from instapy import InstaPy

session = InstaPy(username="<your_username>", password="<your_password>")
session.login()
session.like_by_tags(["bmw", "mercedes"], amount=5)
session.set_dont_like(["naked", "nsfw"])
```

With this change, posts that have the words naked or nsfw in their descriptions won't be liked. You can flag any other words that you want your bot to avoid.

Next, you can tell the bot to not only like the posts but also to follow some of the authors of those posts. You can do that with `set_do_follow()`:

Python

```
from instapy import InstaPy

session = InstaPy(username="<your_username>", password="<your_password>")
session.login()
session.like_by_tags(["bmw", "mercedes"], amount=5)
session.set_dont_like(["naked", "nsfw"])
session.set_do_follow(True, percentage=50)
```

If you run the script now, then the bot will follow fifty percent of the users whose posts it liked. As usual, every action will be logged.

You can also leave some comments on the posts. There are two things that you need to do. First, enable commenting with `set_do_comment()`:

Python

```
from instapy import InstaPy

session = InstaPy(username="<your_username>", password="<your_password>")
session.login()
session.like_by_tags(["bmw", "mercedes"], amount=5)
session.set_dont_like(["naked", "nsfw"])
session.set_do_follow(True, percentage=50)
session.set_do_comment(True, percentage=50)
```

Next, tell the bot what comments to leave with `set_comments()`:

Python

```
from instapy import InstaPy

session = InstaPy(username="<your_username>", password="<your_password>")
session.login()
session.like_by_tags(["bmw", "mercedes"], amount=5)
session.set_dont_like(["naked", "nsfw"])
session.set_do_follow(True, percentage=50)
session.set_do_comment(True, percentage=50)
session.set_comments(["Nice!", "Sweet!", "Beautiful :heart_eyes:"])
```

Run the script and the bot will leave one of those three comments on half the posts that it interacts with.

Now that you're done with the basic settings, it's a good idea to end the session with `end()`:

Python

```
from instapy import InstaPy

session = InstaPy(username="<your\_username>", password="<your\_password>")
session.login()
session.like_by_tags(["bmw", "mercedes"], amount=5)
session.set_dont_like(["naked", "nsfw"])
session.set_do_follow(True, percentage=50)
session.set_do_comment(True, percentage=50)
session.set_comments(["Nice!", "Sweet!", "Beautiful :heart\_eyes:"])
session.end()
```

This will close the browser, save the logs, and prepare a report that you can see in the console output.

A Peer-to-Peer Learning Community for Python Enthusiasts...Just Like You

[pythonistacafe.com](#)



[Remove ads](#)

Additional Features in InstaPy

InstaPy is a sizable project that has a lot of [thoroughly documented features](#). The good news is that if you're feeling comfortable with the features you used above, then the rest should feel pretty similar. This section will outline some of the more useful features of InstaPy.

Quota Supervisor

You can't scrape Instagram all day, every day. The service will quickly notice that you're running a bot and will ban some of its actions. That's why it's a good idea to set quotas on some of your bot's actions. Take the following for example:

Python

```
session.set_quota_supervisor(enabled=True, peak_comments_daily=240, peak_comments_hourly=21)
```

The bot will keep commenting until it reaches its hourly and daily limits. It will resume commenting after the quota period has passed.

Headless Browser

This feature allows you to run your bot without the GUI of the browser. This is super useful if you want to deploy your bot to a server where you may not have or need the graphical interface. It's also less CPU intensive, so it improves performance. You can use it like so:

Python

```
session = InstaPy(username='test', password='test', headless_browser=True)
```

Note that you set this flag when you initialize the InstaPy object.

Using AI to Analyze Posts

Earlier you saw how to ignore posts that contain inappropriate words in their descriptions. What if the description is good but the image itself is inappropriate? You can integrate your InstaPy bot with [ClarifAI](#), which offers image and video recognition services:

Python

```
session.set_use_clarifai(enabled=True, api_key='<your\_api\_key>')
session.clarifai_check_img_for(["nsfw"])
```

Now your bot won't like or comment on any image that ClarifAI considers [NSFW](#). You get 5,000 free API-calls per month.