



Python Ping: How to ping with Python (super-easy)

Alessandro Maggio(<https://www.ictshore.com/author/alessandro-maggio/>)

August 23, 2018

Share This Post



Any of us needs to ping at some point in his life. In fact, it doesn't matter if you are a network engineer or a programmer – anyone in the IT world knows the ping. This simple command allows you to check if a remote device is alive, and it offers some other amenities. Since its task is simple, we expect ping to be simple. However, Python does not support natively a quick way to ping, so doing it can become a real pain. It is time to change: with **Python Ping**, you can have your python script pinging in seconds.

Python Ping



Python Ping (pythonping), an easy way to ping in Python.

Installing Python Ping

Python Ping (`pythonping`) is a public repository you can find on PyPI. We have released it with the MIT license, so anyone can use it. Since the repository is on PyPI (<https://pypi.python.org/pypi>), you can quickly install it with `pip`.

```
pip install pythonping  (https://www.ictshore.com)
```

Done, we are ready to ping! Just read on...

Just Ping!

```
1. from pythonping import ping
2.
3.
4. ping('8.8.8.8')
```

This code doesn't need much explanation. With it, we simply ping Google. However, you won't see anything in your console if you just run this script. This is because our ping is **silent by default**, and does not print anything to screen. Why? Because **it returns the results** instead. If we want to see everything on-screen, we can simply use the **verbose** flag.

```
1. ping('8.8.8.8', verbose=True)
```

This will print to screen something like this:

```
Reply from 8.8.8.8, 9 bytes in 8.17ms
Reply from 8.8.8.8, 9 bytes in 7.14ms
Reply from 8.8.8.8, 9 bytes in 8.12ms
Reply from 8.8.8.8, 9 bytes in 8.12ms
```

Working with the return values

Printing on-screen is not what we want all the time. This is the reason that makes our script silent by default. Instead, we want other parts of our program to work with the results of the ping. For example, if the remote device is not available we might need to run a script or send an email to the administrator. We might want to prepare custom actions if the latency exceeds a threshold, and so on.

Because of that, our Python Ping returns all the details you may possibly need, in a custom object: **ResponseList**. This object is an iterable containing other custom objects, instances of **Response**. Each **Response** represents the response received from a given ICMP request. It contains its payload (the message), and the time it took to receive it.

Since a **Response** is an object, you can get its properties from its members.

- **error_message** contains a string describing the error this response represents. For example, an error could be "Network Unreachable" or "Fragmentation Required". If you got a successful response, this property is **None**.
- **success** is a bool indicating if the response is successful
- **time_elapsed** and **time_elapsed_ms** indicate how long it took to receive this response, respectively in seconds and milliseconds.

You can access individual responses by accessing the **_responses** property of your **ResponseList** object, returned from **ping()**. In this case, **_responses** is simply a list. However, **_responses** are not meant to be accessed from outside the **ResponseList**, you should work with **ResponseList** directly. On top of that, **ResponseList** adds some intelligence you can access from its own members. The fields are self-explanatory:

- **rtt_min** and **rtt_min_ms**
- **rtt_max** and **rtt_max_ms**
- **rtt_avg** and **rtt_avg_ms**

Clarify with an example

All of this **ResponseList** and **Response** concept may seem complex at first. Trust me, it isn't, and we can see that with an example. First, we need to store our results in an object.

```
1. from pythonping import ping
2. response_list = ping('8.8.8.8', size=40, count=10)
```

With this ping, we sent 10 ICMP packets with a payload of 40 bytes to Google. To see our average RTT, we can print the **rtt_avg_ms** from the **response_list** object.

```
1. print(response_list.rtt_avg_ms)
```

That, in our case, yields this:



And this is it! Simple enough, uh?

Advanced Python Ping

All the parameters of “ping”

Ping is awesome, and simply using what we described above can save you a lot of time. However, you may need to ping in a more sophisticated way from time to time. This is why we packed our Python Ping with features while keeping it simple. In fact, the `ping()` function expect many parameters besides the target host, that you may want to specify. Here they are.

Parameter	Type	Description
<code>target</code>	<code>str</code>	The remote device to ping. This is the only mandatory parameter.
<code>timeout</code>	<code>int</code>	How long before considering a non-received response lost, in seconds.
<code>count</code>	<code>int</code>	How many ICMP packets to send in sequence.
<code>size</code>	<code>int</code>	The size of the entire ICMP packet we want to send. You can leave it to 0, the default value, to adapt the size to the payload and not the other way around.
<code>payload</code>	<code>str</code> or <code>bytes</code>	The payload of the packet. If you provide also the size of the payload, this will be cut or repeated to match the desired size.
<code>sweep_start</code>	<code>int</code>	To be used together with <code>sweep_end</code> . It ignores the <code>size</code> and sends each subsequent packet by incrementing payload size by one byte, starting from <code>sweep_start</code> size all the way to <code>sweep_end</code> size. Useful when trying to find MTU.
<code>sweep_end</code>	<code>int</code>	When doing a ping sweep (with <code>sweep_start</code>), maximum payload size to reach.
<code>df</code>	<code>bool</code>	Value for the Don't Fragment flag of the IP header.
<code>verbose</code>	<code>bool</code>	True if you wish to write output to the screen.
<code>out</code>	<code>file</code>	Where to direct the output, by default <code>sys.stdout</code> .

PythonPing Parameters

Customizing the ping

According to the details above, we can customize and tune the ping to perfectly meet our needs. Below, some common examples. Many of you may be familiar with them, as you might have used them in the system ping. With Python Ping, you can do the same in Python.

Stress Test

With this example, you can ensure that the link can handle the load effectively. You send many large packets and see what happens.

```
1. ping('8.8.8.8', count=10000, size=1500)
```

Since we did not specify the payload, it will be random.

Custom payload

In developing an advanced application, you may want to use a custom payload. For example, you can implement a responder to verify the time the packet spent on the network, and the time the packet was processed inside the end devices. In such an example, you may need to use a timestamp

as payload. We could do something like this.



(<https://www.ictshore.com>)

```
1. import time
2.
3. ping('8.8.8.8', payload=int(round(time.time())).to_bytes(10, 'little'))
4.
5. # Use this instead for Python prior to 3.8
6. ping('8.8.8.8', count=1, payload=bytes(time.clock()))
```

Check MTU with Ping Sweep

To verify how big a packet can be, you can do a ping sweep. This way you will identify the size that will make the remote device stop responding. To do that, we need to start from a relatively small size and go up to a large one.

```
1. ping('8.8.8.8', sweep_start=100, sweep_end=1550, df=True)
```

Note that here we absolutely need the `df` flag. In fact, if we don't use it, routers in the path may fragment the packet so we won't be able to see the actual MTU.

Conclusion (and link to source!)

So now, with Python Ping, pinging in Python is extremely easy. I hope you can find this module helpful and save time, focusing on what you actually need to do. I suggest you to check two key resources:

- pythonping project on GitHub (<https://github.com/alessandromaggio/pythonping>)
- pythonping on PyPi (<https://pypi.python.org/pypi/pythonping>)

Now it is time for you to go out and ping the world! And as always, let me know your thoughts in the comments, and let me know how you are going to use pythonping.

◀ (<https://www.ictshore.com/quick-and-easy/ping>) ▶ (<https://www.ictshore.com/advanced-networking>)

Never Stop Learning...

Be a Social Media Influencer in 3 Easy Steps

(<https://www.ictshore.com/business/social-media-influencer/>)


The 1 Best Datacenter Network Design to Know

(<https://www.ictshore.com/networking/datacenter-network-design/>)



(<https://alessandromaggio.com>)

Alessandro Maggio

(<https://alessandromaggio.com>)
Project manager.  <https://www.python.org/> about networking & coding. I believe that time is the most precious resource we have, and that technology can help us not to waste it. I founded ICTShore.com with the same principle: I share what I learn so that you get value from it faster than I did.

14 Responses



ValentinV says:

September 7, 2018 at 8:51 am (<https://www.ictshore.com/python/python-ping-tutorial/#comment-110>)

Hello Alessandro,

Thank you for sharing this project. I intend to use PythonPing in my work but I am an absolute beginner with Python.

I tried to check the MTU but I get the following error:

```
>>> ping('8.8.8.8', sweep_start=100, sweep_end=1550, df=True)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/val/tutorial_env/lib/python3.7/site-packages/pythonping/__init__.py", line 52, in ping
    comm = executor.Communicator(target, provider, timeout, socket_options=options, verbose=verbose, output=out)
  File "/home/val/tutorial_env/lib/python3.7/site-packages/pythonping/executor.py", line 193, in __init__
    self.socket = network.Socket(target, 'icmp', source=None, options=socket_options)
  File "/home/val/tutorial_env/lib/python3.7/site-packages/pythonping/network.py", line 27, in __init__
    self.socket.setsockopt(options)
TypeError: setsockopt() takes exactly 3 arguments (1 given)
```

I am using Python 3.7.0

Maybe you can help.

Regards,

Valentin



Alessandro Maggio (<http://alessandromaggio.com>) says:

September 7, 2018 at 9:26 am (<https://www.ictshore.com/python/python-ping-tutorial/#comment-112>)

Hello Valentin,

Thank you so much for this comment! It looks like a bug, due to a misconfiguration inside pythonping. I've had a quick look into the code and found a likely issue, I'll publish a patch very soon. For now, I can tell you it is related to the don't fragment flag, and besides this feature everything works like a charm.

I'll keep you posted, and thanks again for experimenting with Python Ping 😊



Alessandro Maggio (<http://alessandromaggio.com>) says:

September 8, 2018 at 2:30 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-113>)

PythonPing version 1.0.2 is out now! Install it with *pip install pythonping --upgrade*.

The program won't generate an exception anymore, but apparently the DF flag is still not being set on Windows. I'll need to investigate why, as raw socket programming is somewhat nasty. Nonetheless, I'll be tracking the bug on GitHub here (<https://github.com/alessandromaggio/pythonping/issues/7>). 😊



ValentinV says:

September 13, 2018 at 9:47 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-114>)

Thanks for your support Alessandro. I hope I will get some spare time during the weekend to test. If it makes any difference, I am using an Arch Linux machine.




ValentinV says:

September 15, 2018 at 7:50 am (<https://www.ictshore.com/python/python-ping-tutorial/#comment-115>)

Hi Alessandro,

Here it is:

 (https://www.ictshore.com)
>>> ping('8.8.8.8', sweep_start=100, sweep_end=1550, ur=True)
Reply from 8.8.8.8, 9 bytes in 0.18ms
Reply from 8.8.8.8, 9 bytes in 0.1ms
Reply from 8.8.8.8, 9 bytes in 0.1ms
Reply from 8.8.8.8, 9 bytes in 0.09ms

Round Trip Times min/avg/max is 0.09/0.12/0.18 ms

I am not sure that this is the output we are supposed to get from this command though.



Alessandro Maggio (<http://alessandromaggio.com>) says:

September 15, 2018 at 5:49 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-116>)

Yes, the output is the same for all commands at the moment. I am also considering adding a cisco-like output with exclamation marks and dots.



bhavik says:

March 7, 2019 at 11:15 am (<https://www.ictshore.com/python/python-ping-tutorial/#comment-125>)

Hi, Alessandro

Traceback (most recent call last):

File "/plugin.py", line 9, in <module>

from pythonping import ping

File "/venv/local/lib/python2.7/site-packages/pythonping/__init__.py", line 2, in <module>

from . import network, executor, payload_provider

File "/venv/local/lib/python2.7/site-packages/pythonping/executor.py", line 159

print(value, file=self.output)

^

SyntaxError: invalid syntax



Alessandro Maggio (<http://alessandromaggio.com>) says:

March 7, 2019 at 7:52 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-128>)

Hello bhavik,

Thank for reaching out. I see you are using a Python 2.7 virtual environment (from File "/venv/local/lib/python2.7/"), but Python Ping supports only Python 3.

Try using Python 3.x and let me know! I would also consider switching your entire project to Python 3, as Python 2.7 will be retired in 2020.



mschormann says:

March 26, 2019 at 10:40 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-134>)

Hello Alessandro,

When I try to iterate over the object ResponseList to get at the individual Responses, I get an error message saying that ResponseList is not an iterable object. Could you please provide a code example of how to use the values in the Response objects that make up ResponseList.

Thanks



Alessandro Maggio (<http://alessandromaggio.com>) says:

March 29, 2019 at 5:09 am (<https://www.ictshore.com/python/python-ping-tutorial/#comment-135>)

Hey Mschormann,

Absolutely. Which python version and pythonping version are you using?




Bikrant50 says:

July 12, 2019 at 3:46 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-156>)

Hi Alessandro,

I'm a beginner with Python so bear with me, but can this be used to with the internal IP address of another PC on the same network? I.e. `ping('10.x.x.x', verbose = True)`? I am currently trying to do and getting "Request timed out".

Thank you,

 **Alessandro Maggio** (<http://alessandromaggio.com>) says:

July 13, 2019 at 8:42 am (<https://www.ictshore.com/python/python-ping-tutorial/#comment-157>)

Hello Bikran50!

Yes! You can use an IP or hostname you want. If you get request timeout, it is probably because you don't have network connectivity to that PC.


 **Bikrant50** says:

July 18, 2019 at 5:55 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-164>)

Could you elaborate on how the 'out' parameter works? If I want to output to a .csv file would

`ping('8.8.8.8',out = 'file.csv')`

work?

 **Alessandro Maggio** (<http://alessandromaggio.com>) says:

July 19, 2019 at 4:28 pm (<https://www.ictshore.com/python/python-ping-tutorial/#comment-166>)

The output is a file stream, if you don't want the output to be written in the terminal. However, it just writes text, it does not populate a CSV file.

Comments are closed.

Join the Newsletter to Get Ahead

Revolutionary tips to get ahead with technology directly in your Inbox.

Join the club →

Want Visibility from Tech Profes

If you feel like sharing your knowledge, we are open to guest posting - and it's



ACCESSIBILITY

[Login](#)

[Register](#)

[Password lost?](#)

[ICTShore.com](#)

ABOUT

[ICTShore.com](#)

[About](#)

[Guest Posting](#)

FREE COURSES

[CCNA Free Course](#)

[Full Stack Free Course](#)

© 2021 Rational Scale Srl unico socio

[Terms & Conditions](#)

[Privacy Policy](#)