

# Changing the "tick frequency" on x or y axis in matplotlib?

[Ask Question](#)

I am trying to fix how python plots my data.

280 Say

```
x = [0,5,9,10,15]
```

and

91 

```
y = [0,1,2,3,4]
```

Then I would do:

```
matplotlib.pyplot.plot(x,y)
matplotlib.pyplot.show()
```

and the x axis' ticks are plotted in intervals of 5. Is there a way to make it show intervals of 1?

[python](#)[matplotlib](#)[share](#)[improve this question](#)[edited Sep 27 '12 at 12:43](#)[Daan](#)[1,954](#) 11 20[asked Sep 26 '12 at 19:12](#)[Dax Feliz](#)[2,363](#) 7 15 21

4 Though ticks is the appropriate word here, change ticks to step size will definitely guide more newbies to this question. – [Sibbs Gambling](#) Nov 18 '13 at 16:01

7 Closely related question: [stackoverflow.com/questions/6682784/...](http://stackoverflow.com/questions/6682784/...) and a great solution: `pyplot.locator_params(nbins=4)` – [Jan-Philip Gehrcke](#) Apr 8 '14 at 11:49

`nbins` seems to have been deprecated in `matplotlib2.x`, unfortunately – [jeremy\\_rutman](#) Mar 8 at 8:40

[add a comment](#)

9 Answers

[active](#)[oldest](#)[votes](#)

You could explicitly set where you want to tick marks with `plt.xticks` :

338

```
plt.xticks(np.arange(min(x), max(x)+1, 1.0))
```

For example,

```
import numpy as np
import matplotlib.pyplot as plt

x = [0,5,9,10,15]
y = [0,1,2,3,4]
plt.plot(x,y)
plt.xticks(np.arange(min(x), max(x)+1, 1.0))
plt.show()
```

( `np.arange` was used rather than Python's `range` function just in case `min(x)` and `max(x)` are floats instead of ints.)

The `plt.plot` (or `ax.plot` ) function will automatically set default `x` and `y` limits. If you wish to keep those limits, and just change the stepsize of the tick marks, then you could use `ax.get_xlim()` to discover what limits Matplotlib has already set.

```
start, end = ax.get_xlim()
ax.xaxis.set_ticks(np.arange(start, end, steps
```

The default tick formatter should do a decent job rounding the tick values to a sensible number of significant digits. However, if you wish to have more control over the format, you can define your own formatter. For example,

```
ax.xaxis.set_major_formatter(ticker.FormatStrF
```

Here's a runnable example:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

x = [0,5,9,10,15]
y = [0,1,2,3,4]
fig, ax = plt.subplots()
ax.plot(x,y)
start, end = ax.get_xlim()
ax.xaxis.set_ticks(np.arange(start, end, 0.712
ax.xaxis.set_major_formatter(ticker.FormatStrF
plt.show()
```

share

improve this answer

edited Oct 1 '13 at 17:50

answered Sep 26 '12 at 19:24



unutbu

519k 94 1089 1177

- 
- 41 Is there no way to get it to still decide it's own limits, but just change the step size? This method is not very good if the min is something like 3523.232512! – [Corone](#) Oct 1 '13 at 16:41
- 
- 2 @Corone, It has been a while since you asked, but I have posted an answer below that allows for easy control of step size while still using automatic bounds determination. – [jthomas](#) Mar 26 '16 at 13:04
- 
- 1 Note that the `+1` in `plt.xticks(np.arange(min(x), max(x)+1, 1.0))` is required to show the last tick mark. – [Alex Willison](#) May 19 '17 at 15:23
- 
- 1 Yes, `np.arange(start, stop)` generates values in the *half-open* interval `[start, stop)`, including `start` but excluding `stop`. So I used `max(x)+1` to ensure that `max(x)` is included. – [unutbu](#) May 19 '17 at 16:19
- 
- 2 is there an equivalent for datetime e.g. `plt.xticks(np.arange(min(dates), max(dates)+0.1, 0.1))` ? it seems to only plots the year – [WBM](#) Jan 10 at 11:42
- 

[show 2 more comments](#)

Another approach is to set the axis locator:

133

```
import matplotlib.ticker as plticker

loc = plticker.MultipleLocator(base=1.0) # thi
ax.xaxis.set_major_locator(loc)
```

There are several different types of locator depending upon your needs.

share

improve this answer


answered Nov 14 '13 at 8:38



robochat

1,712 1 9 12

- 
- 3 This does not work as expected. Specifically, when using dates, it does not use the appropriate dates. – [Chris Fannesbeck](#) Feb 5 '14 at 17:02
-

- 21 When using dates, you should use the methods in the `matplotlib.dates` module. For example `matplotlib.dates.AutoDateLocator()`  
– [robochat](#) Mar 20 '14 at 13:06 
- 5 This should be the accepted solution. – [onewhaleid](#) Mar 17 '16 at 21:19
- 2 It worked as expected for me, with dates. This solution is much easier than the accepted one. – [Pablo Suau](#) Jul 8 '16 at 13:58

[add a comment](#)

I like this solution (from the [Matplotlib Plotting Cookbook](#)):

67

```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

x = [0,5,9,10,15]
y = [0,1,2,3,4]

tick_spacing = 1

fig, ax = plt.subplots(1,1)
ax.plot(x,y)
ax.xaxis.set_major_locator(ticker.MultipleLoca
plt.show()
```

This solution give you explicit control of the tick spacing via the number given to `ticker.MultipleLocator()`, allows automatic limit determination, and is easy to read later.

[share](#)

[improve this answer](#)

answered Mar 25 '16 at 23:24



[jthomas](#)

957 7 13

- 2 A way to do this without calculating the ticks explicitly! – [Zelphir](#) Aug 31 '16 at 9:48
- 1 This is the same answer as [this one](#). It does not make sense to add an identical answer two years later. – [ImportanceOfBeingErnest](#) Jun 23 '17 at 10:41
- 4 Good catch. I did not recognize them as the same when I posted the answer. Still, I think this presentation is a little easier to understand. – [jthomas](#) Jun 23 '17 at 15:17

The book reference in this answer also provide a helpful source for more information. – [Steven C. Howell](#) Apr 16 at 19:50

[add a comment](#)

40

In case anyone is interested in a general one-liner, simply get the current ticks and use it to set the new ticks by sampling every other tick.

```
ax.set_xticks(ax.get_xticks()[::2])
```

share

improve this answer



glopes

1,151 1 12 21

add a comment

answered Apr 15 '16 at 11:46

23

This is a bit hacky, but by far the cleanest/easiest to understand example that I've found to do this. It's from an answer on SO here:

[Cleanest way to hide every nth tick label in matplotlib colorbar?](#)

```
for label in ax.get_xticklabels()[::2]:
    label.set_visible(False)
```

Then you can loop over the labels setting them to visible or not depending on the density you want.

edit: note that sometimes matplotlib sets labels == '', so it might look like a label is not present, when in fact it is and just isn't displaying anything. To make sure you're looping through actual visible labels, you could try:

```
visible_labels = [lab for lab in ax.get_xtickl
plt.setp(visible_labels[::2], visible=False)
```

share

improve this answer



Community ♦

1 1

edited May 23 '17 at 12:34

answered Aug 13 '15 at 20:15



choldgraf

936 2 10 21

- 2 This is the most simple and generic solution. A tiny adjustment: usually `ax.get_xticklabels()[1::2]` are the labels to be hidden. – [jolvi](#) Sep 22 '15 at 12:02

This doesn't work with `matplotlib.finance.candlestick2` – [BCR](#) Feb 12 '16 at 16:12

@BCR it could be that some of the xticklabels are just set to '' so that when you loop through them, you're making xticklabels that are empty invisible (which would

have no effect on the visualization, but might mean that you aren't pulling the correct labels). You could try:

```
vis_labels = [label for label in ax.get_xticklabels() if label.get_visible() is True];
plt.setp(vis_labels[::2], visible==False)
```

– [choldgraf](#) Feb 15 '16 at 19:57

[add a comment](#)

This is an old topic, but I stumble over this every now and then and made this function. It's very convenient:

10

```
import matplotlib.pyplot as pp
import numpy as np

def resadjust(ax, xres=None, yres=None):
    """
    Send in an axis and I fix the resolution a
    """

    if xres:
        start, stop = ax.get_xlim()
        ticks = np.arange(start, stop + xres,
                           xres)
        ax.set_xticks(ticks)

    if yres:
        start, stop = ax.get_ylim()
        ticks = np.arange(start, stop + yres,
                           yres)
        ax.set_yticks(ticks)
```

One caveat of controlling the ticks like this is that one does no longer enjoy the interactive automagic updating of max scale after an added line. Then do

```
gca().set_ylim(top=new_top) # for example
```

and run the resadjust function again.

[share](#)

[improve this answer](#)

[edited Apr 2 '15 at 10:04](#)

[answered Dec 17 '14 at 19:18](#)



[Tompa](#)

2,071 1 8 12

[add a comment](#)

I developed an inelegant solution. Consider that we have the X axis and also a list of labels for each point in X.

7

Example:

```
import matplotlib.pyplot as plt

x = [0,1,2,3,4,5]
y = [10,20,15,18,7,19]
xlabels = ['jan', 'feb', 'mar', 'apr', 'may', 'jun']
```

Let's say that I want to show ticks labels only for 'feb' and 'jun'

```
xlabelsnew = []
for i in xlabels:
    if i not in ['feb', 'jun']:
        i = ' '
        xlabelsnew.append(i)
    else:
        xlabelsnew.append(i)
```

Good, now we have a fake list of labels. First, we plotted the original version.

```
plt.plot(x,y)
plt.xticks(range(0,len(x)),xlabels,rotation=45)
plt.show()
```

Now, the modified version.

```
plt.plot(x,y)
plt.xticks(range(0,len(x)),xlabelsnew,rotation=45)
plt.show()
```

share

improve this answer

edited May 12 '15 at 0:55

answered Mar 11 '15 at 13:26



Deninhos

331 4 4

add a comment

1

```
xmarks=[i for i in range(1,length+1,1)]

plt.xticks(xmarks)
```

This worked for me

if you want ticks between [1,5] (1 and 5 inclusive) then replace

```
length = 5
```

share

improve this answer

edited Jun 30 '17 at 7:56



BartoszKP



26.1k 10 62 99

answered Jun 30 '17 at 7:33



Piyush Gupta

81 1 4

fyi, you could simply write `xmarks = range(1, length+1, 1)` . pretty sure the list comprehension is redundant. – Neal Jul 21 '17 at 13:30

Thanks. Tried many ways but found this to do what exactly I was looking for – Megha Nov 3 '17 at 16:50

[add a comment](#)

Here's a pure python implementation of the desired functionality that handles any numeric series (int or float) with positive, negative, or mixed values:

```
def computeTicks (x, step = 5):
    """
    Computes domain with given step encompassi
    @ params
    x - Required - A list-like object of in
    step - Optional - Tick frequency
    """
    import math as Math
    xMax, xMin = Math.ceil(max(x)), Math.floor
    dMax, dMin = xMax + abs((xMax % step) - st
    return range(dMin, dMax, step)
```

Sample Output:

```
# Negative to Positive
series = [-2, 18, 24, 29, 43]
print(list(computeTicks(series)))

[-5, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45]

# Negative to 0
series = [-30, -14, -10, -9, -3, 0]
print(list(computeTicks(series)))

[-30, -25, -20, -15, -10, -5, 0]

# 0 to Positive
series = [19, 23, 24, 27]
print(list(computeTicks(series)))

[15, 20, 25, 30]

# Floats
series = [1.8, 12.0, 21.2]
print(list(computeTicks(series)))
```



```
[0, 5, 10, 15, 20, 25]

# Step - 100
series = [118.3, 293.2, 768.1]
print(list(computeTicks(series, step = 100)))

[100, 200, 300, 400, 500, 600, 700, 800]
```

And Sample Usage:

```
import matplotlib.pyplot as plt

x = [0,5,9,10,15]
y = [0,1,2,3,4]
plt.plot(x,y)
plt.xticks(computeTicks(x))
plt.show()
```