U1BP42MRS : You can go through the python tutorial, otherwise I just look at stuff as they come up.
<https://docs.python.org/3/tutorial/index.html>
Later when you know Python better, Effective Python is a good book - but again, after you know the language and can appreciate the tips

U5CGPBF0U : Exactly how much water should be in the paddy for my summer crop of rice? I need agricultural advice.
U60KNBMPX : and getting stackoverflow working for you is also a skill that should be honed.
U60KNBMPX : `from stackoverflow import fibonacci`
U6BEWEP7Z : if you want to understand OOP, I would recommend learning Java. Java is a statically typed language, but all of it is object oriented and must be inside of a class.
U5JF1KD18 : Has any one here ever tried to implement a SDN using python ?
U1UFZTD4J : <@U5JF1KD18> sdn?
U07JGLLKF : Software Defined Networking
U07JGLLKF : <https://en.wikipedia.org/wiki/Software-defined_networking>
U5GJVTRGB : Hi.I have a time in a string format as
`2017-07-21T02:16:51.449-07:00`

I am seeking this to convert it into unix-timestamp.
Anybody have any idea ?

U5LNXQHN3 : ```$ pip install arrow
... then, in Python ...
t = arrow.get('2017-07-21T02:16:51.449-07:00')
print(t.timestamp)
```

U5LNXQHN3 : If that doesn't work, you can use `arrow.get` with a specific format string
U5LNXQHN3 : I recommend `arrow` over standard library functionality.
U5GJVTRGB : <@U5LNXQHN3> : Thanks for response!that worked :slightly_smiling_face:
i hope there's no problem in using that...

U0LSCQQNR : shouldn't be
U0LSCQQNR : since its intended to be as close as a drop in replacement
U5GJVTRGB : Also it does consider `07:00`
from the time, right ? which i think is the timezone, right ?

U5LNXQHN3 : specifically it's -7:00, and yes
U0LSCQQNR : so, java had a really crappy time/calendar library in the standard library for years before it pulled in `joda-time` into the core API
U0LSCQQNR : now, if only python could do something similar...
U5LNXQHN3 : I think the success of pypi has meant that standard library development has stagnated somewhat, which is a shame
U5LNXQHN3 : we have a poor selection of date/time libraries, and nothing much of use for multimedia generally. But hey, we have `dbm` for "databases", `nntplib` for all the newsgroups we need to access regularly, and `sunau` for reading ancient audio files, so it's not all bad
U5NMSURAQ : is this sarcasm? :disappointed:
U5LNXQHN3 : just a little
U5NMSURAQ : sqlite and shelve / dbm modules aren't bad
U5NMSURAQ : they are very much OK
U5LNXQHN3 : sqlite is great
U5LNXQHN3 : But I do think Python would be better if there was more effort spent on getting good packages into the standard library
U5NMSURAQ : yeah.. maybe.. but it's not that easy
U5NMSURAQ : say you want a better datetime. what do you pick?
U5NMSURAQ : Arrow? Why not Delorean?
U5LNXQHN3 : That's not the sort of thing you need to answer immediately in a chat... you'd get a few knowledgeable people, they'd discuss it for a bit, find use cases, check code quality, etc
U5NMSURAQ : There's going to be a huge debase, because it's not like it's clear which of the two is massively better.
U5LNXQHN3 : Doesn't matter. This happens already for other PEPs. There's a discussion, then a decision is made
U5NMSURAQ : But PyPI provides an illusion of a democracy, where you can chose whatever you want.
```

U5NMSURAQ : Python and PEPs isn't a democracy at all
U5NMSURAQ : so maybe it's a calculated decision -- to not introduce some things into stdlib
U5LNXQHN3 : PyPI wouldn't go away. Users are always free to reinvent the wheel
U5LNXQHN3 : And yeah, it probably is a calculated decision; one I disagree with
U5LNXQHN3 : If you're gonna say that batteries are included, they should be good batteries, ones suitable for powering modern software
U28MDQRL2 : Reasons for importing something inside of a function?. Besides avoiding circular imports
U5LNXQHN3 : Maybe it's not relevant to anything else in the program
U5LNXQHN3 : Generally speaking I avoid it if possible
U28MDQRL2 : I have just done it for avoiding circular imports on some celery tasks. But i have seem it on open source projects
U0LSCQQNR : it you're trying to limit the number of imports, I can understand
U0LSCQQNR : especially highly specific imports
U1NSCAY6R : It also violates PEP8
U0LSCQQNR : but isn't there a bit of a penalty in doing so?
U0LSCQQNR : performance penalty, that is
U0LSCQQNR : because that method gets hit
U0LSCQQNR : has to import a dependency before continuing execution
U28MDQRL2 : I read that imports are expensives
U5LNXQHN3 : they can be
U1NSCAY6R : and some people leave the more expensive ones in functions that arent often called, thats a main reason i see it too
U6944D5GU : hellooo friends
U6944D5GU : anybody knows what this means
U6944D5GU : i saw this code in here:
U0E44UP6G : first link in google by query numpy arrange
U0E44UP6G : <https://docs.scipy.org/doc/numpy/reference/generated/numpy.arange.html>
U0E44UP6G : or did you ask something specific?
U6944D5GU : what about t**2?? <@U0E44UP6G>
U0E44UP6G : t power of 2
U6BD02RQV : but t is an array?
U6944D5GU : how does t**2 and t**3 work in this image <@U0E44UP6G>
U0E44UP6G : may be overloaded `__pow__`
U0E44UP6G : I haven't work with numpy yet :slightly_smiling_face:
U6944D5GU : :eyes:
U5NMSURAQ : the function applies to every element of the original array
U5NMSURAQ : so you see three plots: `y = x^1`, `y = x^2` and `y = x^3`
U5ZPMJA06 : I have here a very small unit test program which is supposed to test that an object throws an AttributeError exception when a nonexistent
attribute lookup is attempted.

```
"""
It works fine, but I had to put the attribute access in a function.
I'd rather not define that function and use the assertRaises as follows:

  self.assertRaises(AttributeError, lambda item: item.donation)

However, this gives me a "TypeError: &lt;lambda&gt;() takes exactly 1 argument (0 given)"

Is this somehow possible without helper function?
"""

import unittest

class Person(object):
    pass


class MyTestCase(unittest.TestCase):
```

```
    def testNonexistentAttibute(self):
        def bombfunc():
            p = Person()
            p.name = "Joe"
            p.money = 2800
            p.money += p.donation # boom!
        self.assertRaises(AttributeError, bombfunc)
        self.assertRaises(AttributeError, lambda item: item.donation) # How to make this work?


if __name__  == "__main__":
    unittest.main()
```

U5LNXQHN3 : I know you can use a `with` block in some other test packages, but to be honest this seems like a weird thing to be testing for
U5LNXQHN3 : You could probably use `self.assertIn("donation", item.__dict__)` or some similar abomination if you really want a one-liner
U5ZPMJA06 : <@U5LNXQHN3> Yeah it sounds weird, but actually the object under test is some kind of container giving both attribute and keyed lookup access to a set of properties, and I need to test whether the right exceptions are thrown during lookup of a nonexistent property.
U5LNXQHN3 : Sounds like a bad idea to me. But if I had to write tests for it, I'd just use the __dict__ check directly. Or `hasattr`.
U5ZPMJA06 : <@U5LNXQHN3> Bad idea? You have a better idea? Thanks for the `hasattr` tip! This is what it's all about, the `Bunch` object: <https://github.com/motoom/bunch>
U5LNXQHN3 : I think that is unnecessarily blurring the lines between a container and a type. If you don't know what attributes a type has, then you don't really know what interface it provides, which makes it a very awkward object to work with
U5ZPMJA06 : <@U5LNXQHN3> I use it declutter my source code. Basically it is a `dict` like object where you don't have to type `["` and `"]` all the time. So I can write:```
U5ZPMJA06 : ```for r in bunched(recordset): # Where recordset is fetch_all() of DictCursor    if r.salary < 3000:
      print r.name, "could use a raise"
```

U5ZPMJA06 : Psycopg2 has a `NamedTupleCursor`, which provides the same syntax.
```