U673BSS76 : The import on line 6 is circular without case-sensitivity
U47HK8BS8 : Well there is a class Queue in the Queue module, maybe that's messing it up? What happens if you do `from Queue import Queue`?
U673BSS76 : The issue was, that small piece of code tries to import `Queue`, while it itself is `queue`, which effectively allows code to import python2 `Queue` even though they refer to python3 `queue`
U673BSS76 : But it broke without case-sensitivity
U673BSS76 : The actualy implementation of `Queue` is perfectly fine
U47HK8BS8 : oh - is your python file itself named queue.py?
U673BSS76 : There is a queue module (basically the code I pasted), I think it's a dependency from another package
U673BSS76 : Probably just to bridge python 3 and 2
U4EEBC4SJ : Following instructions from <https://docs.pytest.org/en/latest/getting-started.html>
U2BS4M1RV : Without looking it up, I believe fcntl is a Linux only library.
U2BS4M1RV : ```NAME   fcntl

MODULE REFERENCE
   <https://docs.python.org/3.6/library/fcntl>
   The following documentation is automatically generated from the Python
   source files. It may be incomplete, incorrect or include features that
   are considered implementation detail and may vary between Python
   implementations. When in doubt, consult the module reference at the
   location listed above.

DESCRIPTION
   This module performs file control and I/O control on file
   descriptors. It is an interface to the fcntl() and ioctl() Unix
   routines. File descriptors can be obtained with the fileno() method of
   a file or socket object.
```

U4EEBC4SJ : <@U2BS4M1RV> yeah that is consistent with what I'm seeing after some further reading, but why would that happen when following the Windows instructions?
U1NSCAY6R : <@U4EEBC4SJ> can you link the instructions you are following?
U4EEBC4SJ : <https://docs.pytest.org/en/latest/getting-started.html>
U1NSCAY6R : Hmm. But pip was happy when you ran the install?
U66HHFPT2 : Hey, I've been working with python lately, and I think I have most of the basic stuff down, but I'm kinda lost as to where to go moving forward, and so I wanted to ask if there was any advice for going 'beyond the basics', and really getting confident with python and everything it has to offer. Thanks!
U1BP42MRS : Build something with it!
U0NRYQNAZ : doesn't sirbot have a command for what's next?
U0NRYQNAZ : nice <@U66HHFPT2> :point_up:
U1BP42MRS : It's `/file &amp;lt;name&amp;gt;`
U0NRYQNAZ : ah ok
U1BP42MRS : Is the interpreter set up right?
U28MDQRL2 : yeah. I'm restarting the IDE
U28MDQRL2 : pycharm can't handle my code :100:
U28MDQRL2 : too good for it.
U1BP42MRS : Haha, woot
U1BP42MRS : :dagger_knife:
U28MDQRL2 : ha. Missed the `{}` around the json response. It wasn't that wrong
U1BP42MRS : Haha, yep. I was looking at general structure vs details
U68BS678X : Hi guys! Anyone who knows how to set up queued or delayed mails in django? Kind of like how you do in Laravel
U47HK8BS8 : <@U68BS678X> have you looked at something like django-celery?
U68BS678X : <@U47HK8BS8> No - I will have a look at it. Thanks!
U68BS678X : <@U47HK8BS8> btw, do you know anything about best practice for using environment vars in Django? It's very easy with Laravel, pulling the env vars from a .env file... But django doenst seem to do it that way.
So far, I have just placed an env.py file in the root of the project, and imported it into my settings.py.

U47HK8BS8 : yeah I actually wrote a little tool to simplify doing that in my python projects:
<https://github.com/jimjkelly/yaep>
```

U47HK8BS8 : but the basics if you don't want anything fancy is just `os.getenv(KEY, default)`
U47HK8BS8 : oh, and loading the `.env` file I suppose
U68BS678X : Okay, thanks <@U47HK8BS8>
U4EEBC4SJ : Yeah, pip was happy, confirmed that all of my dependencies are there too. I am puzzled
U5CGPBF0U : I'm using Flask and have fallen victim to the surprisingly common circular dependency issue. Do I pretty much have to use an app factory pattern to get around this?
U1BP42MRS : <@U5CGPBF0U> pretty much, if you have source to look at we may be able to suggest something else, but my guess is the factory is the simplest way
U5LNXQHN3 : I also found that it helped to avoid various flask extensions, because they would usually mean dragging in a further dependency on the `app` object to implement whatever behavior the extension added
U1BP42MRS : <@U5CGPBF0U> Here is a sample app we deprecated from <#C2FMLUBEU|community_projects> - <https://github.com/pyslackers/website-old>
U5CGPBF0U : <@U1BP42MRS> Well, my current code works, _but_ I'm making use of `from models import *` and using any other method of import fails. I heard that importing like that should generally always be avoided though. I'm not entirely sure why that is (yet) so I'm not sure if it's worth fussing over or if I might as well continue with it as is.
U5CGPBF0U : <@U5LNXQHN3> I'm using several Flask extensions, but I suppose the issue in this case comes from Flask-SQLAlchemy specifically. I'm wondering why the extensions exist if they're so prone to these problems though? Is the user just expected to use an app factory pattern? I'm surprised that even the most basic Flask tutorials seem to have this issue. The first tutorial I ever did wouldn't run due to a circular import error, even though the tutorial didn't mention any error and seemed to produce the correct results of the app running as if there was no error.
U1BP42MRS : It's discouraged because it pollutes the namespace with everything from `models`
U1BP42MRS : Including its imports
U5LNXQHN3 : All I can say is that I stripped Flask-SQLAlchemy out for exactly this reason. Other people managed to coerce it into working. :slightly_smiling_face:
U1BP42MRS : for example:
```
# module a.py
import secrets

def do_thing():
    pass

# module b.py
from a import *
locals()   # will have do_thing and secrets available...!
```

U5CGPBF0U : <@U5LNXQHN3> To be frank the reason I'm using the extension is because it seemed a lot more straightforward to use than SQLAlchemy by itself. Guess I'll have to choose between app factory and that.
U1BP42MRS : Any reason you dislike the app factory? I'd say the extension is worth it IMO
U5LNXQHN3 : I have an app factory, and even that didn't solve all my problems, because various functionality likes to have the app at module-level for decorators, etc
U5CGPBF0U : <@U1BP42MRS> Despite being a coder for longer than I'd like to specify, I've never had to implement an app factory pattern, and I'm wondering how much learning there would be. I generally learn pretty quickly but I am on a fairly tight schedule.
U1BP42MRS : It depends on if you need to import the app elsewhere to work with it, I usually avoid that (but sometimes you cant)
U5CGPBF0U : Also wondering if that is in fact the best move considering what Kylotan said. If implementing the pattern would still leave me with import issues later on...well...it sounds like it could be more efficient to just ditch problematic extensions
U1BP42MRS : In those cases, I use an `init_app` in a module that initializes the top level things that need to be decorated by `@app`
U1BP42MRS : I can't say I have had those issues
U1BP42MRS : You just need to pass the app around sometimes during the startup/bootstrap phase