U69HXQZJ9 : and those files are around 1 gb to 1.5 gb each
U5ZPMJA06 : *who tweets the most* -> Every tweet has a 'user' object, and in there is a 'name' attribute with the name of the user who made the tweet. You could use a dictionary to store all usernames and the tweet count. You know, going from the beginning to the end of the tweets and storing the username in a dict, or incrementing the tweet count if the user is already in the dict.
U5ZPMJA06 : The when the counting is done, you could convert the dict to a list of (name, tweetcount) tuples and sort that on decreasing tweetcount. Then print out the first 10 items of the result and you have a TOP 10 of most prolific tweeters.
U5ZPMJA06 : Or is it twitterers?
U5ZPMJA06 : :stuck_out_tongue_winking_eye:
U69HXQZJ9 : Either name works :)
U5ZPMJA06 : <@U69HXQZJ9> Dont worry about the file size because the construction 'for line in f: ...' doesn't load the entire file into memory.
U69HXQZJ9 : Ahh
U69HXQZJ9 : So no need to iterate
U69HXQZJ9 : Would I be too lazy if I asked someone here how I can do that?
U5ZPMJA06 : <@U69HXQZJ9> For example, to show the Twitter users with the most tweets:```
import json
import collections

twitterers = collections.defaultdict(int)

with open("stream_BhagNawazBhag.json") as f:
    for line in f:
        ob = json.loads(line)
        # text = ob.get("text", "")
        user = ob.get("user")
        if user:
            username = user.get("name", "")
        if username:
            twitterers[username] += 1

print "Most prolific twitterers (or tweeters? Twitter users?)"

for k, v in sorted(twitterers.iteritems(), key=lambda item: item[1], reverse=True):
    msg = u"%s - %d tweets" % (k, v)
    print msg.encode("utf8")
    if v < 6:
        break # We're not interested in people with less than 6 tweets

''' Will output:

Most prolific twitterers (or tweeters? Twitter users?)
KhalidMunawarPTI - 29 tweets
Adger Alam - 15 tweets
Mamma Mia - 12 tweets
Kashif Mughal ?? - 12 tweets
zain ali - 9 tweets
Nuzhat Khan - 9 tweets
lubna - 9 tweets
Fahid Gill - 8 tweets
PeacefulBalochistan - 8 tweets
????? ???? ?? - 7 tweets
Osman Kasim - 7 tweets
Maida Farid - 7 tweets
balochi - 7 tweets
Farrukh Hasan - 6 tweets
Pervez esabzai - 6 tweets
Ali Irfan - 6 tweets
PM - Imran Khan - 5 tweets

'''

```

U69HXQZJ9 : Wow
U69HXQZJ9 : That's really good
U69HXQZJ9 : I need to sit down and work on my python skills after September
U5ZPMJA06 : <@U69HXQZJ9> Yeah it really pays to know the data structures and the common idioms.
U5ZPMJA06 : I am sure my code can be made even better.
U5ZPMJA06 : Do you see a bug there? `not interested in people with less than 6 tweets` still, I see a line which says `PM - Imran Khan - 5 tweets`. How do you think that could be prevented?
U69HXQZJ9 : Maybe not include RTs
U69HXQZJ9 : It probably has included RTs
U5ZPMJA06 : No, I don't do anything with retweets in the code above.
U69HXQZJ9 : Maybe I can let that slide
U69HXQZJ9 : Will just truncate the rest
U5ZPMJA06 : The solution is to test _before_ outputting something, like:```
for k, v in sorted(...):
    if v &amp;lt; 6:
        break # We're not interested in people with less than 6 tweets
    msg = u"%s - %d tweets" % (k, v)
    print msg
```

instead of:

```
for k, v in sorted(...):
    msg = u"%s - %d tweets" % (k, v)
    print msg
    if v &amp;lt; 6:
        break # We're not interested in people with less than 6 tweets
```

U69HXQZJ9 : Ahh
U5ZPMJA06 : You know what the `break` statement does?
U69HXQZJ9 : Terminate
U5ZPMJA06 : Terminate what? The whole program?
U69HXQZJ9 : Inside the loop
U5ZPMJA06 : Correct! The break terminates the loop in which it is placed
U69HXQZJ9 : Glad I remember some of my stuff
U69HXQZJ9 : Will try this out
U5ZPMJA06 : Another nice question. What's the difference between a *class* and an *instance* ?
:stuck_out_tongue_winking_eye: