larger websites:
- can take too long for many html requests
- may have detectors for crawlers

Twitter limits: 180 calls/15 min
Reddit limits: 60 calls/minute is OK. (check Reddit Dev).
↑ ok; 30 is the official
1 sec/call is bare level of politeness (small sites may not have infrastructure to check limits).

multiple calls to search API meant duplicated data (Twitter doesn't let you go back in time/ 3200 tweets back in user history.
user view meant you can order things to avoid the duplicates.

tweepy pkg - very usable, altho' aimed @ application developers. - so take things out of python classes.

* KAIST "full" Twitter social graph (@ the time)

Reddit's usual API returns just most recent
But "cloudsearch" takes timestamps. (a backdoor!)
(using ↑option

Amit has exceptions mailed to him, . (just using unix mail) (∃ sysadmin-type stuff that helps, too, such as 'splat', 'sanglia',)
esp. b/c many servers.
Unexpected error codes can happen.

Keeping logs!

Amit ended up converting: C-Pickle → flatfile → SQL
esp. when going beyond single queries.
Flat files, hurrah!
JSON is slow, so convert to 1-line version.
For reddit, too big was faster to put in SQL (lots of data, lots of fields)
Reddis: too long for not faster than mySQL, but lots of data on one machine kills the memory.
this is best for 'find one record'.
But sequential: flat is usually what you want.
R is good: it's a direct memory-map.
Python is doing a system-independent store.

Social-integrator

what can handle mark-up data that can handle buggy XML.
(just find what you are looking for).
some recs: BeautifulSoup.

URL-generation - finding the URLs? (URL layout is speed to be considered private).

wget/curl - shell scripting
↓ ↘ can specify username. Use the API, not on the commandline.
less vulnerable re: resolving

also pandas, esp for tabular data.

just try to get everything to fit in memory.