Notes and Recommendations for Future Python Development

Fields to Consider Adding to JSON Output:

- Repository data from GitHub
  - Number of commits
  - Number of stars
  - Number of forks
  - Number of branches
  - Number of issues
  - Number of contributors
  - Number of releases
  - Whether repo is archived
  - Whether repo is active (e.g. was the last commit 4 years ago or 4 days ago?)
- Lines of code in each analyzed file + total lines of code analyzed in entire repo
- Industry/content type of the repo (e.g. gaming, security, educational, utility, API, etc.)

Errors:

- Repos that throw errors are skipped. Note that the *entire repository* is skipped, not just the failing file. Their results are not outputted because they are considered incomplete.
- The most frequent error is a `SyntaxError` thrown by the `ast parse` library. The most frequent cause is a print statement without parentheses (e.g. `print "Resizing glyph margins"` instead of `print("Resizing glyph margins")`).

Pending Tasks:

- TODO comments in all files
- Flesh out naming analysis
- Check repos before starting analysis
  - Check if repo is already in MongoDB. Currently only catching `DuplicateKeyErrors` thrown by `pymongo` library, but should check before running the repo to improve efficiency.
  - Check if all files compile? It would be inefficient to clone all the files and check that they compile before beginning the analysis on the repo, but it's also inefficient to begin analysis on the repo and quit halfway through when a file fails to compile. Currently, this program takes the latter approach. It may be worth reconsidering as the codebase grows.
- Finish building analyzer
  - Check file encoding types
  - Check different style guide expectations for line length, and implement custom check for expected line length. Notes included in `analysis.py` file.
- Write script to crawl GitHub and run analyzer on Python repos automatically (instead of listing repos for analysis manually).