# Assignment 1: Fyndiq code reading assignment

A very common thing you encounter in your work is that you need to modify an unknown file written by some unknown person. Open the file taskrunner.py (which is used in Fyndiqs systems) and study it.

Then explain:

- What do you think the purpose of this file is? What value does it add? What is it used for?
- Explain how the failure counter works and what it is used for.

# Assignment 2: Fyndiq programming assignment

The assignment is to use Django to create a URL shortener similar to http://bit.ly, but with a different URL encoding scheme. The shortened URLs will have the form http://myurlshortener.com/<word>/ where <word> is a word from the english language.

- Make sure to fulfill all the requirements stated in this document.
- Focus on writing clean, good looking and easily understandable code.
- Follow recommended Django coding style which is basically PEP8 with some amendments as described here: https://docs.djangoproject.com/en/dev/internals/contributing/writing-code/coding-style/
- Always use Django functionality where possible (for example, use Django form system and validation for any form fields required).
- At Fyndiq it is more important to write robust, easily readable and correct code than to do it quickly, so please pay attention to detail and corner cases.
- Provide a README with instructions on how to install and run the code in a contained environment, assuming that the computer that is to run the assignment does not have Django or any of your libraries installed.
- Use sqlite for database, and you could attach the database file with the finished code so that it is easier to run it.
- Make tests (unit tests and/or integration tests) for your code.

## Front page

On the front page, there should be a form to input a URL that should be shortened. When a URL is submitted via this form, the result page should give the user the shortened URL. If I then visit the shortened URL I should of course be redirected to the original URL.

## URL shortening scheme

It is common for URL shortening services to create a unique key consisting of the characters a-z, A-Z, 0-9, so that a key could be *a74Bd* and the corresponding shortened URL would then be *http://myurlshortener.com/a74Bd*. We will not follow this scheme, but instead make use of a word list (you should have gotten a file called *words.txt*). The key we use will always be a word from this word list.

## Wordlist

You must clean the wordlist yourself by converting all words to lowercase, and remove any characters that are not [0-9a-z]. Create a shell command that cleans the wordlist and loads it into the database. When a new request to shorten a URL comes from the form on the front page, the application should make a key by trying to pick a word from the wordlist that exists in the URL. The algorithm that picks a word from the wordlist should be fast and not take several seconds.

For example, if a user enters the URL http://techcrunch.com/2012/12/28/pinterest-lawsuit/ it should pick the first word in the wordlist that is a part of this URL. I haven't checked myself, but I would guess that the word in this case would be "lawsuit". If none of the words in the wordlist is a part of the URL, or if all words in the wordlist that are part of the URL are already used for shortening other URLs, any word from the wordlist should be used. When all the words in the wordlist have been used up as keys, the oldest existing key/URL should be deleted and that key should be reused for new URL submissions.

### Example

I enter the URL http://techcrunch.com/2012/12/28/pinterest-lawsuit/ into the field on the frontpage and press enter. The shortened URL I get back on the result page would then be http://myurlshortener.com/lawsuit/ if lawsuit is the first word in the wordlist that is part of the URL and that is not already used for any other shortened URL. If I would then go to the front page again and enter the URL http://lawsuit.se the shortened URL could be http://myurlshortener.com/windmill/, as no *unused* word in the wordlist was part of the URL and the application picked a random word from the wordlist.

## Questions?

Just e-mail niclas.helbro@fyndiq.se and I will answer as soon as possible.