Lazy code clone detector

Preprocessing

- Split code into blocks
    - e.g. by function (`def`)
- Remove unwanted text
    - Comments, blank lines, `import` statements

Hash processing

- Hash each block with the same Locality Sensitive Hash (`tlsh` library)
- Use a string similarity metric like Levenshtein distance to compare hash values
    - Not alpha order because differences in hash values can occur anywhere

Candidate searching

- For each block (represented by a hash value):
    - Check k many of its nearest later neighbors in the hash space
        - Calculate $d = \text{hash}(f_a) - \text{hash}(f_b)$
    - Find matches in degrees of closeness
        - Very close ($d \le d_{very\_close}$): should be **narrow margin**
        - Slightly close ($d_{very\_close} < d \le d_{slightly\_close}$): should be **wider margin** to allow for diverse variable name changes in stage 1
        - Not close ($d > d_{slightly\_close}$)

Clone selection

- If the pair of blocks are *very close*, do a text compare ("diff")
    - Split lines into words by whitespaces
    - Compare words from the same line in A and B and count pairs of not equal words
        - If the code is tokenized, even slightly different tokens mean very different code
    - Pass the diff result to clone analysis function
- Make a list of all the blocks that are part of *slightly close* pairs
    - Tokenize these blocks to standardize variable names
    - Rewrite the blocks with tokens replacing original text
    - Redo hash processing and candidate searching on the tokenized blocks
- **Don't tokenize** pairs that are *not close* (*may miss some type II clones, could investigate this)
- Look for *very close* pairs among the rehashed blocks again
    - Diff any new pairs found
    - Mark them as "found after tokenize" and pass to clone analysis
- **Don't analyze** any pairs that are still only *slightly close* after rehash

Clone analysis

- Classify the selected clone pairs by type of clone
    - *Very close* pairs found after the first hash are eligible for type I
    - *Very close* pairs found after the second hash are eligible for type II
- Use `diff` results to calculate ratios to evaluate clone exactness
    - # words modified / # words in a line (average over all lines in block)