Part 2 Exercises

More Regular Expression Exercises

Decimal Numbers

Write a function to match decimal numbers.
We want to allow an optional `-` and we want to match numbers with or without one decimal point:

```
>>> is_number("5")
True
>>> is_number("5.")
True
>>> is_number(".5.")
False
>>> is_number(".5")
True
>>> is_number("01.5")
True
>>> is_number("-123.859")
True
>>> is_number("-123.859.")
False
>>> is_number(".")
False
```

Hex Colors

Write a function to match hexadecimal color codes. Hex color codes consist of an octothorpe symbol followed by either 3 or 6 hexadecimal digits (that's `0` to `9` or `a` to `f`).

```
>>> is_hex_color("#639")
True
>>> is_hex_color("#6349")
False
>>> is_hex_color("#63459")
False
>>> is_hex_color("#634569")
True
>>> is_hex_color("#663399")
True
>>> is_hex_color("#000000")
True
>>> is_hex_color("#00")
False
>>> is_hex_color("#FFffFF")
```

```
True
>>> is_hex_color("#decaff")
True
>>> is_hex_color("#decafz")
False
```

Palindromes

Using the dictionary file, find all five letter palindromes.

Double Double

Find all words that have a consecutive repeated letter two times with only one other letter between them.
For example, these words should be matched:

- freebee
- assessed
- voodoo

Repetitive Words

Find all words that consist of the same letters repeated two times.
Examples:

- tutu
- cancan
- murmur

Substitution Exercises

Get File Extension

Make a function that accepts a full file path and returns the file extension.
Example usage:

```
>>> get_extension('archive.zip')
'zip'
>>> get_extension('image.jpeg')
'jpeg'
>>> get_extension('index.xhtml')
'xhtml'
```

```
>>> get_extension('archive.tar.gz')
'gz'
```

Normalize JPEG Extension

Make a function that accepts a JPEG filename and returns a new filename with jpg lowercased without an `e`.
Example usage:

```
>>> normalize_jpeg('avatar.jpeg')
'avatar.jpg'
>>> normalize_jpeg('Avatar.JPEG')
'Avatar.jpg'
>>> normalize_jpeg('AVATAR.Jpg')
'AVATAR.jpg'
```

Normalize Whitespace

Make a function that replaces all instances of one or more whitespace characters with a single space:

```
>>> normalize_whitespace("hello  there")
"hello there"
>>> normalize_whitespace("""Hold fast to dreams
... For if dreams die
... Life is a broken-winged bird
... That cannot fly.
...
... Hold fast to dreams
... For when dreams go
... Life is a barren field
... Frozen with snow.""")
'Hold fast to dreams For if dreams die Life is a broken-winged bird That cannot f
```

Compress blank links

Write a function that accepts a string and an integer `N` and compresses runs of `N` or more consecutive empty lines into just `N` empty lines.
Example usage:

```
>>> compress_blank_lines("a\n\nb", max_blanks=1)
'a\n\nb'
>>> compress_blank_lines("a\n\nb", max_blanks=0)
'a\nb'
>>> compress_blank_lines("a\n\nb", max_blanks=2)
'a\n\nb'
```

```
>>> compress_blank_lines("a\n\n\n\nb\n\n\nc", max_blanks=2)
'a\n\n\nb\n\n\nc'
```

Normalize URL

I own the domain treyhunner.com. I prefer to link to my website as `https://treyhunner.com`, but I have some links that use `http` or use a `www` subdomain.
Write a function that normalizes all `www.treyhunner.com` and `treyhunner.com` links to use HTTPS and remove the `www` subdomain.
Example usage:

```
>>> normalize_domain("http://treyhunner.com/2015/12/python-list-comprehensions-no
'https://treyhunner.com/2015/12/python-list-comprehensions-now-in-color/'
>>> normalize_domain("https://treyhunner.com/2016/02/how-to-merge-dictionaries-in
'https://treyhunner.com/2016/02/how-to-merge-dictionaries-in-python/'
>>> normalize_domain("http://www.treyhunner.com/2015/11/counting-things-in-python
'https://treyhunner.com/2015/11/counting-things-in-python/'
>>> normalize_domain("http://www.treyhunner.com")
'https://treyhunner.com'
>>> normalize_domain("http://trey.in/give-a-talk")
'http://trey.in/give-a-talk'
```

Linebreaks

Write a function that accepts a string and converts linebreaks to HTML in the following way:

- text is surrounded by paragraphs
- text with two line breaks between is considered two separate paragraphs
- text with a single line break between is separated by a `<br>`

Example usage:

```
>>> convert_linebreaks("hello")
'<p>hello</p>'
>>> convert_linebreaks("hello\nthere")
'<p>hello<br>there</p>'
>>> convert_linebreaks("hello\n\nthere")
'<p>hello</p><p>there</p>'
>>> convert_linebreaks("hello\nthere\n\nworld")
'<p>hello<br>there</p><p>world</p>'
```

Lookahead Exercises

All Vowels

Find all words that are at most 9 letters long and contain every vowel (`a`, `e`, `i`, `o`, `u`) in any order.

Unique Letters

Find all words that are 10 letters long and do not have any repeating letters.

HTML Encode Ampersands

Replace all `&` characters which are not part of HTML escape sequences by an HTML-encoded ampersand (`&`).
Example:

```
>>> encode_ampersands("This &amp; that & that &#38; this.")
'This &amp; that &amp; that &#38; this.'
>>> encode_ampersands("A&W")
'A&amp;W'
```

Broken Markdown Links

Make a function that accepts a string and returns a list of all reference-style markdown links that do not have a corresponding link definition.
Example:

```
>>> find_broken_links("""
... [working link][Python]
... [broken link][Google]
... [python]: https://www.python.org/""")
[('broken link', 'Google')]
```

As a **bonus**, make your function also work with implicit link names. For example:

```
>>> find_broken_links("""
... [Python][]
... [Google][]
... [python]: https://www.python.org/""")
[('broken link', 'Google')]
```

Camel Case to Underscore

Make a function that converts camelCase strings to under_score strings.

Get Inline Markdown Links

Make a function that accepts a string and returns a list of all inline markdown links in the given string.
Inline markdown links look like this:

```
[text here](http://example.com)
```

Example:

```
>>> get_inline_links("""
... [Python](https://www.python.org)
... [Google](https://www.google.com)""")
[('Python', 'https://www.python.org'), ('Google', 'https://www.google.com')]
```

Get All Markdown Links

Modify your `get_inline_links` function from the previous exercise to make a `get_markdown_links` function which finds all markdown links.
This function should work for inline links as well as reference links (including reference links with implicit link names).
Example:

```
>>> get_inline_links("""
... [Python](https://www.python.org)
... [Google][]
... [Another link][example]
... [google]: https://www.google.com
... [example]: http://example.com""")
[('Python', 'https://www.python.org'), ('Google', 'https://www.google.com'), ('An
```