re methods

First of all, small addition to flags

What if you want to use several flags?

```
# That's not enough!
pat = re.compile(r'abc', flags=re.I)
pat = re.compile(r'abc', flags=re.M)
```

Just or them!

```
pat = re.compile(r'abc', flags=re.I | re.M)
```

```
re.I

<RegexFlag.IGNORECASE: 2>


re.M

<RegexFlag.MULTILINE: 8>


re.I | re.M

<RegexFlag.MULTILINE|IGNORECASE: 10>
```

Other re methods

We haven't covered these ones yet

- `split(pattern, string, maxsplit=0, flags=0)` - split `string` by the `pattern`, if `maxsplit` > 0, perform only specified number of splits
- `sub(pattern, repl, string, count=0, flags=0)` - substitute parts of `string` matched with `pattern` on `repl`, if `count` > 0, perform only specified number of substitutions. `repl` could also be a function taking match object and returning a string
- `subn(pattern, repl, string, count=0, flags=0)` - as previous method, returns tuple with new string and number of substitutions made

If you have a compiled pattern omit `pattern` parameter in these methods

```
import re


s = 'Some string with punctuation... with not so
     much punctuation to be honest'

# Define a pattern
pat = re.compile(r'[\s\.]+')

# Split text by spaces and dots
pat.split(s)
['Some', 'string', 'with', 'punctuation', 'with',
 'not', 'so', 'much', 'punctuation', 'to', 'be',
 'honest']
```

```
pat.sub('!', s)
'Some!string!with!punctuation!with!not!so!much!punctuation!to
 !be!honest'

pat.sub('!', s, 5)
'Some!string!with!punctuation!with!not so much punctuation to
 be honest'

pat.subn('!', s)
('Some!string!with!punctuation!with!not!so!much!punctuation!to
  !be!honest', 11)
```

```python
def dot_to_dash(m):
    """
    Replace matches with dots to ' - ' and other matches to '_'
    :param m: re match object
    :return: str
    """
    if '.' in m.group(0):
        return ' - '
    return '_'


pat.sub(dot_to_dash, s)
'Some_string_with_punctuation - 
with_not_so_much_punctuation_to_be_honest'
```