More on Regular Expressions

More character classes

- `\s` matches any whitespace character (space, tab, newline etc)
- `\w` matches a word character (letters, numbers, underscore)
- `\b` matches an empty string at the beginning or end of a word

What would the match be?

- For the string `'The quick brown fox jumps over the lazy dog.'`
- `\w*`
- `\s\w*`
- `.*z.*`
- `\w*z\w*`

Verifying a credit card

- How do we check for a number such as `1234 5678 9012 3457`?

- Start with

```
\b\d{4}[ -]\d{4}[ -]\d{4}[ -]\d{4}\b
```

- We can group the 3 repeating patterns to

```
\b(\d{4}[ -]){3}\d{4}\b
```

Extracting

```
# The regex will accept any 16 digit number which has 4 groups
# of 4 digits.
# Each of the groups is separated by a space or a dash '-'.
# There can be text before or after the card number.

card = 'my card number is 1234 5678 9012 3452 don\'t tell anyone.'

pattern = r'\b(\d{4}[ -]){3}\d{4}\b'
match = re.search(pattern, card)

if match:
    print(match.group(), '- ', end='')
    number = match.group().replace(' ','')
    luhn(number)
```

Luhn Algorithm

```
def add_digits(string):
    '''Converts the chars of string into ints and adds them.

    string must only consist of digits
    '''
   return sum([int(c) for c in string])

def luhn(string):
    '''Print a result determined by the string and Luhn algorithm.

    'possible' if string is ok
    'invalid' if string is not.
    '''
    total = 0
    odd = False
    for c in reversed(string):
        if odd:
            n = int(c) * 2
            total += add_digits(str(n))
        else:
            total += int(c)
        odd = not odd
    print('possible' if total % 10 == 0 else 'invalid')
```

Beginnings and Endings

- I mentioned last time that there are two methods you can use when comparing regexes with a string, the `match` and `search` methods. `search` finds matches anywhere in the string. `match` finds matches only at the beginning of the string.

- You can also find matches at the beginning or end of lines. A string can run over several lines if `re.MULTILINE` is used.

- ^ matches the beginning of a string (or line if multiline).

- $ matches the end of a string (or line if multiline).

findall

- `search` returns one match (the first one) in a string
- `findall` returns a list of all the matches

Given the string

```
'k.shan@auckland.ac.nz, pbsord@lm.se,
 lme@123-4.com, one_two@three.four.com'
```

and the pattern `[\w.]+@[\w.]+` (special characters, in this case '`.`', lose their special meaning inside square brackets)

`re.findall(pattern, string)` returns

```
['k.shan@auckland.ac.nz', 'pbsord@lm.se', 'lme@123',
'one_two@three.four.com']
```

findall and files

- When combined with reading data from files `findall` is particularly powerful.
  - e.g. to find all of the tags in a web page you could do

```
file = open('index.html')
result = re.findall(r'<.*>', file.read())
```

  - This may not give the result you expected because the star operator is greedy. Use `<.*?>` to get the smallest match.

Replacement

- Just as you can replace any part of a string with the `replace` method you can do the same with regular expression matches and the `sub` method.

```
>>> s = "Replace all of my vowels with underscores."
>>> re.sub(r'[aeiou]', '_', s)
'R_pl_c_ _ll _f my v_w_ls w_th _nd_rsc_r_s.'
```