# Assignment 5 (Regexes) (30+10 bonus / 40 points)

All solutions should be stored in a directory called `assignment5` in your private repository. INF3331 students may solve all problems in this assignment to obtain a maximum of 40 points.

## 5.1: Nwodkram parser (10 points)

Nwodkram is a Markdown-like language for styling text. In this assignment, you will write a script which takes some text written in Nwodkram, and converts it into HTML. In Nwodkram, **bold** text can be written by surrounding it with `%`s, *italic* text can be written by surrounding it with `*`s, and to make a hyperlink, you can simply write `[www.nwodkram.com](displayed text)`.

How very useful! Quick, write a function `parse_nwodkram(text)` which takes a string written in nwodkram, and returns the corresponding HTML.

(See `examples_assignment_5.py` for examples.)

Name of files: `parser.py`

## 5.2: Extending nwodkram (6 points)

Modify your solution in 5.1 to add support for embedded images, quotes and super quick Wikipedia searches.

Images should be specified as `<imageURL>(w=WIDTH,h=HEIGHT)` where `WIDTH`, `HEIGHT` are the width and height in pixels.

Quotes should be specified as `>>QUOTELINE` at the start of a line, and become a HTML `<blockquote>`

Wikipedia searches should be specified as `[wp:QUERY]`, and become a hyperlink to the Wikipedia search page [1].

Name of files: `parser.py`

## 5.25: Background and disclaimer (0 points)

In the rest of this assignment, you will make a program which scrapes email addresses from raw HTML. This is useful if you want to get a bunch of email addresses and send them spam, but please don't do that, it's probably illegal and definitely not nice.

I am pretty confident the approach I am suggesting is way too slow to be of any real-world use, which is part of why I feel okay giving you the assignment, but in case it's not, don't actually compile lists of real people's emails without running it past an ethics committee.

Also, during the course of this assignment (especially the last part), be careful when testing your program on public websites - if you go downloading

[1] `https://en.wikipedia.org/w/index.php?title=Special:Search&search=QUERY`

a bunch of webpage, you may end up IP blocked because you're mistaken for a botnet. To make testing easier for you, we have been allowed to use an external site for testing - see clarifications.

## 5.3: Finding emails (9 points)

Write a function `find_emails(text)` which takes as argument a (long) string, and returns a list of all its email-like substrings. An email-like substring is of the form

`NAME@SERVER.DOMAIN`

where `NAME` and `SERVER` all consist of characters which are alphanumeric or one of `.#$%&~'*+-/=?_`` `|{}`. Additionally, `DOMAIN` must consist only of characters which are alphabetical or `.`, with the first and last characters required to be alphabetical. .

For some example input/output, see the file `examples_assignment_5.py` on the course page.
Name of files: `scraper.py`

## 5.4: Finding URLs (10 points)

Remaining in the point of view of someone who would like to scrape email addresses from webpages, it could happen that your would-be victims have elected not to put all their email addresses on a single page. To work around this, you might decide to also grab all URLs on the webpage, follow those and search them for emails as well and just keep doing this.

To do this, you need to add a function to the scraper you wrote. Write a function `find_urls(text)` which takes as argument a string of HTML, and returns a list of all the URLs pointed to by hyperlinks on the page. You may assume that in the raw HTML, a hyperlink is written as one of the following

```
<a href="PROTOCOL://www.HOST.DOMAIN/PATH"></a>
  <a href="PROTOCOL://HOST.DOMAIN/PATH"> </a>
<a href='PROTOCOL://www.HOST.DOMAIN/PATH'></a>
  <a href='PROTOCOL://HOST.DOMAIN/PATH'></a>
```

where:

- `PROTOCOL` is one of `http` or `https`
- `HOST` and `DOMAIN` consists of alphanumeric characters or `.`s, `-`s or `~`s
- `PATH` is a string of alphanumeric characters, `/`s, `.`s, `-` or `~`s

Your function should find all occurences of those tags in the text, and return the URLs (i.e. the thing between the quotes).

For some example input/output, see the file `examples_assignment_5.py` on the course page.
Name of files: `scraper.py`

## 5.5: Search (almost) ALL the pages (5 points)

We're almost ready for a life of riches and cybercrime. Write a function `all_the_emails(url, depth)`, which takes the url of a page, fetches the HTML, and stores a list of all the email addresses (defined according to 5.3) in that HTML. Then, it finds all hyperlinks (defined according to 5.4), and calls itself on all of those, appending the emails it finds there to the list.

## Clarifications

### 5.1

- It is possible to write wonky nwodkram like `% * text % *` (should this be bold? italic? both? neither?) or more sophisticated examples (like embedding bold text within URLs - who does that) which are ambiguous at best, and unparsable at worst[2].

  The nwodkram specification doesn't really care what you do with this kind of input - if you really care, you should probably just write HTML instead. So, to summarize, don't be lazy, but don't bend over backwards trying to make a robust language either.

- If you're uncertain about HTML, https://www.w3schools.com/html/html_formatting.asp is one of many references. In this assignment and the next, the only things you really need to know is that:

  - `<b>this</b>` is bold text,
  - `<i>this</i>` is italics,
  - `<a href="http://www.google.com">this</a>` is a hyperlink to Google,
  - `<blockquote>this</blockquote>` is a blockquote.

- If a hyperlink is written without a leading `http://` or `https://` (like `[www.google.com](google)`), you should prepend `http://` to it to make the link point where it seems to be intended (so that it becomes `<a href="http://www.google.com">this</a>`).

[2] If you're the kind of person who tries to find ambiguous input, I think that's great, and I would love to have you as a friend, coworker or both. Please don't interpret my dismissal as saying your concerns are silly - I'm only trying to keep the assignment 1) related to regexes, 2) interesting and 3) not too hard, and am trying to achieve 3) by not worrying too much about edge cases.

### 5.3

- We have been allowed to use for testing.
- This isn't actually the correct email specification. In reality, it's a bit trickier, with
  シモン@シムラ.jp being (I think) a legal email address, but we're going to ignore that in this assignment.
- The simplified definition given is ambiguous in that in `name@dom.co.uk`, it's unclear whether the domain is `.co.uk` or `.uk`. However, that doesn't really matter because we only care about the full string, not the exact decomposition.

### 5.4

- This won't actually describe all conceivable hyperlinks on a given webpage. For the purposes of keeping the assignment managable, we're ignoring that too.
- You may also assume that anything which looks like what is described in the example is in fact a hyperlink, even if it for example won't actually be present on the rendered page because it's within a comment tag. Can't go developing too robust software for *cybercriminals,* after all.

### 5.5

- Note that if the webpage you're on uses relative hyperlinks (links like `<a href="otherpage.html">` as opposed to `<a href="http://page.com/otherpage.html">`), as most webpages do, you will have to take this into account when designing your program if you want it to perform well. This means that you will have to make your hyperlink scraper from 5.4 match more links than specified in that assignment so that relative links are followed correctly.