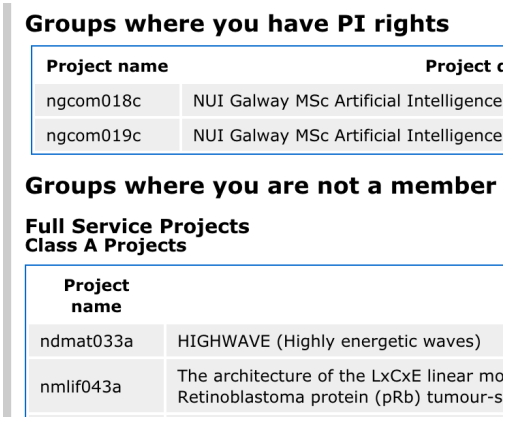
CT5132/CT5148 Lab Week 9

James McDermott

Regular expressions and web scraping

In lectures we studied regular expressions and used <regex101.com> to test regexs interatively. Now let's practice using them in Python.



Groups where you have PI rights

Project name	Project c
ngcom018c	NUI Galway MSc Artificial Intelligence
ngcom019c	NUI Galway MSc Artificial Intelligence

Groups where you are not a member

Full Service Projects
Class A Projects

Project name	
ndmat033a	HIGHWAVE (Highly energetic waves)
nmlif043a	The architecture of the LxCxE linear mo Retinoblastoma protein (pRb) tumour-s

Figure 1: On https://nationalservice.ichec.ie/login/login.php, there is a list of all the ICHEC projects, Classes A, B and C.

We can use Ctrl-A, Ctrl-C, Ctrl-V to put this data in a text file: `data/ichec_projects_scrape.txt`. However, it is now unstructured plain text. Let's use regular expressions to extract the project codes. Each code is like `ngcom018c` or `ulphy033a`.

1. `import re`
2. Read the data: `s = open("../data/ichec_projects_scrape.txt").read()`
3. Write a pattern `p` to match codes (maybe test on regex101.com)
4. Call a Python `re` function to find all the project codes.
5. Notice that the codes seem to have a specific encoding: `ngcom018c` is NUI Galway, Computer Science, 18, Class-C. `ulphy033a` is University of Limerick, Physics, 033, Class-A. Use **grouping** `( )` to extract the four individual parts in each code. Using this, how many Class-C Computer Science projects are there across all universities?
6. Write a new pattern to match only NUI Galway projects, and test it.

(Solutions: `code/count_ichec_projects.py`.)

Generative art using grammars

We already have the following code which will generate an image given a string (the string representing an arithmetic expression). Notice here we are using `x[0]` and `x[1]` to represent the two axes (not `x` and `y` as in the notebook).

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
n = 200
xs = np.linspace(0, 1, n)
ys = np.linspace(0, 1, n)
x = np.meshgrid(xs, ys) # x contains x[0] and x[1]
ps = "np.sin(40 * x[0]) * np.sin(30 * (x[1]+0.5)) * x[0] * x[1]"
p = eval("lambda x: " + ps)
plt.imshow(p(x))
plt.axis('off')
plt.show()
```

7. Change `ps` to make cooler/more complex images.

We also have the following code which will derive a new string we can use instead of `ps`:

```
from grammar import Grammar # assume we are in code/ directory
fname = "arithmetic.bnf"
g = Grammar(file_name=fname)
ps = g.derive_string()
print(ps)
```

8. Use this to generate several images. If you sometimes see the error `TypeError: Invalid shape () for image data`, that's probably because the grammar generated a string like `0`, i.e. a constant. There are ways to work around this, but we can just ignore it and generate a new one.
9. If you like, put everything in a convenient function or in a loop to make the process of trying new ones quicker.

10. Change `arithmetic.bnf` to allow some cooler/more complex images. Post your best images on the Discussion Board.

    Optional ideas: try different colour maps (see `matplotlib.cm`), or create polar coordinate variables (r, θ).

11. Optional. Take a look at `derive_string()`, defined in `grammar.py`, to see the implementation of the simple algorithm that we defined in lectures for deriving a string from a grammar.