Lab 3.1 (Deadline Friday 4 February 23:59)

- Upload your code to Einstein to have it verified.

List comprehensions

- Write a program called *numcomps_031.py* that uses a `for` loop and the `range()` function to generate a list containing the numbers 1, 2, 3, …, N (where N is an integer supplied from `stdin`).
- Use *list comprehensions* to have your program extract from the above list the following:

    1. All multiples of 3.
    2. The squares of all multiples of 3.
    3. The double of all multiples of 4.
    4. All multiples of either 3 or 4.
    5. All multiples of both 3 and 4.
- Here is an example of how the program should behave:

```
$ cat numcomps_stdin_00_031.txt
8
12
```

```
$ python3 numcomps_031.py < numcomps_stdin_00_031.txt
Multiples of 3: [3, 6]
Multiples of 3 squared: [9, 36]
Multiples of 4 doubled: [8, 16]
Multiples of 3 or 4: [3, 4, 6, 8]
Multiples of 3 and 4: []
Multiples of 3: [3, 6, 9, 12]
Multiples of 3 squared: [9, 36, 81, 144]
Multiples of 4 doubled: [8, 16, 24]
Multiples of 3 or 4: [3, 4, 6, 8, 9, 12]
Multiples of 3 and 4: [12]
```

Comprehensions with replacement

- Write a program called *repcomps_031.py* that uses a `for` loop and the `range()` function to generate a list containing the numbers 1, 2, 3, …, N (where N is an integer supplied from `stdin`).
- Use a *list comprehension* to have your program duplicate the above list but with all multiples of 3 replaced by ‘X’. For example:

```
$ cat repcomps_stdin_00_031.txt
8
12
```

```
$ python3 repcomps_031.py < repcomps_stdin_00_031.txt
Multiples of 3 replaced: [1, 2, 'X', 4, 5, 'X', 7, 8]
Multiples of 3 replaced: [1, 2, 'X', 4, 5, 'X', 7, 8, 'X', 10, 11, 'X']
```

Primes

- Write a program called *primecomps_031.py* that uses a `for` loop and the `range()` function to generate a list containing the numbers 1, 2, 3, …, N (where N is an integer supplied from `stdin`).
- Use a *list comprehension* to have your program extract all prime numbers from the above list. For example:

```
$ cat primecomps_stdin_00_031.txt
8
12
```

```
$ python3 primecomps_031.py < primecomps_stdin_00_031.txt
Primes: [2, 3, 5, 7]
Primes: [2, 3, 5, 7, 11]
```

More list comprehensions

- Write a program called *wordcomps_031.py* that reads words from `stdin` (one word per line) and stores them all in a list.
- Using *list comprehensions* and ignoring differences in case, have your program print the following lists:

    1. Words that contain exactly 17 letters.
    2. Words that contain 18+ letters.
    3. Words that contain four a's.
    4. Words that contain two or more q's.
    5. Words that contain the sequence 'cie'.
    6. Words that are anagrams of 'angle'.

- Your program should produce the following output when run against dictionary05.txt:

```
$ python3 wordcomps_031.py < dictionary05.txt
Words containing 17 letters: ['contradistinguish', 'counterproductive', 'counterrevolu
Words containing 18+ letters: ['arteriolosclerosis', 'counterrevolutionary', 'diethyls
Words with 4 a's: ['Alabama', 'Alabamian', 'amalgamate', 'Anastasia', 'Appalachia', 'A
Words with 2+ q's: ['Albuquerque']
Words containing cie: ['ancient', 'coefficient', 'concierge', 'conscience', 'conscient
Anagrams of angle: ['angel', 'Galen', 'glean', 'Lange']
```

- Hint: When checking if a word should be included in the new list convert it to lower case for the purposes of the check but add the original word to the list (should it pass the test).

Q no u

- Write a program called *qnou_031.py* that reads words from `stdin` (one word per line) and stores them all in a list.
- Making appropriate use of a *list comprehension* and ignoring differences in case, have the program print a list of all words in the list that contain a *q* that is not immediately followed by a *u*.

For example:

```
$ cat qnou_stdin_00_031.txt
question
Colloq
IQ
Iraq
Iraqi
q
Qatar
QED
q's
seq
inquest
```

```
$ python3 qnou_031.py < qnou_stdin_00_031.txt
Words with q but no u: ['Colloq', 'IQ', 'Iraq', 'Iraqi', 'q', 'Qatar', 'QED', "q's", '
```

- Note: Part of this exercise involves you coming up with test cases not present above but present in the hidden input/output on Einstein.