CT5132/CT5148 Lab Week 03

James McDermott

Solutions are available in the `/code` directory in our `.zip`.

1. Last week we calculated a list containing $e^x \quad \forall x \in [0.0, 0.1, ..., 1.0]$, using `range`, `lambda`, `map` and `math.exp`. Now, let's do the Numpy way, using `np.linspace` and `np.exp`. Solution: `exp_np.py`.
2. In the ECG (heartbeat) example, we saw how to use a Boolean expression to give a new array telling us where the Boolean expression is true:

```
x = np.array([-3, -2, -1, 0, 1, 2, 3])
x > 0
# array([False, False, False, False,  True,  True,  True])
```

We can also create a new array consisting of the values where it is true:

```
x = np.array([-3, -2, -1, 0, 1, 2, 3])
x[x > 0]
# array([1, 2, 3])
```

We can use the same idea now on the left-hand side of an assignment, to overwrite only values in an array where the Boolean expression is true. Use this to set all negative values in `x` to zero. Solution: `numpy_boolean_lhs.py`.

3. Find the biggest jump between any two consecutive values in the heartbeat (ECG) data. When does it occur? What were the before and after values? Hint: recall we have seen `np.diff` and `np.argmax`. I suggest `np.diff(x, prepend=x[0])` this time instead of `prepend=0`. Solution: `heartbeat_jump.py`.
4. Use Numpy to create a "chessboard" pattern like this:

```
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

Hints: use `np.linspace`, `np.meshgrid`, and the `%` operator. For the final touch, look up `astype` so that your array is int-valued, not float. Solution: `chessboard.py`.

5. Create the 2d array a_{ij}, a "vertical" 1d array b_i, a "horizontal" 1d array c_j, and a scalar d. Calculate the new 2d array q where $q_{ij} = a_{ij} + b_i + c_j + d$. (This scenario will be familiar to CT5141 Optimisation students as it occurs in linear objective functions.) Solution: `2d_1d_scalar.py`.

```
a: 9 5 1    b: 10    c: 100 200 300    d: 1000    q: 1119 1215 1311
   4 3 8       20                                    1124 1223 1328
   2 7 6       30                                    1132 1237 1336
```

6. Implement the broadcasting rules as a Python function. See `broadcastable.py` for a skeleton and doctests. Run it using `python -m doctest broadcastable.py`. See `broadcastable_sol.py` for a solution.

7. In our fractals example, we claimed that when a point "escapes" from a circle of radius 2, it will never come back, but in `test_escape_and_return()`, that's exactly what seemed to happen. Why? Solution: `fractal_escape_and_return.py`.

8. Try different values for `zmin` and `zmax` in the Julia example. What is the effect? Try different values for `c`. Do you get any interesting images? I liked `c = -0.015 + 0.66j`.

9. Try different colourmaps in the Julia set. Look up:

   `https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html`

   to see a gallery.