Python better practices demo

docker build -t mydemos .
docker run -it mydemos

Chapter 4:

P130 I27: comprehensions instead of map and filter to make a list:

- Much easier to read
- Can use multiple conditions and per loop level

But what if the list is massive? RAM in danger?

P145 I32: Consider generation expressions instead of building big lists

- Perhaps I only care about the current state (did somebody say Markov chains?)
- Huge lists memory intensive
- Chaining together generator expressions can execute quickly, and is memory efficient
- Example case: ensuring a model only sees one batch at a time, no memory explosions

P137 I30 Consider Generator functions instead of building big lists

- Now we can extend this idea to functions to do more complex sequence of tasks
- This can be clearer than having a function return a list of accumulated results
- E.g data augmentation, batch processing

P147 I33: Combine multiple generators with yield from

- Yield from lets you chain them together without writing nested loops
- This is useful for making pipelines
- Yield from follows the stream all the way back to stage 1
- NB: The point is not to avoid using lists. It is that this is a clean way to combine generators. There are no intermediate lists.
- Speed improvements over manual yields (in book 13.5% example)

P150 I36: Itertools

- Itertools toolbox makes iteration pipelines elegant, fast and memory efficient
- Fall into 3 main categories: linking iterators, filtering items from an iterator, and producing combinations of values

Why is this useful?

- When dataset is bigger than RAM, cannot load everything at once. Generator is a nice way to do in batches
- In similar way, can do on the fly temporary transformations/augmentations without creating massive augmented datasets
- Can see generator use for sequential sampling (MCMC, RL)

- May be useful for real time data processing where data arrives continuously.