COMP10001 Foundations of Computing
Semester 2, 2016

Tutorial Questions: Week 11

1. Implement a function `powerset(items)`, which generates a list of all the subsets of `items`. For example:

```
>>> powerset([1,2,3])
[[], [1],[2],[3],[1,2],[2,3],[1,3],[1,2,3]]
```

You may use `itertools.combinations`, but not `itertools.power`, in your solution.

A:

```
import itertools

def powerset(items):
    pset = [[]]
    for i in range(1,len(items)+1):
        for subset in itertools.combinations(items,i):
            pset.append(list(subset))
    return pset
```

2. The following code is meant to output a dictionary of player IDs, and the list of cards played by a given player:

```
from collections import defaultdict
players = iter(range(4))
cards = [('3C',2),('5D',3),('8C',1),('2H',2)]
holds = defaultdict(list)
for card in cards:
    for player in players:
        if card[1] == player:
            holds[player].append(card[0])
holds = dict(holds)
```

On completion of the code, however, the content of `holds` is:

```
{2: ['3C']}
```

and not the expected:

```
{1: ['8C'], 2: ['3C', '2H'], 3: ['5D']}
```

What is the issue, why does the problem occur, and how can it be fixed?

A: *The issue is that we attempt to iterate back over an iterator we have exhaustively iterated over (`players`); recall that once we have iterated to the end of an iterator, there is no way to rewind. The (easiest) solution is to convert the object into a sequence (which we can iterate over as many times as we want), e.g. change `players` to `range(4)`.*

3. Timbuktu Public Library has recently decided to digitize its archive of local newspapers from the last 50 years. The pages of the newspapers will be scanned and stored as digital color images. Suppose that each newspaper page will be scanned as a *3000 pixels wide* and *6000 pixels high* RGB image. The library currently owns 15 TB (Terabytes) of storage (you may assume that one Terabyte is equal to 1,000,000,000,000 bytes). How many pages of newspaper can be hosted using the library's existing storage?

Note: Assume uncompressed RGB format is used, with one byte for each of the R, G, and B components.

A:

$$\left\lfloor \frac{15 \times 10^{12}}{3000 \times 6000 \times 3} \right\rfloor = 277777 \text{ images}$$

4. Write the following functions (potentially reusing `open_img` from the code provided):

(a) `rotate180`, that takes an input file `infile` and an output file `outfile` as arguments, and generates a new image in `outfile` based on rotating `infile` by 180 degrees

A:

```
def rotate180(infile,outfile):
    dim,pix_in,mode = open_img(infile)
    img_out = Image.new(mode, dim)
    pix_out = img_out.load()
    for x in range(dim[0]):
        for y in range(dim[1]):
            pix_out[x,y] = pix_in[dim[0]-x-1,dim[1]-y-1]
    img_out.save(outfile)
```

(b) `rotate90`, that takes an input file `infile` and an output file `outfile` as arguments, and generates a new image in `outfile` based on rotating `infile` by 90 degrees clockwise

A:

```
def rotate90(infile,outfile):
    dim,pix_in,mode = open_img(infile)
    dim = (dim[1],dim[0])
    img_out = Image.new(mode, dim)
    pix_out = img_out.load()
    for x in range(dim[0]):
        for y in range(dim[1]):
            pix_out[x,y] = pix_in[y,dim[0]-x-1]
    img_out.save(outfile)
```

(c) `rescale`, that takes an input file `infile` and an output file `outfile` as arguments, and generates a new image in `outfile` based on scaling `infile` to half of its original size

A:

```
def rescale(infile,outfile):
    dim,pix_in,mode = open_img(infile)
    dim = (dim[0]//2, dim[1]//2)
    img_out = Image.new(mode, dim)
    pix_out = img_out.load()
    for x in range(dim[0]):
        for y in range(dim[1]):

            # construct a list of pixel values for each RGB
            # channel to average over
            pixels = [[],[],[]]
            for i in range(2*x,2*x+2):
                for j in range(2*y,2*y+2):
                    for k in range(len(pix_in[i,j])):
                        pixels[k].append(pix_in[i,j][k])

            # calculate the average for each channel
            pixel = []
            for channel in pixels:
                pixel.append(sum(channel)//len(channel))

            pix_out[x,y] = tuple(pixel)
    img_out.save(outfile)
```