CSC108H Week 12 Lab

To earn your lab marks, you must actively participate in the lab. *You do not need to finish in the time allotted, you just need to arrive on time and work hard.*

1 Objectives

1. Write a permutation generator
2. Practice with an interesting application of 2D images

2 Driver and navigator

Pick which of you will drive first, log in, and start up Wing.

3 Generating Permutations

Last week, we developed an anagram generator that started by generating all permutations of a word. The example we used was the word **bore**, which has four letters, and so has $4! = 4 * 3 * 2 * 1 = 24$ permutations of its letters. Those permutations are as follows: bore, boer, broe, breo, beor, bero, obre, ober, orbe, oreb, oebr, oerb, rboe, rbeo, robe, roeb, rebo, reob, ebor, ebro, eobr, eorb, erbo, erob.

This week, we will write an iterative algorithm that, given a word, creates a list of its permutations. Thanks to the CSC108 student who gave me this idea!

1. Download the file `generate_perms.py`
2. Write the helper function `add_char` according to its docstring. For example,
   `add_char ('c', ['ab', 'ba'])`
   should yield
   `['cab', 'acb', 'abc', 'cba', 'bca', 'bac']`. Be careful that you do not add duplicates to your list. For example, `add_char ('a', ['ab', 'ba'])` should yield `['aab', 'aba', 'baa']`.
3. Write the function `generate_perms` according to its docstring. Start with the list consisting of just the empty string, and make use of `add_char` as part of this function.
4. Test your `generate_perms` on the empty string, a string of length 1, a string of length 2, and a string of length 3.

4 Virtual Traveling

In this section, you are going to help some CS profs travel the world without leaving home. Here's how to do it:

- Pick a 320x240 picture of a CS prof against a green or grey background, and a 320x240 picture of wilderness. There are several available on the Labs webpage. Also, download the starter code `travel.py`.
- Import `media`. Select a picture using `choose_file` and `load_picture`, and then call `inspect` on that picture. This will open up a picture tool that you can use to examine colour values and (x, y) coordinates. Where is pixel (0, 0)? What are the coordinates of the bottom-right pixel? What RGB value does pixel (0, 0) have? How about a person's eyes? Skin? Shirt?

- We can start traveling by replacing all the pixels that are part of the grey or green wall — they're all about the same colour — with pixels from another picture. Using the `distance` function (get `help` on it!) you can tell how close one colour is to another.
- Each pixel has an (x, y) coordinate. You can get those by calling `get_x(pixel)` and `get_y(pixel)`. Using `get_color`, you can then get the colour value. Once you've got the (x, y) values, you can get the pixel at the same location in the background picture using `get_pixel`, get that pixel's colour, and set the pixel from the person's picture to that colour.
- How do you know which colour to compare the pixels with using `distance`? Pick a colour from a pixel in the picture; you'll need to experiment. Perhaps the pixel at (0, 0) would work. Maybe (160, 10). And how close should the other pixels be? Maybe `distance` should return less than 30? 20? 40? Too small and not enough pixels get replaced. Too large and all of them do.

  Choosing these values is a fun but tricky problem — be sure to ask your TA for help when you get stuck! And you'll never get it quite perfect, although you can come close. (In order to do this fully, you'll need pictures of people against a bright blue or green background.)
- Once you've written your travelling code, add a name tag to an instructor's shirt using `add_text` (look it up using `help()`).
- Now use the shape functions (`add_line`, `add_oval`, `add_oval_filled`, `add_rect`, `add_rect_filled`, ...) to draw glasses, a moustache or something else on these defenseless instructors. (It is no coincidence that Dan's picture is not here!)

Show your work to your TA.