Introductory Programming for Data Science Assignment 3

Dr Kieran Hughes

January 2024

Instructions

This assignment is due by 11pm on Friday 26th April. The answer to all your questions must be in a Jupyter workbook with all cells having being already run (so the output is visible). Name your workbook as IntroProgAssignment3 - StudentID StudentNAME.ipynb. Label the start of each part of each question and put any written answers in markdown cells. All answers should appear in order, so your answer to question 2 B should be below the answer to question 2 A and above the answer to question 2 C.

Include any libraries when you first require them. Ensure when you are uploading your work you only upload your own work. Just upload the ipynb file and the image files requested. Do not zip anything. This is NOT a group assignment. You must complete the assignment individually. If you share your work, your marks will be divided. Some students may be asked to demonstrate their knowledge of the submitted work. Each assignment question clarifies the libraries that can be used for each part of each question.

If you are asked to write a function called `sample_name` then name the function `sample_name` not `samplename` or `Sample_name` or `Sample_Name` or anything else for that matter. Failure to do so makes testing your code more difficult and will result in a loss of marks.

Please note that you will be docked 5% for each day you are late with submission, in accordance with ATU Sligo Marks and Standards.

Questions

1. In this question you will be manipulating images of your choice.

   A note on libraries for Q1: In parts (a), (b), (c), and (d), you are only to use the PIL library to open and save the images and you are not allowed any other modules or libraries. All manipulation of the pixels you must do yourself. In parts (e) and (f) you are allowed to use the libraries PIL, numpy, matplotlib, cv2 and any other libraries you like.

   When saving images, save them as PNG. Choose images to demonstrate your functions, any images are fine (as long as safe for work!) and are in any format that Pillow reads. You will include these images as well as the altered versions in your submission. All functions should include as input an opened PIL image object. The functions must always create a copy of the original image and they will return the manipulated image. Do not put saving images as part of the function. Using your chosen image, display it altered in the workbook and save the altered version to a file that you include with your submission.

   (a) Write a function called `just_blue` that isolates the blue channel by setting the values associated with the other channels to 0

   (b) Write a function called `grey_scale` that returns a new grey scale version of the input image.

   (c) Write a function called `weighted_grey_scale` that returns a new grey scale version of the input image that is based on a user defined set of weights.

   (d) Write a function called `mirror_image` that returns an altered version of the input image such that the right hand side is a reflection of the left hand side.

   (e) Write a function called `measure_symmetry` that uses the function `mirror_image` on images of peoples faces and returns a score of how symmetrical they are.

(f) This part is where you get to explain and demonstrate your knowledge. There is a substantial portion of the marks going for this part of the question. Having read your answer to this part, I should be convinced that you fully understand the code you have written, it's applications and limitations. You should discuss and justify any decisions you have made. List, justify and explain all the investigations and experimentation you did with your code and what you have learnt. Clearly and succinctly explain any conclusions you have drawn from investigating your code. Feel free to write code (including additional functions that you have written) to perform investigations. You can also include code output, results, tables and visualisations. Use markdown cells for text.

2. In this question, you will be analysing the efficiency of common sorting algorithms.

   A note on libraries for Q2: In part (a) no libraries are allowed. In parts (b) and (c), you are only allowed to use the libraries time, numpy, timeit, random, sys, resource, cProfile, memory_profiler and psutil. In part (d) you are allowed to use any library you like.

   (a) Write functions called `merge_sort`, `quick_sort` and `bubble_sort`. These functions should implement the sorting algorithm indicated by the name. Do not use any built in sorting function. Each function should take as input a list of numbers (integers and/or floats) and returns a copy of the list with the elements sorted in non-decreasing order. Your `quick_sort` should use the last element of the input list as the pivot.

   (b) Write a function called `sort_timing`, which takes a list of numbers (integers and/or floats) as its only input. Calling this function should run each of the three sorting algorithms implemented in the previous sub-problem on the given list, timing each call. The function should return a tuple of floats `(t_merge, t_quick, t_sort)` corresponding to the times that it took to sort the input list with `merge_sort`, `quick_sort` and `bubble_sort` respectively.

   (c) Of course timing is only 1 aspect of algorithm efficiency. Space complexity is also important. Write a function called `sort_space_complexity` which takes a list of numbers (integers and/or floats) as its only input. Calling this function should run each of the three sorting algorithms implemented in the previous sub-problem on the given list, calculating the space requirements for each. The function should return a tuple of floats `(s_merge, s_quick, s_sort)` corresponding to the space requirements that it took to sort the input list with `merge_sort`, `quick_sort` and `bubble_sort` respectively. Note if using Anaconda on windows you may want to use the psutil library.

   (d) This part is where you get to explain and demonstrate your knowledge. There is a substantial portion of the marks going for this part of the question. Having read your answer to this part, I should be convinced that you fully understand the code you have written, it's applications and limitations. You should discuss and justify any decisions you have made. List, justify and explain all the investigations and experimentation you did with your code and what you have learnt. Clearly and succinctly explain any conclusions you have drawn from investigating your code. Feel free to write code (including additional functions that you have written) to perform investigations. You can also include code output, results, tables and visualisations. Use markdown cells for text.