ECOR1051 PROJECT DESCRIPTION

MILESTONE 1

This document is meant to be read in conjunction with the overall Project Description. It describes only those tasks associated with Milestone 1.

The first milestone spans three regular lab periods. Accordingly, to help you manage your workload, the work is described as three labs. All three labs are included in this one document so that you can work at your own pace and you can see the progression of tasks that compose a project.

- Attendance at lab periods is mandatory
- Time management is one of the skills that you are learning to develop during this project. Not all tasks have deliverables due immediately after the lab period; however, it is highly recommended that you complete the work of each task by the end of each corresponding lab period.

File Naming Convention: Throughout the project, you will be given specific formats for your filenames that include your team identifier.

Example format: xxx_file.ext where xxx is your team identifier and “ext” is the required file extension (e.g. “docx”, “pdf”, “py”, etc).

Team identifiers are numbers ranging from 1 to 171. If your team identifier is 1, then your filename should be 1_file.ext

Getting Started Problems

In the first milestone, although labs are mandatory, it may happen that one of your teammates does not show up. Things usually settle by the second week of the project.

If one of your team members doesn't show up at the first project lab (P1):

- The team leader should tell one of the TAs the name of the missing student. (If the team leader is absent, one of the other team members should do this). The TAs will collect this information and forward it the instructors.
- The team should complete the P1 tasks, but don't put the absent student's name in the contract (as they didn't participate in the drafting of that document).
- We will attempt to locate the student and assign an appropriate penalty after discussing the absence with him/her.

Table of Contents

Milestone 1 Lab 1 (P1)

There are two major tasks to complete in your first lab period: team formation and problem exploration.

P1 Task 1: Team Formation

You have just met your new colleagues with whom you will work for the remainder of the term. Get to know each other *informally* – Exchange phone numbers and emails. Tell each other about your backgrounds and your interests. There are good reasons to learn a bit about each other, such as:

- Where do you live? (Some people may need to travel in order to work together)
- Do you have a job? (Some people may have time constraints)

You must also get to know each other *formally* by collaborating completing your Team Contract.

1. Download the ***Team Contract Template*** posted under Milestone 1 P1.
    - Required Filename: xxx_teamContract.pdf (or .docx) where xxx is your team identifier
2. **Follow the instructions** within the contract, participate fully, spend time thinking about your answers, plan out all your meeting times and your conflict resolution strategies. Should conflict occur, the instructors will be using this contract to assess the situation and decide how to intervene.
3. Give a printed and signed copy of the final page of your team contract to your TA.
    - The TAs will not have paper copies. You must print this page yourself.
    - If it is submitted at a later lab, it will be accepted but will be marked as late (i.e. zero marks)
    - Ideally, give your sheet to your project TA. In the uncommon case where your marker TA is not present in your lab period, please give the sheet to any TA. As long as your name, lab section and cuLearn team identifier are correctly recorded on the page, we can route it to your project TA.
4. (Team Submission): Submit the electronic version of your team contract on CULearn.
    - Every individual whose signature appears on the printed final page will earn the same mark.

P1 Task 2 – Project Understanding

As a team, watch the Video Depiction of your Final Project.

As a team, (re-)read the Project Overall Description. This document provides an overview of the project. You won't need all of the information at this very moment, but the intention is to let you see the "big picture", so you understand what program you will ultimately develop.

Do you have any questions about the project? Are there ambiguities? Do you see any upcoming challenges for team work? You don't have to have all the answers now because some will come about as you begin to work, but make sure that everyone in the team has a common understanding.

There is no submission required. Please do the work for your own understanding.

P1 Task 3 - Problem Exploration – Understanding RGB Colour Representation

Our photo-editing program is based on the *RGB Colour Representation* for digital images. Your project begins by understanding RGB and in turn how this is programmed in Python.

There is an excellent online tutorial at https://www.w3schools.com/colors/. Take the time to explore this tutorial, including experimenting with all the examples that they provide (Try it Yourself). As explained there, colours are represented in several ways: RGB, Hexadecimal, HSL and HWB. **Read only the sections about RGB.** Introductory topics to read (on the left-hand menu) are:

- Colors HOME
- Color Shades
- Color RGB
- Color Hues (Hues, Tins, Shades and Tones)
- Color Picker
- Color Mixer
- Colour Converter

There is no submission required at this time. Please do the work for your own understanding.

P1 Task 4 – Coding Exercise to Learn CIMPL

Several 3rd-party Python libraries support image processing. We'll use **Cimpl**: ***C***arleton ***i***mage ***m***anipulation ***P***ython ***l***ibrary. This library contains a set of functions that let us load/save digital image files, retrieve and modify the colour of individual pixels, display images. The Cimpl library, in turn, uses Pillow to load images encoded in several popular formats (JPEG, GIF, PNG, etc.) and to store modified images in these formats

There are two CULearn postings to help you get started – they are similar but one is in text and one is a video

1. Video: Getting Started with CimpL
2. Text : Getting Started with CimpL
   - A much more detailed document that includes instructions on how to check whether your installation of Python, Wing as well as Pillow are up to date; and how to iterate over an image.

You may work collaboratively; however, before your next lab, each individual must know how to write a CImpl program that loads an image, and then iterates over every pixel in that image.

There are some notes below for those who are using MACs, if you experience difficulty.

There is no submission required at this time. Please do the work for your own understanding.

P1 Task 5 – Testing Your Understanding

Individually, complete the CULearn Quiz titled – **Understanding RGB**. The Quiz has a 15-minute time-limit. Complete the readings and work described in Task 3 and Task 4 before you begin.

Milestone 1 Lab 2 (P2)

It is time to start writing your first image filters. In this lab, your task is to first understand what a "*channel*" is, and then to divide the work up so that everyone writes one filter each, along with a test function of that filter (in a standard team of four people).

P2 Task 1 : Understanding the Design Solution of an Image Filter

As you know, the colour of a RGB pixel is represented by a triplet of integer components, each of which ranges from 0 to 255. The first component is the quantity of red, the second component is the quantity of green, and the third component is the quantity of blue. We can think of an RGB image as being composed of three overlapping *monochromatic* images (one consisting of pixels containing shades of red, another consisting of shades of green, and the third consisting of shades of blue). These monochromatic images are often referred to as *channels*.

Image *filtering* is the process of changing or extracting one aspect of the digital image representation to make the image look different. For instance, a **red-channel** filter uses only the red channel of an image, clearing the blue and green channel. In contrast, a **green-channel** filter uses only the green channel of an image, clearing the red and blue channels. The effect of these two filters are shown below.

In Milestone 1, your team will write four of the simplest image filters, to be called respectively:

- red_channel
- green_channel
- blue_channel
- combine

Each filter reads in an image file and writes out an altered version of that image file. The red/green/blue_channel filters are shown on left hand side of Figure 1, below. Each one of these returns an image that contains only the red/green/blue portions of the input image (e.g. p2_original.png). The `combine` filter (shown on the right hand side of Figure 1) reverses this process. The `combine`() filter accepts three image files (red/blue/green_channel.png), and returns a single image file that is a combination of those three files. If you have coded everything correctly, the `combined_image.png` will be identical to the `p2_original.jpg`. We shall be able to know whether your software worked at a glance.

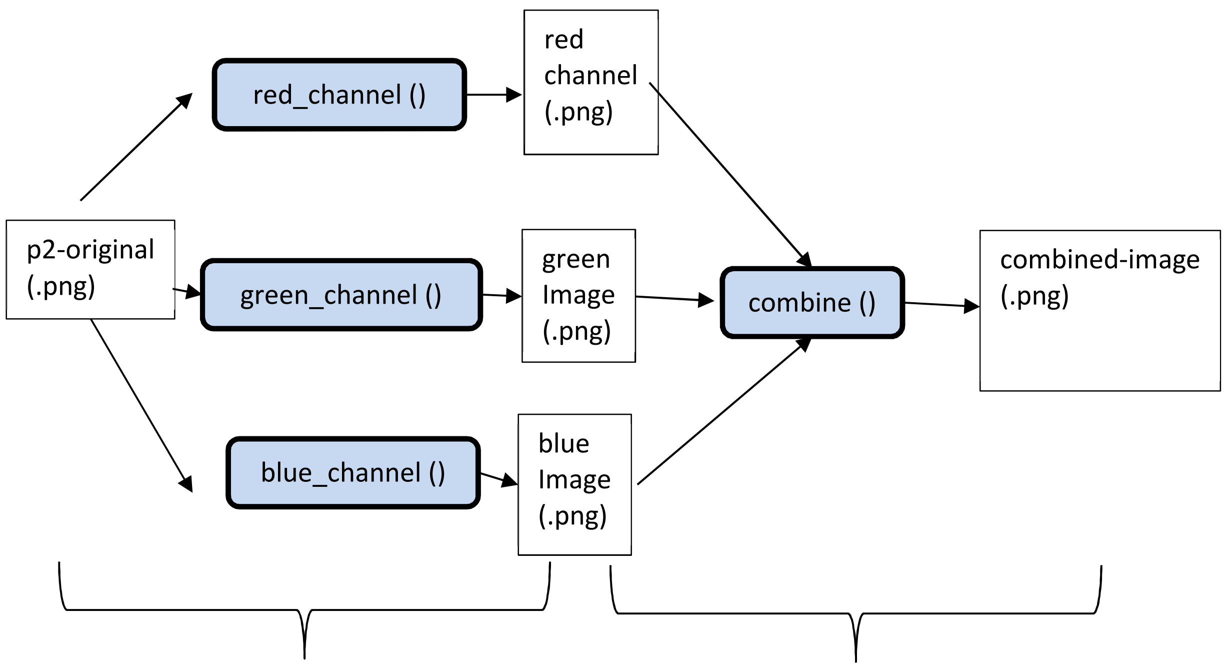

FIGURE 1 WORKFLOW FOR MILESTONE 1'S IMAGE FILTERS

CAUTION

For this lab, you must use the .png versions of the images, not the .jpg.

For the remainder of the project, you may use either version (.png or .jpg)

The .jpg format is a compressed format. When a .jpg image is saved, the compression algorithm will change the r,g,b values that you just finished zero'ing! Your combine() filter will not work.

There is no submission required. Please do the work for your own understanding.

P2 Task 2: Writing your First Image Filter(s)

There are four filters (i.e. functions) and in most teams, there are four members.

The leader of the team shall:

1. Ensure that everyone understands the task to be completed. The leader of the team does not need to be the expert and "have the solution"; instead, the leader must ensure that each member is empowered to do their portion of the work.
2. Delegate who is writing which filter. In teams of three (or, if there are absent teammates), one person will have to write more than one filter.

Each individual shall:

1. Follow the FDR to construct their assigned filter function.
    - The filter must not destroy the original image. It must create a new image that is a copy of the original image, with appropriate changes
    - Required Filename: **xxx_P2_yyy.py** where **xxx** is your team identifier and **yyy** = {red, green, blue or combine)
    - Look in the Cimpl.py file for examples of type annotations for images.
2. You can first test your filter by showing the image. Showing the image is the most fun way to verify that your program is working. You will get immediate visual feedback when your image turns red or green or blue.
3. You must also test your filter **programmatically**, as we've been doing in previous labs. Write a corresponding test function for your filter, following the practices taught in the unit testing lecture. This test function must verify that each pixel is correct, where "*correct*" is defined by your particular filter. **See "test_grayscale.py" and "test_invert" for examples of how to use the `check_equal()` function to correctly test a filter using a small test image for which we can calculate the expected pixel values.**
    - Example: For the `red_channel()` filter, all pixels must have zero green and blue components
    - Example: For the `combine`() filter, each pixel must have the red, green, blue component values from the same pixel location in the red, green, blue images, respectively.
    - If your test function finds an error, it would be useful to give the location of the pixels that failed. A software developer would likely want to know the location of the pixels where the test failed (perhaps the problem occurs in specific regions of the image) as well as the RGB values of those pixels (perhaps the problem only occurs for some colours, but not others)
    - Note to advanced programmers: When writing the test functions, you may only make use of Cimpl functions in your code. For instance, you are not allowed to use Manhattan norm or zero norm when comparing the original image to the filtered image.

4. Your final version of your file should contain (1) the filter function (2) the test function and (3) a main script in which the test and filter functions are called.

Each ***INDIVIDUAL*** must submit their own work to be marked individually. Please refer to the Marking Rubric. If a person is writing two filters because the group size is less than the standard size of four:

- each filter (and its test function) should be in a separate file.
- Two files should be submitted on CULearn by this individual

Additional Notes for the combine() filter

The person who writes the `combine()` filter does not need to wait until the others have completed their work. Single-channel images are provided on CULearn, in the folder containing ***CImpl and Approved Sample Images***. Use the .PNG versions, not the .JPG versions.

Ultimately, you want to test your combine() filter against those of your teammates' filters.

```
# Load up a sample image (either .png or .jgp)
 image = load_image(choose_file())

# Use your teammate's filters to separate out the colour channels
# into three separate image-variables
#    NO NEED TO SAVE THESE IMAGES INTO FILES
 red_image = red_channel(image)
 green_image = green_channel(image)
 blue_image = blue_channel(image)
# Re-combine these separate image-variables into one image
 combined = combine(red_image, green_image, blue_image)
# Save the re-combined image in a file.
# You can then use any photo-viewer to see if it's the same as the
# original file.
 compare_images(image, combined)
```

While your teammates are busy working, you need to get busy too! You can use the red/green/blue_channel images that we've provided, but you must use the .PNG versions.

```
combined = combine( load_image('red_image.png'),
                    load_image('green_image.png'),
                    load_image('blue_image.png'))
save_as(combined, 'combined.png')
```

Once you have convinced yourself that your `combine()` filter is working, run it again using the .JPG versions. You should see blurring in the images. It doesn't mean that the filters are incorrect. It simply means that some information is lost every time an Image object in memory is compressed and saved on disk as a jpeg file. This is not an issue that your filters have to fix. You can avoid blurring by (1) using the .PNG format which is not compressed or (2) keeping your images in memory (i.e. in variables) and not saving them in files.

Milestone 1 Lab 3 (P3)

The third lab is a time to wrap-up the first portion of the project, by documenting the project. In addition, if needed, you can demo your final code submission, if not already done so.

P3 Task 1: Milestone 1's Final Team Code

As a team, package your code together.

1. Create a new Python file
    - Required Filename: **xxx_P2_channel-filters.py, where xxx is your CULearn team identifier**
2. Copy-paste every one's code into this one file.
3. Review the project-wide expectations for code submissions. For instance, the file must have a header with the team identifier, and each function must have the name of the author.
4. Make sure it still compiles and runs.
5. Submit the team's code. One submission for the whole team.

P3 Task 2: The Project Report (aka Engineering Solution)

As a team, you are required to write a **Project Report** that describes your understanding of the problem, and how you will write a program to solve this problem.

Required Filename: xxx_ProjectReport.pdf (or .docx, but not .txt) where xxx is your team identifier

The following are posted on CULearn to help you complete this task

- **Project Report Template** – An empty skeleton of a document with primary headings. Hopefully, you can simply fill in the blanks.
- **Project Report Sample** – A lab re-written in the style of the project report.
- **Project Report Rubric** – How your report will be marked.

Even though you will be writing the project report at the end of Milestone 1, your statement of the project goal must describe the whole project (until the end of Milestone 3). You must read the provided resources to consider the project *as a whole*, including looking ahead to the end product (not only the tasks done as part of Milestone 1). We acknowledge that this requirement may seem odd and unfathomable, but it is an exercise in extrapolating from your tasks in the first Milestone to how you will design and work as a team for the remainder of the project.

It is strongly recommended that you prepare a draft of your report **before** coming to the lab, so that you can get some early feedback from your TA.

Milestone 1 Marking Rubrics

P1 Task 1 – Team Contract

Satisfactory (2)	Marginal (1)	Unsatisfactory (0)
The contract is complete. The answers to the questions are thoughtful. The document is tidy and well-written	The contract is incomplete, or the answers are mostly superficial and short. The document is untidy and shows a lack of care, making it difficult to read	It is late or the contract is missing

P2 Task 3 – Individual Filter and Corresponding Test Function

This submission will be marked on-the-fly during the lab using the same rubric as the Tutorial Labs.

Preconditions

A student's work will not be marked (i.e. will earn UNSATISFACTORY=0) if any of the following are true:

- the files are not in the proper .py format (no .pdf, zip or py.py file extensions)
- the requested files are either not present or don't have the correct name
- the author of the file is not identified in the heading of the file.
- if applicable, they use programming techniques that have been explicitly discouraged from the lab for teaching reasons (e.g. don't use dictionaries)

Documentation Expectations

- All functions headers must have type annotations on all arguments, and on the return type
- All functions must have a good docstring describing WHAT the functions does, not HOW.
    - Good Example: Returns the average of the three integer parameter
    - Poor Example: The function returns the average of three numbers
    - Poorest Example: The function adds the three numbers and returns their average
- All functions must have a thoughtful set of tests in the proper shell format.

Coding Expectations

- Variables and functions use meaningful names, in the proper style
- Constants must be used instead of hard-coded values, in the proper style

P3 Task 2 – Project Report

The rubric for the project report is posted on CULearn along with the template and sample report.