Laboratory 11 — Processing image data

Topics covered:

- Creating tuples; assigning values using tuples; returning tuples
- Processing and manipulating images using a provided image library
- Creating functions that manipulate images

PREPARATION

Lab attendance is compulsory. You will receive 1 mark for being present at the start of the lab and staying at least until the tutor has finished introducing the lab and has signed your attendance sheet.

EXERCISES

The following exercises must be completed during your allocated laboratory time. You must show your work to the laboratory tutor who will sign you off when the work is completed correctly.

EXERCISE 11.1

[2 marks] Inside the Ex11.1.py file, complete the `get_square_cube()` function which takes an integer as a parameter and returns 2 values by returning a tuple. Your function should return the square and the cube of the given parameter as a tuple.

Argument: a number (integer)
Returns: the square and the cube of the given parameter (tuple)

Sample output:

```
>>> get_square_cube(3)
(9, 27)
```

EXERCISE 11.2

[3 marks] Inside the Ex11.2.py file, complete the `negative()` function which takes a pixel as a parameter and inverts each colour intensity — light colours become dark, and vice versa.

Argument: a pixel
Task: Subtract each component of the pixel from 255.

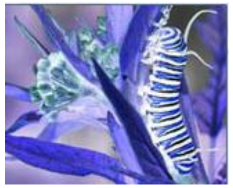
```
from ImageLibrary import *

def negative(pixel):
    #complete this



pict = makePicture("caterpillarSmall.jpg")
for pixel in getPixels(pict):
    negative(pixel)
show(pict)
```

Sample output:



EXERCISE 11.3

[4 marks] Inside the Ex11.3.py file, complete the `to_grayscale()` function which takes a pixel as a parameter and converts the colour to grayscale. Your function should convert the pixel to a shade of gray by setting each of the 3 components (red, green and blue) of the pixel to the same gray value. The standard formula for the gray level of an RGB pixel is:

$$\boldsymbol{gray = 0.299 * red + 0.587 * green + 0.114 * blue}$$

Compute the gray value using the above formula, then assign that (same) gray value to each component of the pixel. This formula comes from video standards, and approximates the way the eye perceives the various colours.

Argument: a pixel
Task: Convert each RGB component of the pixel to the value calculated using the above formula.

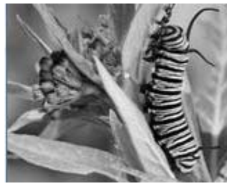
```
from ImageLibrary import *

def to_grayscale(pixel):
    #complete this



pict = makePicture("caterpillarSmall.jpg")
for pixel in getPixels(pict):
    to_grayscale(pixel)
show(pict)
```

Sample output:



Homework Exercises

The following exercises are due at **11:30pm, Friday 6th June 2014**. Include all the exercises in a single module (file), named “Lab11_Homework.py”. Your file must include a docstring at the top of the file containing your name, UPI and ID number. Submit the file containing your exercises using the Assignment Dropbox.

Exercise 11.4

Consider the following list of student names and marks.

```
["Peter,9", "Paul,8", "dick,6", "mohan,9", "sita,7", "ajay,3"]
```

Write the `create_marks_dict()` function which takes a list of strings containing student names and marks as a parameter. Your job is to return a dictionary of (mark, [list of students who got that mark]).

Argument: a list of strings
Returns: a dictionary of (mark, [list of students who got that mark])
Extra knowledge:
You can use the following tuple assignment statement to get each student name and mark from a string:

```
student_name, mark = element.split(",")
```

Sample Test:

```
>>> create_marks_dict(["Peter,9", "Paul,8", "dick,6", "mohan,9",
"sita,7", "ajay,3"])
{'7': ['sita'], '6': ['dick'], '9': ['Peter', 'mohan'], '8': ['Paul'],
'3': ['ajay']}
```

Exercise 11.5

Write the `count_rgb_frequency()` function which takes the filename of an image as a parameter and returns a dictionary of the frequency of pixel colours in the image. The `getPixels()` function in the provided "ImageLibrary.py" file can be used to convert an image into a list of pixels, then the `getRed()`, `getGreen()` and `getBlue()` functions can be used to create an RGB tuple for each pixel in the list. We can then easily count frequencies, ending up with a dictionary of pixel values (as tuples) and frequencies, like so:

Sample Test:

```
>>> count_rgb_frequency("diagonal.png")
{(0, 0, 255): 50, (255, 255, 255): 2450}
```

Arguments: an image file name
Returns: A dictionary of pixel colours and frequencies

Advanced Exercises (optional)

Exercise 11.6

[0 marks] Write the `flip()` function which takes an image filename as a parameter and turns the picture upside-down. It does this by exchanging pixel values. Each pixel in row zero must be exchanged with the corresponding pixel in row height-1. Pixels in row 1 must exchange with pixels in row height-2, and so forth. You can write a for loop which will go through the top half of the rows. The pixels in each row must then be exchanged with those in height - row - 1.

Arguments: an image filename
Shows: the image after it has been flipped.

Sample output:

ASSESSMENT

Name: ______________________________

Lab day and time: ______________________________

Check list for laboratory exercises (to be completed by Lab tutor)

On time: ☐ (1 mark)
Exercise 11.1: ☐ (2 marks)
Exercise 11.2: ☐ (3 marks) Teaching Assistant: ______________
Exercise 11.3: ☐ (4 marks)
Total mark: ____________________/10 Lab tutor: ______________________

Marking Scheme

Marks	Feedback
0.5	Include a docstring at the top of the file
0.5	Include all the exercises in a single file
1	Use good, meaningful variable names.
1	The `create_marks_dict()` function is defined correctly.
1	Your function returns a dictionary.
2	Well done! Your function passed test cases 1 and 2
1	The `count_rgb_frequency()` function is defined correctly.
1	Your function returns a dictionary.
2	Well done! Your function passed test cases 1 and 2.