Assignment 9: Image Processing: duplicatefinder

EC602 Design by Software

Fall 2017

Contents

1 Introduction **1**
- 1.1 Assignment Goals . . . 1
- 1.2 Group Size . . . 2
- 1.3 Due Date . . . 2
- 1.4 Submission . . . 2
- 1.5 Points . . . 2

2 Image Processing **2**

3 The assignment: duplicatefinder.py **3**
- 3.1 The output specification . . . 3
- 3.2 The SHA256 digest . . . 4
- 3.3 Grading Rubric . . . 4

4 Hints **4**
- 4.1 Imports . . . 4
- 4.2 Array Methods Used . . . 5

5 Checker **5**

1 Introduction

1.1 Assignment Goals

The goals for this assignment are to

- design an efficient image categorization algorithm using python data structures and standard library modules
- write Python code that passes a static code-style analyzer.
- write Python code that is also elegant.
- learn about array operations and image processing

- learn about the hash library `hashlib`

1.2 Group Size

For this assignment, the maximum group size is 3.

1.3 Due Date

This assignment is due 2017-12-05 at midnight.

1.4 Submission

You can submit here: `duplicatefinder` submit link

Since you have sufficient materials to check your own program on example suites, the online checker will implement penalties for incorrect submissions. The penalties will be announced on Wednesday when the checker is complete.

The submitted programs will be checked for correctness, style, and elegance online, but the efficiency test will be performed offline to keep the load on the server reasonable.

1.5 Points

This assignment is worth 10 points.

2 Image Processing

The basic task we explore and solve in this assignment is image categorization.

You have a large collection of images: suppose these images represent

- microbes
- molecules
- human faces
- signatures
- DNA strands

and so forth. Our task is to find all the matching objects from the large collection of images.

3 The assignment: duplicatefinder.py

Write a program to examine all the PNG image files from a directory, and determine which ones are images of the same object. There is exactly one object in each image, photographed over a perfectly white background image. Any non-white pixel is part of the object.

You may assume that all the images are RBG with values for each color ranging from 0 to 255 (hence the arrays can be stored using `uint8` data types).

The objects are rigid and two-dimensional. The objects are all photographed from the same distance, so the same object appears the same size in all photographs it appears in. However, the object can be rotated by multiples of 90 degrees in the image, and can appear anywhere within the image. All objects are completely contained within the image. One other complication: due to camera inconsistencies, some of the objects appear as their mirror image.

The best way to understand the image specifications is to examine the files of in the examples: duplicate_examples.zip

Your program should find all the PNG files in the current directory: these are all files with the extension `png`. Every PNG file has exactly one number in the filename, and these numbers are guaranteed to be unique among all image files in the directory.

The program should create a file to record the results of the image categorization process, and print out the SHA256 message digest for this file.

The file name is specified on the command line, like this:

```
python duplicatefinder.py name_of_file_to_create
```

3.1 The output specification

The file is formatted as follows:

- each line contains the filenames of identical objects
- the filenames in the line are sorted in increasing numerical order (of the number in the filename)
- the lines in the file are sorted in increasing numerical order based on the first filename in each line.
- the new line character must be `\n` and NOT `\r\n`

Here is an example of the file format:

```
rasberry0.png cher2.png grap5.png grape6.png
ban1.png rasber8.png bana9.png
grape3.png
pear4.png rasber7.png
```

3.2 The SHA256 digest

Using the facilities of the `hashlib` module, print out the SHA256 digest for the file created. This process will allow me to provide you with information about the file without providing the file itself.

In Unix-based systems, the SHA256 digest (and others) can be calculated using

```
shasum -a 256 answers.txt
```

but in the assignment, you must use the `hashlib` module.

For this reason, importing `os` is only allowable in the following manner:

```
from os import listdir
```

3.3 Grading Rubric

Out of 100, your grade on this assignment will be assessed as follows:

- 40 points: your code passes the unittester for specifications
- 10 points: your code passes the unittester for style (as determined by pep8).
- 10 points: your pylint score
- 20 points: your code will be evaluated for elegance, as determined by code length. Code length is defined as the number of words. Shorter programs will score higher.
- 20 points: your code will be evaluated for efficiency, as determined by the time to process and categorize the image files in the six folrder, which will be dominated by the folder `four`. Faster programs will score higher.

Your python program must be called `duplicatefinder.py`

4 Hints

4.1 Imports

Here is the import section of one solution:

```
from os import listdir
import re
from skimage.io import imread
import numpy as np
import hashlib
```

4.2 Array Methods Used

The following methods/properties of the `ndarray` class may be useful:

```
sum
copy
transpose
flatten
nonzero
shape
```

5 Checker

The checker is here: duplicate_checker.py

The checker automatically downloads the following file: duplicate_checker_dirs.zip

It has six directories (`zero` through `five`) in which the tests are run.

Note that the grading will be done offline.

The local grade report you receive when running the checker is unofficial.