

Assignment 1

GIF Images

1. Submission Guidelines

Deadline:	11:59 PM on Thursday 10 November 2022
	This deadline includes an extension granted due to planned tube strikes.
Submission procedure:	Submit only one file labelled <code>gif.py</code> through blackboard (via TurnItIn)
Version requirement:	Your code must run using Python 3.10.5 on a PC
Allowable import modules:	None

2. Overview

The Graphics Interchange Format (GIF) image file is used to store images on a computer. It is known for compressing image files into a smaller size without losing any image quality. For this assignment, you are tasked with writing code that can be used to open a GIF file and access its contents.

This is a real-world assignment, meaning that you are working on a practical problem that requires that you find additional information to complete the task. The official GIF documentation is on blackboard (`spec-gif89a.txt`) or at <https://www.w3.org/Graphics/GIF/spec-gif89a.txt>. We expect that all students will use the internet and textbook resources to learn concepts needed for this assignment. There will be many concepts, in-built functions, and other aspects of Python coding, which have not been directly taught in labs or lectures. It is up to you to fill in these gaps in knowledge by engaging in both the course and self-learning.

3. GIF Format

You must adhere to the GIF89a specifications, which includes the ability to open GIF files with both the GIF89a and GIF87a formats. You do not need to implement any GIF extensions and can assume that there will be no GIF extensions in my test GIF image cases. You can assume that only a global colour table is used and that it is present in the GIF file. You can assume that the file will not contain a local colour table. You can also assume that, when reconstructing your image, we will use GIF files that only have a sequential order (no interlaced order will be used) with all sort flags set to 0. You can assume that the GIF files tested have only 1 image in it.

When you reach the Table Based Image data, you will need to extract the image from the LZW data compression format, which has been slightly modified for use in GIF. The algorithm is well described on different websites:

<https://www.w3.org/Graphics/GIF/spec-gif89a.txt>
<https://en.wikipedia.org/wiki/GIF>
<http://giflib.sourceforge.net/whatsinagif/index.html>

We have uploaded `squares.gif` as a sample image for you to open and access. However, your code should be able to handle other GIF files. We will certainly test other `.gif` files to ensure that your code adheres to the GIF documentation (`spec-gif89a.txt`).

4. GIF Functions

Write 6 functions that can be used with the `import` system. We will type `import gif` at the top of our own script (eg, `script.py`) and then use your functions to perform tasks. For different arguments passed into the function we will assess the returned values for each function. All code examples described here assume that `import gif` has been executed prior to each function call. While we are providing example code and a GIF file on blackboard, we will mark your code using a far wider range of test code and test GIF files, which are not being provided to you. It is important that you test your code under a wider range of conditions.

data, info = **load_file**(*file_name*):

Take the file name of a .gif file and return a data object and a description line.

The *file_name* argument should be of type `str` and is the path to the .gif file.

The returned *data* should be of type `bytes`. If the GIF file was found, its entire content should be transferred into *data*. If the file was not found, return an object of type `bytes`, which is empty.

The returned *info* should be of type `str`. If the file was opened, return the *file_name*. If the file was not found, return 'file not found'.

header = **extract_header**(*data*):

Extract the GIF header from *data*.

The *data* argument should be of type `bytes` and obtained using `load_file()`.

The returned *header* should be of type `str`.

width, height, gc_fl, cr, sort_fl, gc_size, bcolour_i, px_ratio = **extract_screen_descriptor**(*data*):

Extract the screen descriptors from *data*.

The *data* argument should be of type `bytes` and obtained using `load_file()`.

The returned *width, height, gc_fl, cr, sort_fl, gc_size, bcolour_i, and px_ratio* should be of type `int`. The values returned should be as specified in the GIF documentation.

gc_map = **extract_global_colour_table**(*data*):

Extract the global colour map from *data*.

The *data* argument should be of type `bytes` and obtained using `load_file()`.

The returned *gc_map* should contain the global colour table of type list. Each item in *gc_map* should be of type `int`. *gc_map* should be a 2-dimensional array with each row representing a different colour in the table and each column representing the colour, such as red, green, and blue.

left, top, width, height, lc_fl, itl_fl, sort_fl, res, lc_size = **extract_image_descriptor**(*data*):

Extract the image descriptors from *data*.

The *data* argument should be of type `bytes` and obtained using `load_file()`.

The returned *left, top, width, height, lc_fl, itl_fl, sort_fl, res, and lc_size*, should be of type `int`. The values returned should be as specified in the GIF documentation.

img = **extract_image**(*data*)

Extract the image from *data*.

The *data* argument should be of type `bytes` and obtained using `load_file()`.

The returned *img* should be a 3-dimensional array containing the decompressed GIF image. The first dimension should be rows, the second dimension should be columns, and the third dimension should be the colours, such as red, green, blue. Each element within *img* should be of type `int`. For example, `print(type(img[0][0][0]))` should produce the output `<class 'int'>`.

5. Coding rules

Do not declare any variables in the global space of `gif.py`. The global space definitions must have the 6 functions requested. A `main()` function could be used if you so desire, but is not required. The global space may have additional function definitions that can be called by the 6 functions requested in this assignment. No module is allowed for this assignment.

6. Marking criteria

We will mark your submitted `gif.py` code according to the following categories:

- 70 marks: Implementation and evidence of coding knowledge
- 20 marks: Coding efficiency
- 10 marks: Coding style and commenting

We will use `import gif` near the top of our script to test your code. We will run several test conditions against each of your classes and methods. We will investigate what was assigned in your attribute variables. We will test far more test cases than the examples we have included in this assignment sheet.

7. Sample Code

See separate Appendix I PDF for sample code. This will be released on Thursday 3 Nov.