Assignment – PNG Images

Version 1.00

1. Submission Guidelines

Deadline:	9:00 AM on Friday 13 December
Submission procedure:	Submit only one file labelled `png.py` through blackboard (via TurnItIn)
Version requirement:	Your code must run using **Python 3.11.4 on a PC**
Allowable import modules:	`math` and `zlib` only. No other modules may be imported.

2. Overview

The Portable Network Graphics (PNG) format is a format for storing images. For this assignment, you are tasked with writing code that can be used to open a PNG file, store its contents in the program, and save modified PNG image files.

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 PNG documentation is available online.

https://www.w3.org/TR/png/

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.

Keep track of the version number of this assignment and ensure that your code meets the latest specifications by the deadline. This assignment challenge is very similar to a real-world situation where specifications evolve with the person writing code for the specifications. We will be updating the assignment descriptor throughout the week to add clarity, remove ambiguities, and fix descriptor errors. Post on the Ed Discussion board any clarification requests, ambiguities, or descriptor errors. The Teaching Assistants will then raise this with the instructor who will then update the version of the assignment and announce it via email.

3. PNG Format

You must adhere to the PNG version specified in the hyperlink above. However, you will only need to implement only a small subset of the total PNG specifications.

Implement only the following chunks:

- `IHDR`
- `IDAT`
- `IEND`
- Other chunks. There may be other chunks present in the PNG image file. For this assignment, you do not need to implement those other chunks. However, your programme should run smoothly even if they are present. In other words, ignore the other chunks if they are present in the image.

You will only need to read and write images that have the following image header (`IHDR`) specifications:

- A bit depth of 8
- A colour type of 2
- A compression method of 0
- A filter method of 0
- An interlace method of 0

When you read the example image (`brainbow.png`), you should be able to find these values within the file.

We have uploaded `brainbow.png` as a sample image for you to open and use. Your code should be able to handle other PNG files, including images with different widths and heights. We may also test very large images. We will certainly test other `.png` files to ensure that your code is not hard coded to produce the sample outputs.

4. Class PNG

Create a class named `PNG` that will allow users to interact with PNG files.

4.1. Attributes

Objects of type `PNG` should have the following variable attributes:

`data` A value of type `bytes`.

This represents the data bytes from the png file. The default values should be `b''`

`info` A value of type `str`.

This represents information about the png file. The default value should be `''` and should become either the name of the file or `'file not found'`.

`width` A value of type `int`.

This represents the width of the image in pixels. The default value should be `0`.

`height` A value of type `int`.

This represents the height of the image in pixels. The default value should be `0`.

`bit_depth` A value of type `int`.

This represents the bit depth of the image data stored in the png file. The default value should be `0`.

`color_type` A value of type `int`.

This represents the color type of the image data stored in the png file. The default value should be `0`.

`compress` A value of type `int`.

This represents the compression method of the png file. The default value should be `0`.

`filter` A value of type `int`.

This represents the filter method of the png file. The default value should be `0`.

`interlace` A value of type `int`.

This represents the interlacing method of the png file. The default value should be `0`.

`img` A value of type `list`.

This represents a 3-dimensional array where the 1st dimension is the image rows, the 2nd dimension is the image columns, and the 3rd dimension is the red, green, or blue values.

The red, green, and blue values should be of type `int` and should range between `0` and `255` only.

The default value should be an empty list.

4.2. Methods

For this assignment, you may assume that all arguments put into each method are of the correct type.

`__init__(self)`

Return values: None

Arguments: A reference to itself, `self`.

Implementation: This method should initialise all of the above member attributes to their default values.

10 marks for implementation

`load_file(self, file_name)`

Return values: None

Arguments: A reference to itself (`self`) and a file name (`file_name`) of type `str`.

Implementation: The `load_file()` method should open the file specified by `file_name` and store all of the file's data into the member attribute `data`.

If the file is found, then assign `file_name` into the info member attribute.

If the file is not found and a `FileNotFoundError` exception is raised when trying to open the file, then assign `'file not found'` into the info member attribute.

10 marks for implementation

`valid_png(self)`

Return values: True or False

Arguments: A reference to itself, `self`.

Implementation: Reads the PNG signature and returns whether the correct values were found or not.

Returns `True` if the PNG signature was correctly specified in the file. Returns `False` if the PNG signature was not correctly specified in the file.

10 marks for implementation

`read_header(self)`

Return values: None

Arguments: A reference to itself, `self`.

Implementation: Reads the image header chunk (`IHDR`) and updates the relevant member attributes.

Update the `width`, `height`, `bit_depth`, `color_type`, `compress`, `filter`, and `interlace` attributes according to the information stored in the image header chunk.

10 marks for implementation

`read_chunks(self)`

Return values: None

Arguments: A reference to itself, `self`.

Implementation: Reads through all the chunks and updates the `img` attribute.

Read through the remaining chunks following the PNG signature and `IHDR` chunk. This includes `IDAT` and `IEND` chunk types that include the image data. There may be other

chunk types present in the PNG file, but gracefully ignore those chunk types. Your program should still run smoothly even if those chunk types are present in the PNG file.

One of the challenges of this method is being able to extract the image data from the `IDAT` chunk(s). After extracting the image, store it in the member attribute `img`. `img` represents a 3-dimensional array where the 1st dimension is the image rows, the 2nd dimension is the image columns, and the 3rd dimension is the red, green, or blue values. The red, green, and blue values should be of type `int` and should range between `0` and `255` only.

20 marks for implementation

10 marks for efficiency

`save_rgb(`*`self, file_name, rgb_option`*`)`

Return values: None

Arguments: A reference to itself, `self`, a file name (`file_name`) of type `str`, and a saving option (`rgb_option`) of type `int`.

Implementation: Saves the red, green, or blue channels of the `img` attribute into a PNG file named `file_name` as determined by the setting of `rgb_option`.

When `rgb_option` is set to 1, store the red channel of the image into the PNG file.

When `rgb_option` is set to 2, store the green channel of the image into the PNG file.

When `rgb_option` is set to 3, store the blue channel of the image into the PNG file.

We should be able to open your PNG file using a regular image viewing application.

10 marks for implementation

10 marks for efficiency

5. Coding style and commenting

Ensure that your code is readable and well commented. We will be marking according to spacing, naming, comments, length, and general readability.

10 marks for coding style and commenting

6. Coding rules

Only the PNG class definition and imports to the `math` and `zlib` modules should be in the global space of `png.py`. There should be no other global variables or definitions. You are free to define methods within the PNG class, which we will not call. A `main()` function could be used if you so desire, but it is not required. Only the `math` and `zlib` modules are allowed for this assignment. No other module is allowed.

7. Marking criteria

We will mark your submitted `png.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 import your `png` module near the top of our script to test your code. We will run several test conditions against each of your classes, including its attributes and methods. After running each method, we will investigate what was assigned in your attribute variables. So, the type, length, etc. of your attribute variables must perfectly match how we defined them in this specification. We will test far more test cases than the single example we have included in this assignment sheet.

8. Sample Code

The following code…

```
import png

def main():

   print('PNG')
   print()

   image = png.PNG()

   print('data:       ', image.data)
   print('info:       ', image.info)
   print('width:      ', image.width)
   print('height:     ', image.height)
   print('bit_depth:  ', image.bit_depth)
   print('color_type: ', image.color_type)
   print('compress:   ', image.compress)
   print('filter:     ', image.filter)
   print('interlace:  ', image.interlace)
   print('img:        ', image.img)
   print()

   image.load_file('brainbow.png')

   print(image.data[0:100].hex())
   print(type(image.data))
   print(len(image.data))
   print(image.info)
   print(type(image.info))
   print(len(image.info))
   print()

   if image.valid_png():
      print('This is a PNG file')
   else:
      print('This is not a PNG file')
   print()

   image.read_header()

   print('info:       ', image.info)
   print('width:      ', image.width)
   print('height:     ', image.height)
   print('bit_depth:  ', image.bit_depth)
   print('color_type: ', image.color_type)
   print('compress:   ', image.compress)
   print('filter:     ', image.filter)
   print('interlace:  ', image.interlace)
   print('img:        ', image.img)
   print()

   image.read_chunks()
   for i in range(5):
      for j in range(6):
         print(image.img[i][j], end=' ')
      print()

   image.save_rgb('brainbow_r.png', 1)

if __name__ == '__main__':
   main()
```

Should produce the following output:

```
PNG

data:         b''
info:
width:        0
height:       0
bit_depth:    0
color_type:   0
compress:     0
filter:       0
interlace:    0
img:          []

89504e470d0a1a0a0000000d49484452000000492000002e108020000008be9191d00000009704859730000b1300000b1301
009a9c18000000206348524d00007a25000080830000f9ff000080e9000075300000ea6000003a980000176f925fc5460027
<class 'bytes'>
2588130
brainbow.png
<class 'str'>
12

This is a PNG file

info:         brainbow.png
width:        1170
height:       737
bit_depth:    8
color_type:   2
compress:     0
filter:       0
interlace:    0
img:          []

[25, 0, 4] [21, 0, 0] [3, 0, 0] [0, 6, 0] [0, 11, 3] [0, 10, 2]
[21, 0, 0] [21, 0, 0] [4, 0, 0] [0, 9, 0] [0, 12, 4] [0, 10, 2]
[0, 0, 2] [1, 1, 3] [3, 4, 0] [5, 1, 0] [5, 0, 0] [6, 1, 0]
[1, 1, 3] [3, 3, 5] [1, 2, 0] [3, 0, 0] [4, 0, 0] [8, 3, 0]
[0, 5, 5] [0, 5, 5] [1, 0, 0] [7, 0, 0] [13, 0, 0] [15, 0, 0]
```

The original `brainbow.png` image: