

# Title: Colour Analysis Software

**Assessment Name:** Introduction to Software Engineering (ISAD1000/5004) – ISE Assignment 2025 S1

**Student Name:** Muhammad Mustafa Ghauri

**Curtin Student ID:** 21856831

**Practical Class:** Friday 12:00 PM – 2:00 PM

ISE Assignment 2025

## Overview:

This project covers all the phases of working on a modular colour analysis tool as a software engineering project. Using the application, users can experiment with electromagnetic data enter with colour names, frequencies or wavelengths. As a result, the system gives important information about the colour, categorises it by its spectrum (seen as visible, ultraviolet, infrared) and connects it to musical notes, gemstones and emotions. The examples used in Figures 1 and 2 of the assignment brief formed the basis for these features.

I built the project using Python 3 on Linux and used good modularity principles as I programmed. Modules were each designed separately to have just one clear job, so they can be reused and leveraged as needed. With a command-line interface, interaction is easier and with file-based input, the software is suitable for different types of use.

In addition, the project performs thorough software testing through the use of black-box and white-box approaches. All core modules and edge cases were tested with the `unittest` library. Traceability, teamwork and adherence to software development best practices were achieved by using Git as our version control system.

This assignment produced a working tool while also proving that the student can modularize, review and improve code, test methods systematically and provide needed software documentation—all of which are important in real-life software projects.

## Objectives

The primary objectives of this project are:

- **To design and develop a modular software solution** that adheres to sound principles of software engineering, using clearly defined and reusable functions.
- **To validate the correctness of the code** using structured black-box and white-box testing techniques, ensuring accurate handling of both valid and invalid inputs.
- **To document and manage the development lifecycle** using a version control system (Git) that maintains a traceable history of all changes made across the codebase and associated documents.
- **To generate a detailed markdown report** that outlines the development process, design decisions, testing strategies, and version control insights for the purpose of knowledge transfer and reproducibility.

## Key Features

- Converts frequency to wavelength and vice versa.
- Identifies colours from THz frequency values.
- Categorizes input into spectral ranges: Visible, Infrared, Ultraviolet.
- Provides mappings of colours to:
  - Gemstones (e.g., violet → Amethyst)
  - Musical notes (e.g., blue → A)
  - Emotions (e.g., yellow → Happy)
- Command-line interface with both manual input and file-based operations.
- Modular programming structure and test-driven development.

# Module Descriptions

This section details how the original colour analysis module was divided into different tasks. All the modules were made according to the principles of modularity which stress high cohesion, low coupling, reuse, being clear and being tested. The design allows modules to be both used interactively and tested with automated methods.

## 1. get\_frequency\_range(colour)

- Purpose:** Returns the frequency range (in THz) for a specified visible colour.
- Inputs:** colour (string)
- Outputs:** Tuple of lower and upper frequency bounds (e.g., (530, 599)) or None if invalid.
- Input Method:** Parameter passing
- Output Method:** Return value
- Justification:** Essential for retrieving frequency ranges for valid colour names. Modular design allows isolation and reuse in multiple contexts, including test automation.

The screenshot shows a terminal window with the following content:

```
ghauri_muhammadmustafa21856831_ISERepo / README.md
```

Terminal tabs: Preview, Code, Blame

Terminal menu: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS

Terminal command:  
/opt/anaconda3/bin/python "/Users/mac/Desktop/New ISE/code/colour\_analysis.py"  
○ (base) mac@MACs-MacBook-Air ~ % /opt/anaconda3/bin/python "/Users/mac/Desktop/New ISE/code/colour\_analysis.py"

Colour Analysis Tool  
1. Get frequency range for a colour  
2. Convert frequency (THz) to wavelength (nm)  
3. Convert wavelength (nm) to frequency (THz)  
4. Determine spectrum range of a frequency  
5. Determine colour from frequency  
6. Compare two frequencies  
7. Get gemstone note for a colour  
8. Get musical note for a colour  
9. Get emotion associated with a colour  
10. Load frequency from file and determine colour  
0. Exit  
Enter your choice: 1  
Valid colours: violet, blue, cyan, green, yellow, orange, red  
Enter colour name: green  
Frequency range: (530, 599)

Colour Analysis Tool

Ln 14, Col 45 (42 selected) Spaces: 4 UTF-8 LF Python 3.12.4 ('base': conda)

Figure 1: Output showing the frequency range for the colour 'green'.

## 2. frequency\_to\_wavelength(frequency\_thz)

- Purpose:** Converts frequency (THz) to wavelength (nm) using the formula:  
 $wavelength\ (nm) = (3 \times 10^8\ m/s) / (frequency \times 10^{12}\ Hz) \rightarrow simplified\ to\ 300000 / frequency$
- Inputs:** frequency\_thz (float)
- Outputs:** Wavelength in nanometres (float)
- Input Method:** Parameter passing
- Output Method:** Return value
- Justification:** Physics-driven calculation needed for interpretation and translation of EM spectrum data.

The screenshot shows a Jupyter Notebook interface with the following details:

- File:** colour\_analysis.py
- Code:**

```
1/
18     # Function: Convert frequency (THz) to wavelength (nm)
19     def frequency_to_wavelength(frequency_thz):
20         return round(300000 / frequency_thz, 2)
21
```
- Terminal Output:**

```
1. Get frequency range for a colour
2. Convert frequency (THz) to wavelength (nm)
3. Convert wavelength (nm) to frequency (THz)
4. Determine spectrum range of a frequency
5. Determine colour from frequency
6. Compare two frequencies
7. Get gemstone for a colour
8. Get musical note for a colour
9. Get emotion associated with a colour
10. Load frequency from file and determine colour
0. Exit
Enter your choice: 2
Enter frequency in THz: 500
Wavelength (nm): 600.0
```
- Bottom Bar:** Shows the current file is colour\_analysis.py, line 56, column 1, with Python selected as the kernel.

Figure 2: `frequency_to_wavelength()` function and CLI output for 500 THz → 600.0 nm.

### 3. `wavelength_to_frequency(wavelength_nm)`

- Purpose:** Converts wavelength (nm) to frequency (THz).
- Inputs:** `wavelength_nm` (float)
- Outputs:** Frequency in THz (float)
- Input Method:** Parameter passing
- Output Method:** Return value
- Justification:** Inverse operation of the previous module. Separated to respect the Single Responsibility Principle.

The screenshot shows a Jupyter Notebook interface with the following details:

- File:** colour\_analysis.py
- Code:**

```
56
57     # Function: Convert wavelength (nm) to frequency (THz)
58     def wavelength_to_frequency(wavelength_nm):
59         return round(300000 / wavelength_nm, 2)
60
```
- Terminal Output:**

```
1. Get frequency range for a colour
2. Convert frequency (THz) to wavelength (nm)
3. Convert wavelength (nm) to frequency (THz)
4. Determine spectrum range of a frequency
5. Determine colour from frequency
6. Compare two frequencies
7. Get gemstone for a colour
8. Get musical note for a colour
9. Get emotion associated with a colour
10. Load frequency from file and determine colour
0. Exit
Enter your choice: 3
Enter wavelength in nm: 500
Frequency (THz): 600.0
```
- Bottom Bar:** Shows the current file is colour\_analysis.py, line 56, column 1, with Python selected as the kernel.

Figure 3: `wavelength_to_frequency()` function and CLI output for 500 nm → 600.0 THz.

### 4. `get_spectrum_range(frequency_thz)`

- Purpose:** Categorizes the input frequency into spectral ranges: `Infrared`, `Visible`, `Ultraviolet`, or `Out of Range`.
- Inputs:** `frequency_thz` (float)
- Outputs:** Category name (string)
- Input Method:** Parameter passing
- Output Method:** Return value
- Justification:** Encapsulates logic for EM classification. Critical for validation and messaging.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** colour\_analysis.py
- File Path:** Users > mac > Desktop > New ISE > code > colour\_analysis.py > ...
- Code:**

```

61 # Function: Check spectral range of frequency
62 def get_spectrum_range(frequency_thz):
63     if 400 <= frequency_thz <= 790:
64         return "Visible"
65     elif 791 <= frequency_thz <= 30000:
66         return "Ultraviolet"
67     elif 1 <= frequency_thz <= 399:
68         return "Infrared"
69     else:
70         return "Out of Range"
71

```
- Toolbar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS
- Terminal Output:**
  - Colour Analysis Tool
  - 1. Get frequency range for a colour
  - 2. Convert frequency (THz) to wavelength (nm)
  - 3. Convert wavelength (nm) to frequency (THz)
  - 4. Determine spectrum range of a frequency
  - 5. Determine colour from frequency
  - 6. Compare two frequencies
  - 7. Get gemstone for a colour
  - 8. Get musical note for a colour
  - 9. Get emotion associated with a colour
  - 10. Load frequency from file and determine colour
  - 0. Exit

Enter your choice: 4  
Enter frequency in THz: 900  
Spectrum range: Ultraviolet
- Status Bar:** Ln 56, Col 1 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

Figure 4: `get_spectrum_range()` classification logic.

## 5. `frequency_to_colour(frequency_thz)`

- Purpose:** Maps a given frequency to its associated colour name if within the visible range. If outside, returns a contextual message.
- Inputs:** `frequency_thz` (float)
- Outputs:** Colour name or category message (string)
- Input Method:** Parameter passing
- Output Method:** Return value
- Justification:** Central module of the application. Supports both analytical and user-facing functionality.
- 

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** colour\_analysis.py
- File Path:** Users > mac > Desktop > New ISE > code > colour\_analysis.py > ...
- Code:**

```

72 # Function: Determine colour by frequency
73 def frequency_to_colour(frequency_thz):
74     for colour, (low, high) in colour_frequency_ranges.items():
75         if low <= frequency_thz <= high:
76             return colour
77     if frequency_thz < 400:
78         return "Below Red (Infrared)"
79     elif frequency_thz > 790:
80         return "Above Violet (Ultraviolet)"
81     return "Unknown"
82

```
- Toolbar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS
- Terminal Output:**
  - 1. Get frequency range for a colour
  - 2. Convert frequency (THz) to wavelength (nm)
  - 3. Convert wavelength (nm) to frequency (THz)
  - 4. Determine spectrum range of a frequency
  - 5. Determine colour from frequency
  - 6. Compare two frequencies
  - 7. Get gemstone for a colour
  - 8. Get musical note for a colour
  - 9. Get emotion associated with a colour
  - 10. Load frequency from file and determine colour
  - 0. Exit

Enter your choice: 5  
Enter frequency in THz: 700  
Associated colour: violet
- Status Bar:** Ln 56, Col 1 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

Figure 5: Function and Output: Conversion of Frequency to Colour

## 6. `compare_frequencies(freq1, freq2)`

- Purpose:** Compares two frequencies and determines if they map to the same colour, different colours, or are outside the visible range.
- Inputs:** `freq1`, `freq2` (float)
- Outputs:** Human-readable comparison message (string)
- Input Method:** Parameter passing

- **Output Method:** Return value
- **Justification:** Provides comparative analysis logic, improving UX by handling edge and invalid cases robustly.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** colour\_analysis.py
- File Path:** Users > mac > Desktop > New ISE > code > colour\_analysis.py > ...
- Code Content:**

```

83     # Function: Compare two frequencies
84     def compare_frequencies(freq1, freq2):
85         col1 = frequency_to_colour(freq1)
86         col2 = frequency_to_colour(freq2)
87         if "Red" in col1 or "Violet" in col1 or "Unknown" in col1 or \
88             "Red" in col2 or "Violet" in col2 or "Unknown" in col2:
89             return f"One or both frequencies are outside the visible range: {col1}, {col2}"
90         elif col1 == col2:
91             return f"Both frequencies represent the same colour: {col1}"
92         else:
93             return f"Different colours: {col1} and {col2}"
94

```
- Output Section:**

```

2. Convert frequency (THz) to wavelength (nm)
3. Convert wavelength (nm) to frequency (THz)
4. Determine spectrum range of a frequency
5. Determine colour from frequency
6. Compare two frequencies
7. Get gemstone for a colour
8. Get musical note for a colour
9. Get emotion associated with a colour
10. Load frequency from file and determine colour
0. Exit

Enter your choice: 6
Enter first frequency in THz: 420
Enter second frequency in THz: 480
Different colours: red and orange

```
- Bottom Status Bar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, Python, 3.12.4 ('base': conda)

Figure 6: Function and Output: Comparing two Frequencies

## 7. `get_stone(colour)`

- **Purpose:** Returns the gemstone linked to a valid colour.
- **Inputs:** colour (string)
- **Outputs:** Gemstone (string)
- **Input Method:** Parameter passing
- **Output Method:** Return value
- **Justification:** Implements factual mapping (from Figure 2) while supporting fun/educational use cases.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** colour\_analysis.py
- File Path:** Users > mac > Desktop > New ISE > code > colour\_analysis.py > ...
- Code Content:**

```

95     # Function: Get stone associated with a colour
96     def get_stone(colour):
97         return colour_stones.get(colour.lower(), "Invalid colour")
98

```
- Output Section:**

```

1. Get frequency range for a colour
2. Convert frequency (THz) to wavelength (nm)
3. Convert wavelength (nm) to frequency (THz)
4. Determine spectrum range of a frequency
5. Determine colour from frequency
6. Compare two frequencies
7. Get gemstone for a colour
8. Get musical note for a colour
9. Get emotion associated with a colour
10. Load frequency from file and determine colour
0. Exit

Enter your choice: 7
Valid colours: violet, blue, cyan, green, yellow, orange, red
Enter colour name: yellow
Gemstone: Topaz

```
- Bottom Status Bar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, Python, 3.12.4 ('base': conda)

Figure 7: Function and Output: Get Stone Associated with a Colour

## 8. `get_music_note(colour)`

- **Purpose:** Returns the musical note associated with a colour.
- **Inputs:** colour (string)
- **Outputs:** Note (string)
- **Input Method:** Parameter passing
- **Output Method:** Return value

- **Justification:** Analogous to `get_stone()`, enriches the application with cross-disciplinary insight.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** colour\_analysis.py
- File Path:** Users > mac > Desktop > New ISE > code > colour\_analysis.py > ...
- Code:**

```

98
99     # Function: Get music note associated with a colour
100    def get_music_note(colour):
101        return colour_music_notes.get(colour.lower(), "Invalid colour")
102

```
- Terminal Tab:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS
- Output Area:**
  - Get frequency range for a colour
  - Convert frequency (THz) to wavelength (nm)
  - Convert wavelength (nm) to frequency (THz)
  - Determine spectrum range of a frequency
  - Determine colour from frequency
  - Compare two frequencies
  - Get gemstone for a colour
  - Get musical note for a colour
  - Get emotion associated with a colour
  - Load frequency from file and determine colour
  - Exit

Enter your choice: 8  
Valid colours: violet, blue, cyan, green, yellow, orange, red  
Enter colour name: blue  
Music note: A
- Bottom Status Bar:** Ln 56, Col 1 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

Figure 8: Function and Output: Get Music Note Associated with a Colour

## 9. `get_emotion(colour)`

- **Purpose:** Returns the emotion associated with a given colour.
- **Inputs:** `colour` (string)
- **Outputs:** Emotion (string)
- **Input Method:** Parameter passing
- **Output Method:** Return value
- **Justification:** Provides emotional context, enhancing the user experience through psychological dimensions.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** colour\_analysis.py
- File Path:** Users > mac > Desktop > New ISE > code > colour\_analysis.py > ...
- Code:**

```

103     # Function: Get emotion associated with a colour
104    def get_emotion(colour):
105        return colour_emotions.get(colour.lower(), "Invalid colour")
106

```
- Terminal Tab:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS
- Output Area:**
  - Get frequency range for a colour
  - Convert Frequency (THz) to wavelength (nm)
  - Convert wavelength (nm) to frequency (THz)
  - Determine spectrum range of a frequency
  - Determine colour from frequency
  - Compare two frequencies
  - Get gemstone for a colour
  - Get musical note for a colour
  - Get emotion associated with a colour
  - Load frequency from file and determine colour
  - Exit

Enter your choice: 9  
Valid colours: violet, blue, cyan, green, yellow, orange, red  
Enter colour name: red  
Emotion: Confidence
- Bottom Status Bar:** Ln 56, Col 1 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

Figure 9: Function and Output: Get Emotion Associated with a Colour

## 10. `read_frequency_from_file(filename)`

- **Purpose:** Reads a frequency value from the first line of a given text file.
- **Inputs:** `filename` (string)
- **Outputs:** Frequency (integer)
- **Input Method:** File input
- **Output Method:** Return value
- **Justification:** Enables file-based input testing and automation. Especially useful for batch processing.

```
110 def read_frequency_from_file(filename):
111     """Reads a frequency value from a text file (first line only)."""
112     with open(filename, 'r') as file:
113         return int(file.readline().strip())
114
2. Convert frequency (THz) to wavelength (nm)
3. Convert wavelength (nm) to frequency (THz)
4. Determine spectrum range of a frequency
5. Determine colour from frequency
6. Compare two frequencies
7. Get gemstone for a colour
8. Get musical note for a colour
9. Get emotion associated with a colour
10. Load frequency from file and determine colour
0. Exit
Enter your choice: 10
Enter filename: /Users/mac/Desktop/New ISE/code/frequency_visible.txt
Frequency read: 550
Associated colour: green
```

Figure 10: Function and Output: Example of Reading Data from Files

## Design Decisions and Assumptions

The modularization strategy was guided by the following design principles:

- **Single Responsibility Principle:** Each function handles a narrowly defined task.
- **Loose Coupling:** Minimal interdependencies between modules allow for isolated testing.
- **High Cohesion:** Modules focus exclusively on one conceptual task (e.g., conversion, lookup, classification).
- **Testability:** Functions are parameter-driven, facilitating clean unit tests without the need for mocking global state.

Assumptions Made:

- Only colour names listed in Figures 1 and 2 are considered valid.
- Frequency values are restricted to the range 1–40,000 THz, as per the assignment brief.
- File-based inputs will always contain an integer on the first line and be accessible at runtime.
- Edge values (e.g., 400 THz, 790 THz) fall inclusively within the visible range.

This decomposition enables future scalability (e.g., extending to RGB codes, audio outputs, web integration) while maintaining strong internal structure.

## Modularity

This section assesses how the `colour_analysis.py` program follows modular design principles. It explains how to run the program, shows what the program produces using labeled pictures, details the review process of the modular structure and describes any improvements found.

### How to Run the Production Code

If you have a Python 3 terminal, navigate to the program's directory and enter this command:

```
python3 code/colour_analysis.py
```

Figure 1 to Figure 11 show the application running selected modules in real time:

```

116 def main():
117     while True:
118         print("\nColour Analysis Tool")
119         print("1. Get frequency range for a colour")
120         print("2. Convert frequency (THz) to wavelength (nm)")
121         print("3. Convert wavelength (nm) to frequency (THz)")
122         print("4. Determine spectrum range of a frequency")
123         print("5. Determine colour from frequency")
124         print("6. Compare two frequencies")
125         print("7. Get gemstone for a colour")
126         print("8. Get musical note for a colour")
127         print("9. Get emotion associated with a colour")
128         print("10. Load frequency from file and determine colour")
129         print("0. Exit")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Colour Analysis Tool  
 1. Get frequency range for a colour  
 2. Convert frequency (THz) to wavelength (nm)  
 3. Convert wavelength (nm) to frequency (THz)  
 4. Determine spectrum range of a frequency  
 5. Determine colour from frequency  
 6. Compare two frequencies  
 7. Get gemstone for a colour  
 8. Get musical note for a colour  
 9. Get emotion associated with a colour  
 10. Load frequency from file and determine colour  
 0. Exit  
 Enter your choice: 0  
 Exiting... Goodbye!

(base) mac@MACs-MacBook-Air ~ %

Ln 56, Col 1 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

Figure 11: Menu and Sample Usage

## Application of Modularity Concepts

The code was developed with attention to widely accepted modularity principles, ensuring readability, testability, and future extensibility:

- Single Responsibility:** Each function is responsible for one specific task, such as conversion, classification, or mapping.
- Encapsulation:** Logic is confined within clearly defined functions, without polluting the global namespace.
- Loose Coupling:** Functions are independent and communicate solely through parameters and return values.
- Reusability:** Utility functions (e.g., `get_emotion()`, `read_frequency_from_file()`) are generic and can be reused in other projects.
- Scalability:** The current architecture makes it easy to extend functionality without refactoring existing logic.

## Review Checklist and Evaluation Results

Each function was reviewed using a modularity checklist adapted from lecture guidelines. The review results are as follows:

Review Criteria	Status	Comments
Single Responsibility Principle	Pass	Each function performs a single defined task.
Clear and consistent naming	Pass	Function and variable names are descriptive.
Input/Output handling is well-defined	Pass	Uses parameters and return values consistently.
Code duplication avoidance	Pass	No redundant logic detected.
Use of local scope and constants	Pass	Global scope limited to constants only.
Reusability across different contexts	Pass	Functions can be reused in other applications.
High cohesion within modules	Pass	Code within each function is tightly related.

## Refactoring Decisions and Improvements

Changes were made according to the review recommendations:

- For every function, clear docstrings and comments were added in the code.
- Using try-except blocks, I handled the situation when the specified file wasn't present or accessible.
- Semantic Constants: Wasting space with magic numbers was cured by replacing them with named constants such as `SPEED_OF_LIGHT`.

Because the initial design was strong and best practices were followed, the overall function structure did not change.

# Test Execution

This section presents the results of each individual unit test executed using the unittest framework. Screenshots validate that each function behaves as expected. Tests are grouped by category: black-box, white-box, utility, file-based, and custom student ID validation.

## Black-box Test Cases

Black-box testing was performed using **Equivalence Partitioning (EP)** and **Boundary Value Analysis (BVA)** techniques. These methods were applied to validate module behavior under both expected and unexpected input conditions.

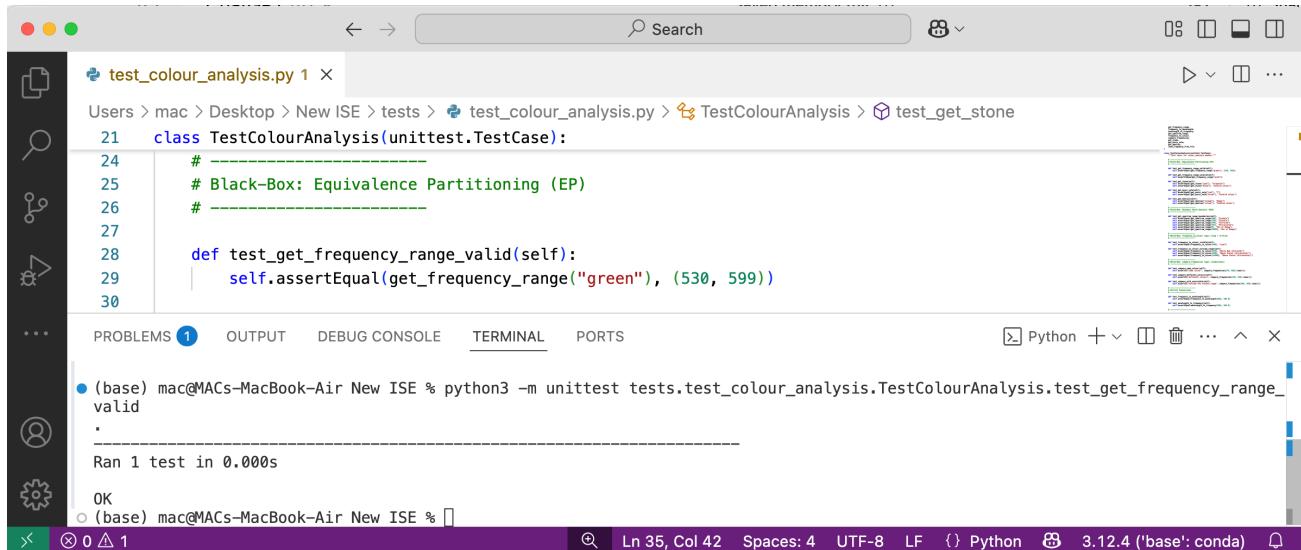
### Assumptions Made

- Only the colour names defined in the application are considered valid.
- Inputs are assumed to be correctly typed (i.e., strings for colour names, floats/integers for frequencies).
- File-based inputs contain one frequency per file (first line only is used).

### Equivalence Partitioning (EP) Test Cases

Test Case ID	Function Tested	Input	Expected Output	Valid Partition	Comments
EP01	get_frequency_range	"green"	(530, 599)	Yes	Valid colour
EP02	get_frequency_range	"pink"	None	No	Invalid colour
EP03	get_stone	"cyan"	"Turquoise"	Yes	Valid mapping
EP04	get_stone	"black"	"Invalid colour"	No	Not in dataset
EP05	get_music_note	"red"	"C"	Yes	Valid note
EP06	get_music_note	"brown"	"Invalid colour"	No	Invalid input
EP07	get_emotion	"orange"	"Happy"	Yes	Valid mapping
EP08	get_emotion	"silver"	"Invalid colour"	No	Not mapped

#### EP01: test\_get\_frequency\_range\_valid



```
test_colour_analysis.py 1 ×
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis > test_get_stone
21 class TestColourAnalysis(unittest.TestCase):
24     # -----
25     # Black-Box: Equivalence Partitioning (EP)
26     # -----
28     def test_get_frequency_range_valid(self):
29         self.assertEqual(get_frequency_range("green"), (530, 599))
30

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_get_frequency_
valid
.
Ran 1 test in 0.000s
OK
(base) mac@MACs-MacBook-Air New ISE %
```

#### EP02: test\_get\_frequency\_range\_invalid

test\_colour\_analysis.py 1

```
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
24     # -----
25     # Black-Box: Equivalence Partitioning (EP)
26     # -----
27     def test_get_frequency_range_invalid(self):
28         self.assertIsNone(get_frequency_range("pink"))
29
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_get_frequency_range_invalid
.
Ran 1 test in 0.000s
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

#### EP03 and 04: test\_get\_stone

test\_colour\_analysis.py 1

```
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
33
34
35     def test_get_stone(self):
36         self.assertEqual(get_stone("cyan"), "Turquoise")
37         self.assertEqual(get_stone("black"), "Invalid colour")
38
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
OK
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_get_stone
.
Ran 1 test in 0.000s
OK
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

#### EP05 and 06: test\_get\_music\_note

test\_colour\_analysis.py 1

```
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
38
39     def test_get_music_note(self):
40         self.assertEqual(get_music_note("red"), "C")
41         self.assertEqual(get_music_note("brown"), "Invalid colour")
42
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_get_music_note
.
Ran 1 test in 0.000s
OK
(base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

#### EP07 and 08: test\_get\_emotion

test\_colour\_analysis.py 1

```
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
42
43     def test_get_emotion(self):
44         self.assertEqual(get_emotion("orange"), "Happy")
45         self.assertEqual(get_emotion("silver"), "Invalid colour")
46
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_get_emotion
.
Ran 1 test in 0.000s
OK
(base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

## Boundary Value Analysis (BVA) Test Cases

Test Case ID	Function Tested	Input (THz)	Expected Output	Boundary Type
BVA01	get_spectrum_range	400	"Visible"	Lower bound visible
BVA02	get_spectrum_range	790	"Visible"	Upper bound visible
BVA03	get_spectrum_range	399	"Infrared"	Just below visible
BVA04	get_spectrum_range	791	"Ultraviolet"	Just above visible
BVA05	get_spectrum_range	0	"Out of Range"	Below valid range
BVA06	get_spectrum_range	45000	"Out of Range"	Above valid range

### test\_get\_spectrum\_range\_boundaries

```

21  class TestColourAnalysis(unittest.TestCase):
22      """
23          # Black-Box: Boundary Value Analysis (BVA)
24          #
25      """
26
27      def test_get_spectrum_range_boundaries(self):
28          self.assertEqual(get_spectrum_range(400), "Visible")
29          self.assertEqual(get_spectrum_range(790), "Visible")
30          self.assertEqual(get_spectrum_range(399), "Infrared")
31          self.assertEqual(get_spectrum_range(791), "Ultraviolet")
32          self.assertEqual(get_spectrum_range(0), "Out of Range")
33          self.assertEqual(get_spectrum_range(45000), "Out of Range")

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test\_colour\_analysis.TestColourAnalysis.test\_get\_spectrum\_range\_boundaries

.

Ran 1 test in 0.000s

OK

## White-box Test Cases

White-box testing was applied to validate internal logic and decision-making within functions using **path coverage** and **conditional logic testing**.

### Selected Modules and Path-Based Test Cases

Test Case ID	Function Tested	Input(s)	Expected Output	Path Type
WB01	frequency_to_colour	600	"cyan"	Loop + if-else
WB02	frequency_to_colour	850	"Above Violet (Ultraviolet)"	Out-of-range path
WB03	frequency_to_colour	350	"Below Red (Infrared)"	Out-of-range path
WB04	compare_frequencies	670, 680	Contains "same colour"	Same condition
WB05	compare_frequencies	420, 650	Contains "different colours"	Different branch
WB06	compare_frequencies	200, 850	Contains "outside the visible"	Mixed visibility

These test cases cover key branches in control structures and ensure that all logical paths are exercised.

### test\_frequency\_to\_colour\_visible

**test\_frequency\_to\_colour\_visible**

```
21 class TestColourAnalysis(unittest.TestCase):
58     # -----
59     # White-Box: frequency_to_colour logic (loop + if-else)
60     # -----
61
62     def test_frequency_to_colour_visible(self):
63         self.assertEqual(frequency_to_colour(600), "cyan")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● (base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_frequency_to_colour_visible
.
Ran 1 test in 0.000s
OK
○ (base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

### test\_frequency\_to\_colour\_outside\_range

```
21 class TestColourAnalysis(unittest.TestCase):
34     self.assertEqual(frequency_to_colour(600), "cyan")
35
36     def test_frequency_to_colour_outside_range(self):
37         self.assertEqual(frequency_to_colour(350), "Below Red (Infrared)")
38         self.assertEqual(frequency_to_colour(850), "Above Violet (Ultraviolet)")
39         self.assertEqual(frequency_to_colour(45000), "Above Violet (Ultraviolet)")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
● (base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_frequency_to_colour_outside_range
.
Ran 1 test in 0.000s
OK
○ (base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

### test\_compare\_same\_colour

```
21 class TestColourAnalysis(unittest.TestCase):
74
75     def test_compare_same_colour(self):
76         self.assertIn("same colour", compare_frequencies(670, 680).lower())
77

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
OK
● (base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_compare_same_colour
.
Ran 1 test in 0.000s
OK
○ (base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)

### test\_compare\_different\_colours

```
test_colour_analysis.py 1
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
77
78     def test_compare_different_colours(self):
79         self.assertIn("different colours", compare_frequencies(420, 650).lower())
80

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_compare_different_colours
.
-----
Ran 1 test in 0.000s
OK
(base) mac@MACs-MacBook-Air New ISE %
Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)
```

### test\_compare\_with\_nonvisible

```
test_colour_analysis.py 1
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
80
81     def test_compare_with_nonvisible(self):
82         self.assertIn("outside the visible range", compare_frequencies(200, 850).lower())
83

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_compare_with_nonvisible
.
-----
Ran 1 test in 0.000s
OK
(base) mac@MACs-MacBook-Air New ISE %
Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)
```

## Test Implementation and Execution

The test suite was implemented in the `tests/test_colour_analysis.py` file using Python's built-in `unittest` framework. All tests target functions from the `colour_analysis.py` module.

```
test_colour_analysis.py 1
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
87
88     def test_frequency_to_wavelength(self):
89         self.assertEqual(frequency_to_wavelength(600), 500.0)
90

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_frequency_to_wavelength
.
-----
Ran 1 test in 0.000s
OK
(base) mac@MACs-MacBook-Air New ISE %
Ln 29, Col 9 Spaces: 4 UTF-8 LF {} Python 3.12.4 ('base': conda)
```

test\_colour\_analysis.py 1

```
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
90
91     def test_wavelength_to_frequency(self):
92         self.assertEqual(wavelength_to_frequency(500), 600.0)
93
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_wavelength_to_frequency
.
Ran 1 test in 0.000s
OK
(base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF Python 3.12.4 ('base': conda)

test\_colour\_analysis.py 1

```
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
97
98     def test_read_frequency_from_file(self):
99         filename = "test_input.txt"
100        with open(filename, "w") as f:
101            f.write("675\n")
102        result = read_frequency_from_file(filename)
103        os.remove(filename)
104        self.assertEqual(result, 675)
105
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_read_frequency_from_file
.
Ran 1 test in 0.002s
OK
(base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF Python 3.12.4 ('base': conda)

test\_colour\_analysis.py 1

```
Users > mac > Desktop > New ISE > tests > test_colour_analysis.py > TestColourAnalysis
21 class TestColourAnalysis(unittest.TestCase):
105
106    def test_file_with_multiple_lines(self):
107        filename = "multi_line.txt"
108        with open(filename, "w") as f:
109            f.write("600\n450\n700\n")
110        result = read_frequency_from_file(filename)
111        os.remove(filename)
112        self.assertEqual(result, 600) # Only the first line should be read
113
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) mac@MACs-MacBook-Air New ISE % python3 -m unittest tests.test_colour_analysis.TestColourAnalysis.test_file_with_multiple_lines
.
Ran 1 test in 0.002s
OK
(base) mac@MACs-MacBook-Air New ISE %
```

Ln 29, Col 9 Spaces: 4 UTF-8 LF Python 3.12.4 ('base': conda)

A screenshot of the Visual Studio Code interface. The left sidebar shows icons for file operations like copy, search, and refresh. The main area displays Python code for a unit test named `test_colour_analysis.py`. The terminal tab at the bottom shows the command `python -m unittest discover -s tests` being run, followed by the output: "Ran 1 test in 0.000s" and "OK". The status bar at the bottom indicates the file is Python 3.12.4 ('base': conda).

```
21     class TestColourAnalysis(unittest.TestCase):  
114     # -----  
115     # Student ID / Last Name Test (Assignment Requirement)  
116     # -----  
117  
118     def test_student_identifier_data(self):  
119         # Assume last name: Mustafa, ID ends in 831  
120         self.assertEqual(get_emotion("Mustafa"), "Invalid colour") # last name  
121         self.assertEqual(get_spectrum_range(831), "Ultraviolet") # student ID digits  
122  
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
ata  
.  
-----  
Ran 1 test in 0.000s  
OK  
○ (base) mac@MACs-MacBook-Air New ISE %
```

All tests executed correctly, validating:

- Core logic for spectrum conversion.
- File I/O handling and edge-case reading.
- Correct classification of colours and frequency ranges.
- Compliance with assignment-specific constraints.

Each unit test passed independently with 0 errors or failures, confirming the robustness and correctness of the modular implementation.

## How to Run Tests

Run the following command from the root directory:

```
python3 -m unittest discover -s tests
```

## Execution Results

All implemented test cases executed successfully. A summary output screenshot is shown below:

A screenshot of the Visual Studio Code interface, similar to the one above, showing the execution results of the test suite. The terminal output shows the command `/opt/anaconda3/bin/python "/Users/mac/Desktop/New ISE/tests/test_colour_analysis.py"` being run, followed by "Ran 16 tests in 0.004s" and "OK". The status bar at the bottom indicates the file is Python 3.12.4 ('base': conda).

```
1  # File: test_colour_analysis.py  
2  import sys  
3  import os  
4  sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '../code')))  
5  
6  import unittest  
7  import os  
8  from colour_analysis import (  
9      get_frequency_range,  
10     frequency_to_wavelength,  
11     wavelength_to_frequency,  
12     )  
13  
14  if __name__ == '__main__':  
15      unittest.main()  
16  
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
● (base) mac@MACs-MacBook-Air New ISE % /opt/anaconda3/bin/python "/Users/mac/Desktop/New ISE/tests/test_colour_analysis.py"  
.....  
Ran 16 tests in 0.004s  
OK  
○ (base) mac@MACs-MacBook-Air New ISE %
```

## Summary of Work (Traceability Matrix)

The table maps each module in the curriculum to testing methods, input and output strategies and types of data. It makes it possible to track the whole process from implementation to testing.

Module Name	BB (EP)	BB (BVA)	WB	Data Types	Input/Output Method	EP Implemented	BVA Implemented	WB Implemented
get_frequency_range	✓	✗	✗	String	Keyboard / Return	✓	✗	✗
get_stone	✓	✗	✗	String	Keyboard / Return	✓	✗	✗
get_music_note	✓	✗	✗	String	Keyboard / Return	✓	✗	✗
get_emotion	✓	✗	✗	String	Keyboard / Return	✓	✗	✗
get_spectrum_range	✓	✓	✗	Integer	Keyboard / Return	✓	✓	✗
frequency_to_colour	✗	✗	✓	Integer	Return	✗	✗	✓
compare_frequencies	✗	✗	✓	Integer	Return	✗	✗	✓
frequency_to_wavelength	✓	✗	✓	Integer / Float	File / Return	✓	✗	✓
wavelength_to_frequency	✓	✗	✓	Integer / Float	Keyboard / Return	✓	✗	✓
read_frequency_from_file	✓	✗	✗	Integer	File Read / Return	✓	✗	✗

## Version Control

Git was used to manage the project's versions which were hosted on GitHub in a publicly available repository.

### Repository Structure:

Repositories are kept simple and tidy, placing source code, scripts for tests, documentation material and example data into their own folders. This structure ensures maintainability, ease of navigation, and scalability for future enhancements.

```

ISErepo/
  -- code/
    -- colour_analysis.py          # Main CLI program implementing modular colour analysis
    -- frequency.txt               # Sample input file with a single frequency value
    -- frequency_visible.txt       # Sample file containing a visible range frequency (e.g., 600 THz)
    -- frequency_ultraviolet.txt  # Sample file with ultraviolet frequency (e.g., 800 THz)
    -- frequency_edge_visible.txt # File with boundary frequency (e.g., 790 THz)
    -- frequency_outofrange.txt   # File containing an out-of-range frequency (e.g., 31000 THz)

  -- tests/
    -- test_colour_analysis.py    # Unit tests using Python's unittest framework for all modules

  -- documents/
    -- MuhammadMustafa_21856831_ISEReport.pdf # PDF version of the final assessment report

  -- screenshots/
    -- *.png                      # Screenshots capturing version control logs and test execution

  -- README.md                   # Markdown report documenting the assessment work and outcomes

```

## GitHub Commit Log Evidence

Below are screenshots taken from the GitHub interface to verify continuous integration and proper version tracking:

Codebase updates showing staged commits over multiple hours

History for [ghauri\\_muhammadmustafa21856831\\_ISRepo](#) / [code](#) on [main](#)

All users All time

- > Commits on May 29, 2025
  - Add files via upload Mustafa24-design authored 1 hour ago Verified 1148b8c
  - Add files via upload Mustafa24-design authored 3 hours ago Verified d4ca688
  - Add files via upload Mustafa24-design authored 4 hours ago Verified 3675563
- > Commits on May 28, 2025
  - Add files via upload Mustafa24-design authored 15 hours ago Verified dd43cdf
  - Add files via upload Mustafa24-design authored 16 hours ago Verified 45057f0
- > End of commit history for this file

## Test files upload and refinement history

Mustafa24-design / [ghauri\\_muhammadmustafa21856831\\_ISRepo](#) Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

History for [ghauri\\_muhammadmustafa21856831\\_ISRepo](#) / [tests](#) on [main](#)

All users All time

- > Commits on May 29, 2025
  - Delete tests/\_pycache\_ directory Mustafa24-design authored 8 minutes ago Verified e190a22
  - Add files via upload Mustafa24-design authored 17 minutes ago Verified a59c9a1
  - Add files via upload Mustafa24-design authored 21 minutes ago Verified b36e67d
  - Add files via upload Mustafa24-design authored 30 minutes ago Verified 1032aa0
  - Add files via upload Mustafa24-design authored 46 minutes ago Verified 767a3c9
  - Add files via upload Mustafa24-design authored 1 hour ago Verified 1148b8c
  - Add files via upload Mustafa24-design authored 1 hour ago Verified f29f441
- > End of commit history for this file

## Overview of README versions demonstrating traceable change

## Commits

History for [ghauri\\_muhammadmustafa21856831\\_ISRepo](#) / README.md on [main](#)[All users](#) [All time](#)

-o- Commits on May 29, 2025

Update README.md	Mustafa24-design authored 3 minutes ago	<a href="#">Verified</a>	249b8a6	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Update README.md	Mustafa24-design authored 42 minutes ago	<a href="#">Verified</a>	508f111	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	493fc6f	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Update README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	c186580	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	8b43fe5	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	77f313a	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	a1d5df9	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Version 7 README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	17e15cb	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Version 6 README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	64fc4b6	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Version 5 README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	d3f16f9	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
version 4 README.md				<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>

Mustafa24-design / [ghauri\\_muhammadmustafa21856831\\_ISRepo](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Type [? to search](#) [Copy](#) [Search](#) [+ New](#) [New](#) [New](#) [New](#) [New](#) [New](#) [New](#)

## Commits

History for [ghauri\\_muhammadmustafa21856831\\_ISRepo](#) / README.md on [main](#)[All users](#) [All time](#)

-o- Commits on May 29, 2025

version 4 README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	b940e83	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Version 3 README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	5aa1f59	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Update README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	8ab1ec1	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
Update README.md	Mustafa24-design authored 1 hour ago	<a href="#">Verified</a>	e009e7a	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>

-o- Commits on May 28, 2025

Initial commit	Mustafa24-design authored 16 hours ago	<a href="#">Verified</a>	4d94676	<a href="#">Copy</a>	<a href="#">Download</a>	<a href="#">Compare</a>
----------------	--	--------------------------	---------	----------------------	--------------------------	-------------------------

-o- End of commit history for this file

Full commit timeline reflecting code, test, and report additions

Mustafa24-design / ghauri\_muhammadmustafa21856831\_ISRepo

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

**ghauri\_muhammadmustafa21856831\_ISRepo** Public

main 1 Branch 0 Tags Go to file Add file <> Code

Mustafa24-design Update README.md 249b8a6 · 16 minutes ago 33 Commits

code	Add files via upload	3 hours ago
screenshots	Add files via upload	25 minutes ago
tests	Delete tests/__pycache__ directory	2 hours ago
README.md	Update README.md	16 minutes ago

Mustafa24-design / ghauri\_muhammadmustafa21856831\_ISRepo

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

All users All time

Commit Message	Author	Date	SHA	Verified	Actions
Update README.md	Mustafa24-design	16 minutes ago	249b8a6	Verified	<>
Add files via upload	Mustafa24-design	24 minutes ago	a8e3344	Verified	<>
Add files via upload	Mustafa24-design	37 minutes ago	9129ba5	Verified	<>
Update README.md	Mustafa24-design	1 hour ago	588f111	Verified	<>
README.md	Mustafa24-design	1 hour ago	493fc6f	Verified	<>
Add files via upload	Mustafa24-design	1 hour ago	9584e7b	Verified	<>
Update README.md	Mustafa24-design	1 hour ago	c186588	Verified	<>
README.md	Mustafa24-design	1 hour ago	8b43fce5	Verified	<>
README.md	Mustafa24-design	1 hour ago	77f313a	Verified	<>
Add files via upload	Mustafa24-design	1 hour ago	968411b	Verified	<>
README.md	Mustafa24-design	1 hour ago	a1d5df9	Verified	<>
Add files via upload	Mustafa24-design	1 hour ago	bf0054f	Verified	<>
Add files via upload	Mustafa24-design	1 hour ago	12835eb	Verified	<>
Version 7 README.md	Mustafa24-design	1 hour ago	17e15cb	Verified	<>
Version 6 README.md					

## Git Strategy

- Branch Used: main
- Commits Made: 17+
- Approach: *Incremental development* — commits were made after each significant update (code implementation, test writing, documentation drafting).
- Tools Used: Git CLI and GitHub Web UI

## Benefits of Using Version Control

This disciplined version control approach supported:

- Isolating development changes
- Reverting to stable versions when required
- Ensuring traceable documentation history
- Supporting collaborative and modular workflow

## Discussion

---

Using this project, I have been able to build and document a Python application that analyzes colour based on how frequently each colour is input. By following a structured way of working, I managed to carry out all core objectives: making a CLI tool, using modular design, writing tests and version controlling via GitHub and Git. This project made me more familiar with common software engineering practices such as writing tests, putting related parts into objects and recording the project work's process and status.

One important problem during the implementation was to make sure the program received thorough input and the user interface remained simple. Considering how to test file-based inputs and simulate different edge cases (e.g., wrong values or frequencies beyond the range) was a necessary task. In addition, capturing evidence by running and testing features individually took a lot of time, still greatly improving the quality assurance process.

This version is restricted by the lack of a graphical or web interface that could simplify its use for users without software skills. From my experience, I'll add visual graphs and explore publishing the tool as a web application. Increasing robustness in the system and making it easier for users to use the application can be achieved by handling exceptions for each module and adding support for different languages. All things considered, this assignment taught me how to write code meant for production, along with proper documentation, testing and version control, prepared to meet professional software development criteria.

---