Lab Project
(color Detection)
On

DATA MINING AND ANALYSIS (CS 655C)

A practical file submitted in partial fulfillment of

the requirements for the award of

Bachelor of Engineering

IN COMPUTER SCIENCE AND ENGINEERING

Submitted by

MANTASH SINGH (Roll no:CO17335)

GAURAV KAUSHAL (Roll no:CO17322)

Under the supervision of

Dr. Varun Gupta

CHANDIGARH COLLEGE OF ENGINEERING TECHNOLOGY

(DEGREE WING)

Government Institute under Chandigarh (UT) Administration,
Affiliated to Panjab University Chandigarh
Sector-26, Chandigarh. PIN-160019

Jan-June 2020

Table of Contents

CHAPTER – 1

1.1 PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.[31] Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. The Python 2 language, i.e. Python 2.7, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it. With Python 2's end-of-life, only Python 3.5 and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

1.2 Features and philosophy

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspectoriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.[56]

The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:[57]

- Beautiful is better than ugly.
- Explicit is better than implicit. • Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is *not* considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity.] When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas*.

1.3 Libraries

Python's large standard library, commonly cited as one of its greatest strengths,[100] provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals,[101] manipulating regular expressions, and unit testing.

Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation wsgiref follows PEP 333), but most modules are not. They are specified by their code, internal documentation, and test suites. However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of November 2019, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 200,000 packages with a wide range of functionality, including:

- Graphical user interfaces
- Web frameworks
- Multimedia
- Databases
- Networking
- Test frameworks
- Automation
- Web scraping[104]
- Documentation
- System administration

- Scientific computing
- Text processing
- Image processing

1.4 Development environments

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statements sequentially and receives results immediately.

Other shells, including IDLE and IPython, add further abilities such as improved auto-completion, session state retention and syntax highlighting.

As well as standard desktop integrated development environments, there are Web browser-based IDEs; SageMath (intended for developing science and math-related Python programs); PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial Python IDE emphasizing scientific computing.

CHAPTER – 2
Python packages

2.1 Python packages

A package is basically a directory with Python files and a file with the name __init__.py. This means that every directory inside of the Python path, which contains a file named __init__.py, will be treated as a package by Python. It's possible to put several modules into a Package.

Packages are a way of structuring Python’s module namespace by using "dotted module names". A.B stands for a submodule named B in a package named A. Two different packages like P1 and P2 can both have modules with the same name, let's say A, for example. The submodule A of the package P1 and the submodule A of the package P2 can be totally different.

A package is imported like a "normal" module.

2.2 PACKAGES WE USED

2.2.1 OpenCV

OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

OpenCV introduces a new set of tutorials which will guide you through various functions available in OpenCV-Python. This guide is mainly focused on OpenCV 3.x version (although most of the tutorials will work with OpenCV 2.x also).

A prior knowledge on Python and Numpy is required before starting because they won't be covered in this guide. Especially, a good knowledge on Numpy is must to write optimized codes in OpenCVPython.

OpenCV

2.2.2 NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code

- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

2.2.3 PANDAS

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Library features

- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.

- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation[4] and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

CHAPTER – 3
ABOUT THE PROJECT

3.1 OUR PROJECT – COLOR DETECTION

In this color detection Python project, we are going to build an application through which you can automatically get the name of the color by clicking on them. So for this, we will have a data file that contains the color name and its values. Then we will calculate the distance from each color and find the shortest one.

3.2 What is Colour Detection?

Colour detection is the process of detecting the name of any color. Simple isn't it? Well, for humans this is an extremely easy task but for computers, it is not straightforward. Human eyes and brains work together to translate light into color. Light receptors that are present in our eyes transmit the signal to the brain. Our brain then recognizes the color. Since childhood, we have mapped certain lights with their color names. We will be using the somewhat same strategy to detect color names.

3.3 The Dataset

Colors are made up of 3 primary colors; red, green, and blue. In computers, we define each color value within a range of 0 to 255. So in how many ways we can define a color? The answer is **256*256*256 = 16,581,375**. There are approximately 16.5 million different ways to represent a color. In our dataset, we need to map each color's values with their corresponding names. But don't worry, we don't need to map all the values. We will be using a dataset that contains RGB values with their corresponding names. The CSV file for our dataset has been taken from this link:

https://github.com/codebrainz/color-names/blob/master/output/colors.csv

The colors.csv file includes 865 color names along with their RGB and hex values.

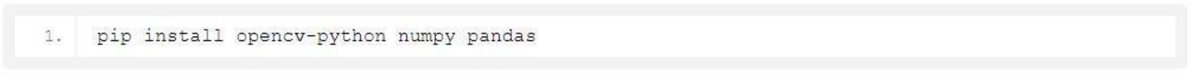
3.3.1 Colors Dataset

The colors.csv file is a data set we used here , it includes **865** color names along with their RGB and hex values.

3.4 Prerequisites

Before starting with this Python project with source code, we should be familiar with the computer vision library of Python that is ***OpenCV*** and ***Pandas*.**

OpenCV, Pandas, and numpy are the Python packages that are necessary for this project in Python. To install them, simply run this pip command in your terminal:

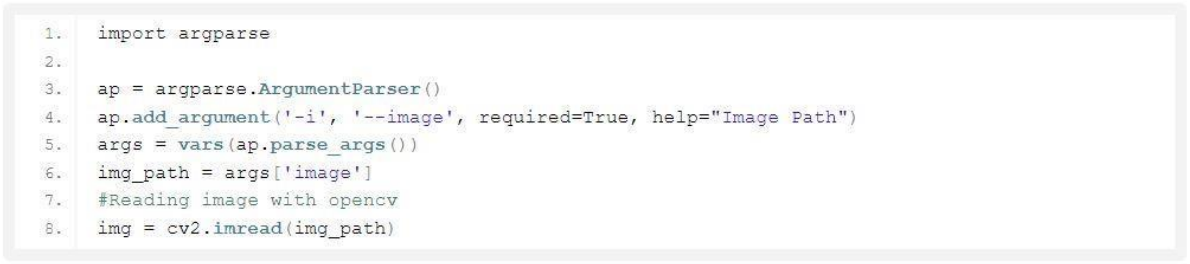


```
pip install opencv-python numpy pandas
```

CHAPTER – 4 STEPS FOR BULIDING COLOR DETECTION PROJECT

4.1 Taking an image from the user

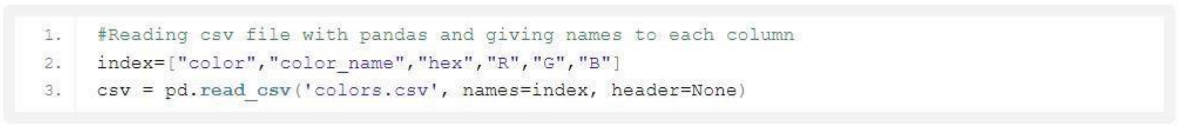
We are using argparse library to create an argument parser. We can directly give an image path from the command prompt:



```
import argparse

ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True, help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']
#Reading image with opencv
img = cv2.imread(img_path)
```

4.2 We read the CSV file with pandas

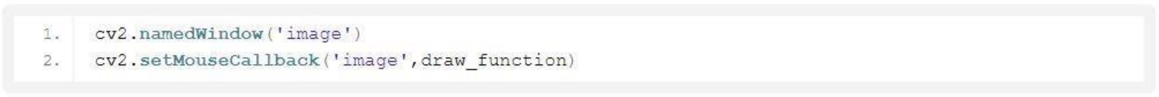
The pandas library is very useful when we need to perform various operations on data files like CSV. **pd.read_csv()** reads the CSV file and loads it into the pandas DataFrame. We have assigned each column with a name for easy accessing.



```
#Reading csv file with pandas and giving names to each column
index=["color","color_name","hex","R","G","B"]
csv = pd.read_csv('colors.csv', names=index, header=None)
```

4.3 Set a mouse call back event on a window

First, we created a window in which the input image will display. Then, we set a callback function which will be called when a mouse event happens.

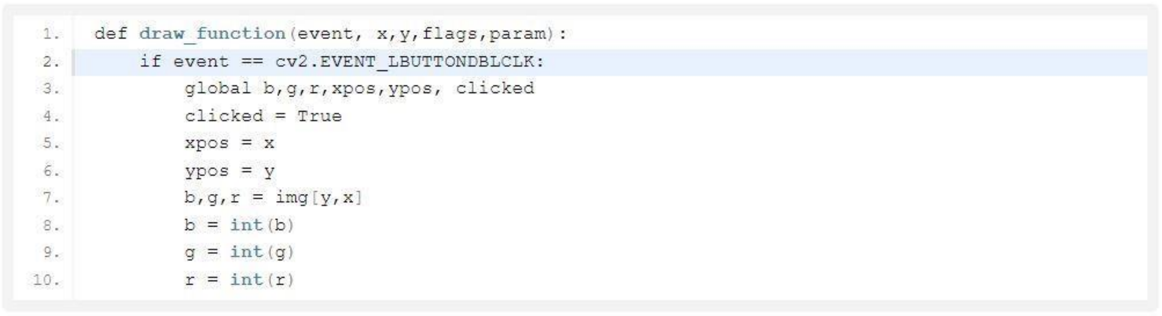


```
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_function)
```

With these lines, we named our window as ‘image’ and set a callback function which will call the **draw_function()** whenever a mouse event occurs.

4.4 Create the draw_function

It will calculate the rgb values of the pixel which we double click. The function parameters have the event name, (x,y) coordinates of the mouse position, etc. In the function, we check if the event is doubleclicked then we calculate and set the r,g,b values along with x,y positions of the mouse.

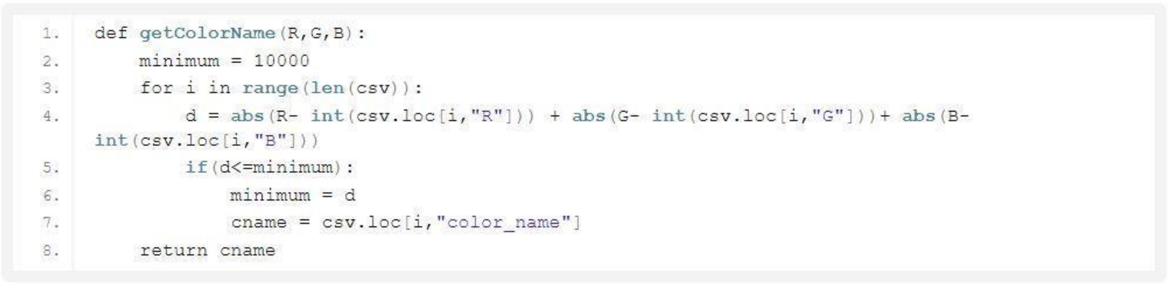


```
def draw_function(event, x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b,g,r,xpos,ypos, clicked
        clicked = True
        xpos = x
        ypos = y
        b,g,r = img[y,x]
        b = int(b)
        g = int(g)
        r = int(r)
```

4.5 Calculate distance to get color name

We have the r,g and b values. Now, we need another function which will return us the color name from RGB values. To get the color name, we calculate a distance(d) which tells us how close we are to color and choose the one having minimum distance. Our distance is calculated by this formula:

d = abs(Red – ithRedColor) + (Green – ithGreenColor) + (Blue – ithBlueColor)

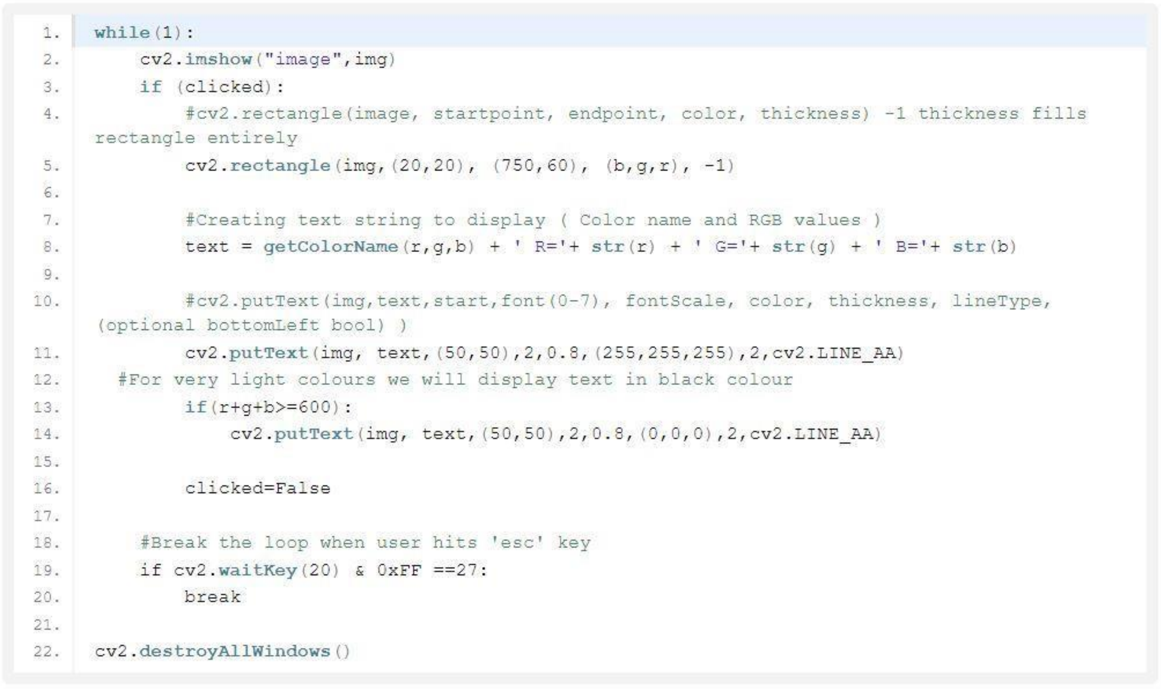


```python
def getColorName(R,G,B):
    minimum = 10000
    for i in range(len(csv)):
        d = abs(R- int(csv.loc[i,"R"])) + abs(G- int(csv.loc[i,"G"]))+ abs(B- int(csv.loc[i,"B"]))
        if(d<=minimum):
            minimum = d
            cname = csv.loc[i,"color_name"]
    return cname
```

4.6 Display image on the window

Whenever a double click event occurs, it will update the color name and RGB values on the window.

Using the **cv2.imshow()** function, we draw the image on the window. When the user double clicks the window, we draw a rectangle and get the color name to draw text on the window using **cv2.rectangle** and **cv2.putText()** functions.

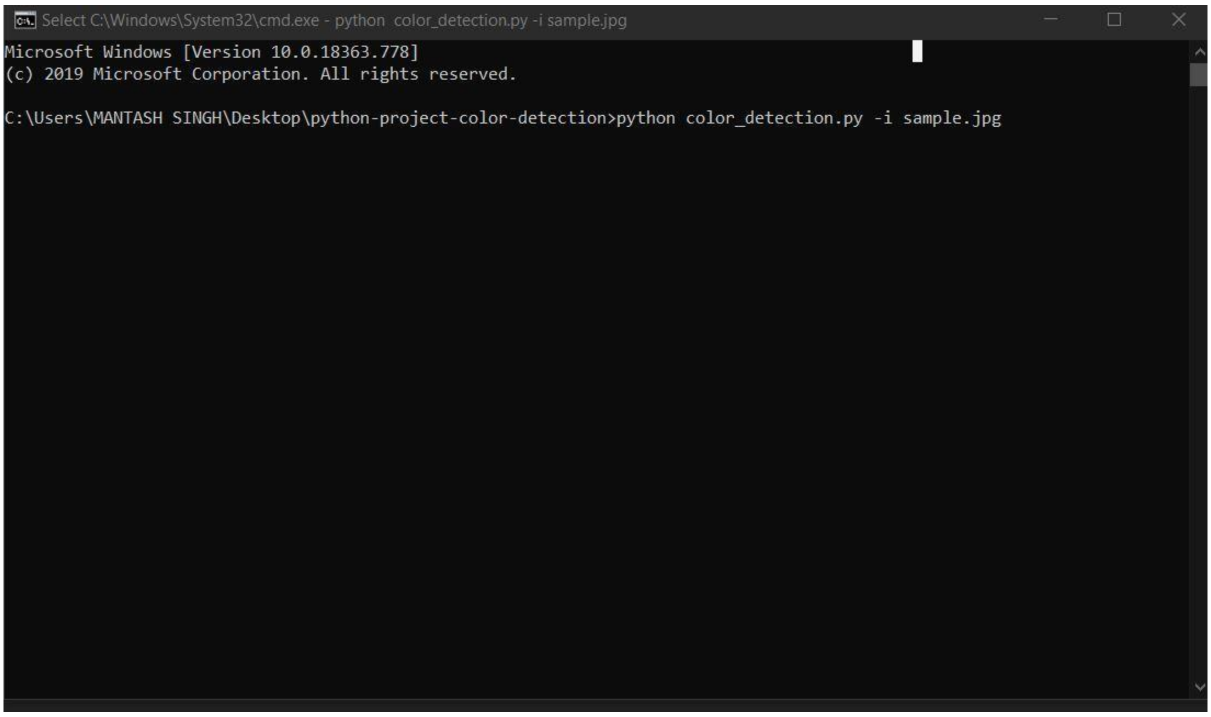


```
while(1):
    cv2.imshow("image",img)
    if (clicked):
        #cv2.rectangle(image, startpoint, endpoint, color, thickness) -1 thickness fills rectangle entirely
        cv2.rectangle(img,(20,20), (750,60), (b,g,r), -1)

        #Creating text string to display ( Color name and RGB values )
        text = getColorName(r,g,b) + ' R='+ str(r) + ' G='+ str(g) + ' B='+ str(b)

        #cv2.putText(img,text,start,font(0-7), fontScale, color, thickness, lineType, (optional bottomLeft bool) )
        cv2.putText(img, text,(50,50),2,0.8,(255,255,255),2,cv2.LINE_AA)
  #For very light colours we will display text in black colour
        if(r+g+b>=600):
            cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

        clicked=False

    #Break the loop when user hits 'esc' key
    if cv2.waitKey(20) & 0xFF ==27:
        break

cv2.destroyAllWindows()
```

CHAPTER – 5 OUTPUT

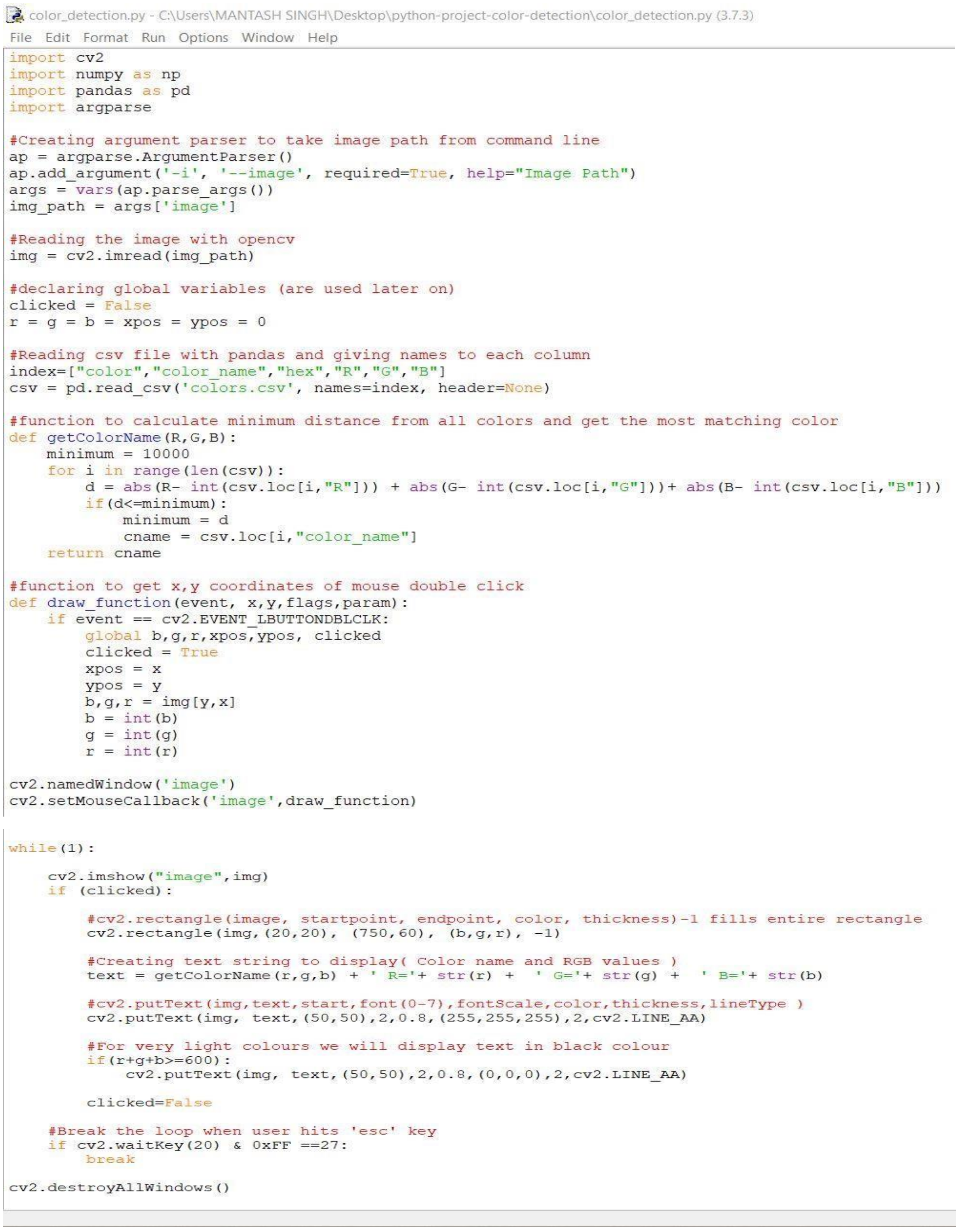
5.1 Run Python File

We can run the Python file from the command prompt. Make sure to give an image path using ‘-i’ argument. If the image is in another directory, then you need to give full path of the image:


Select C:\Windows\System32\cmd.exe - python color_detection.py -i sample.jpg

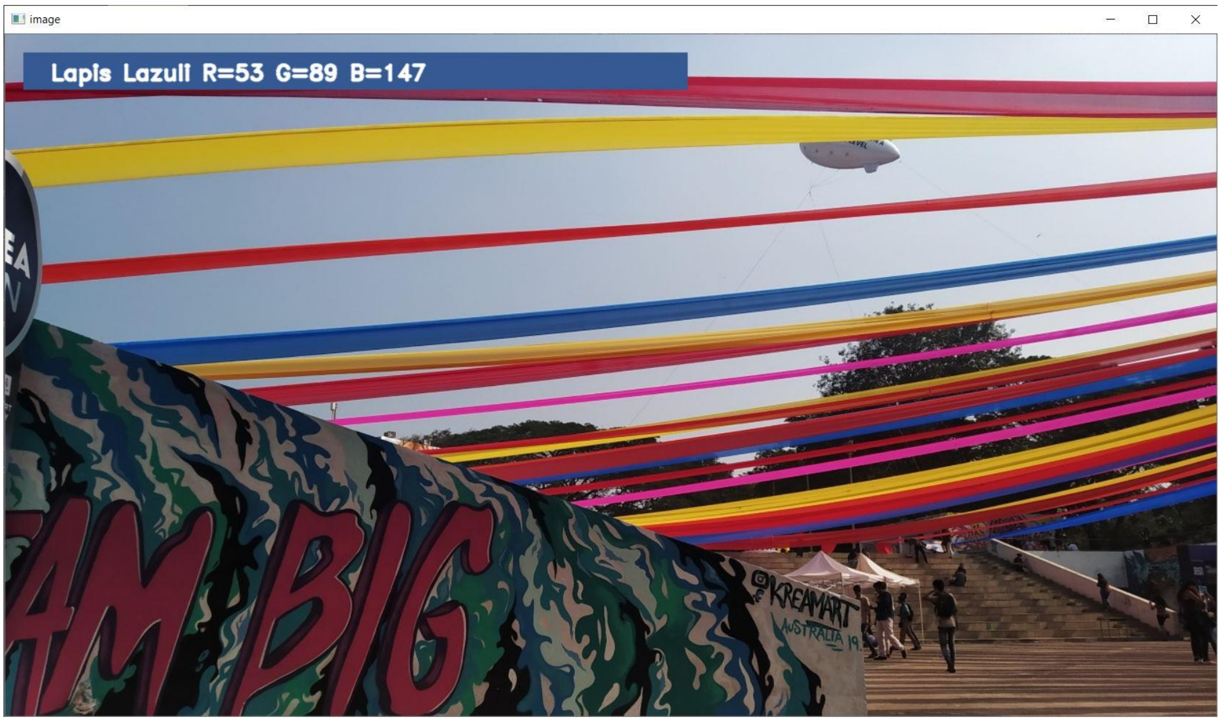
```
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\MANTASH SINGH\Desktop\python-project-color-detection>python color_detection.py -i sample.jpg
```

5.2 Screenshot of the program


color_detection.py - C:\Users\MANTASH SINGH\Desktop\python-project-color-detection\color_detection.py (3.7.3)
File Edit Format Run Options Window Help

```python
import cv2
import numpy as np
import pandas as pd
import argparse

#Creating argument parser to take image path from command line
ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True, help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']

#Reading the image with opencv
img = cv2.imread(img_path)

#declaring global variables (are used later on)
clicked = False
r = g = b = xpos = ypos = 0

#Reading csv file with pandas and giving names to each column
index=["color","color_name","hex","R","G","B"]
csv = pd.read_csv('colors.csv', names=index, header=None)

#function to calculate minimum distance from all colors and get the most matching color
def getColorName(R,G,B):
    minimum = 10000
    for i in range(len(csv)):
        d = abs(R- int(csv.loc[i,"R"])) + abs(G- int(csv.loc[i,"G"]))+ abs(B- int(csv.loc[i,"B"]))
        if(d<=minimum):
            minimum = d
            cname = csv.loc[i,"color_name"]
    return cname

#function to get x,y coordinates of mouse double click
def draw_function(event, x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b,g,r,xpos,ypos, clicked
        clicked = True
        xpos = x
        ypos = y
        b,g,r = img[y,x]
        b = int(b)
        g = int(g)
        r = int(r)

cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_function)

while(1):

    cv2.imshow("image",img)
    if (clicked):

        #cv2.rectangle(image, startpoint, endpoint, color, thickness)-1 fills entire rectangle
        cv2.rectangle(img,(20,20), (750,60), (b,g,r), -1)

        #Creating text string to display( Color name and RGB values )
        text = getColorName(r,g,b) + ' R='+ str(r) +  ' G='+ str(g) +  ' B='+ str(b)

        #cv2.putText(img,text,start,font(0-7),fontScale,color,thickness,lineType )
        cv2.putText(img, text,(50,50),2,0.8,(255,255,255),2,cv2.LINE_AA)

        #For very light colours we will display text in black colour
        if(r+g+b>=600):
            cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

        clicked=False

    #Break the loop when user hits 'esc' key
    if cv2.waitKey(20) & 0xFF ==27:
        break

cv2.destroyAllWindows()
```

5.3 Output



Lapis Lazuli R=53 G=89 B=147