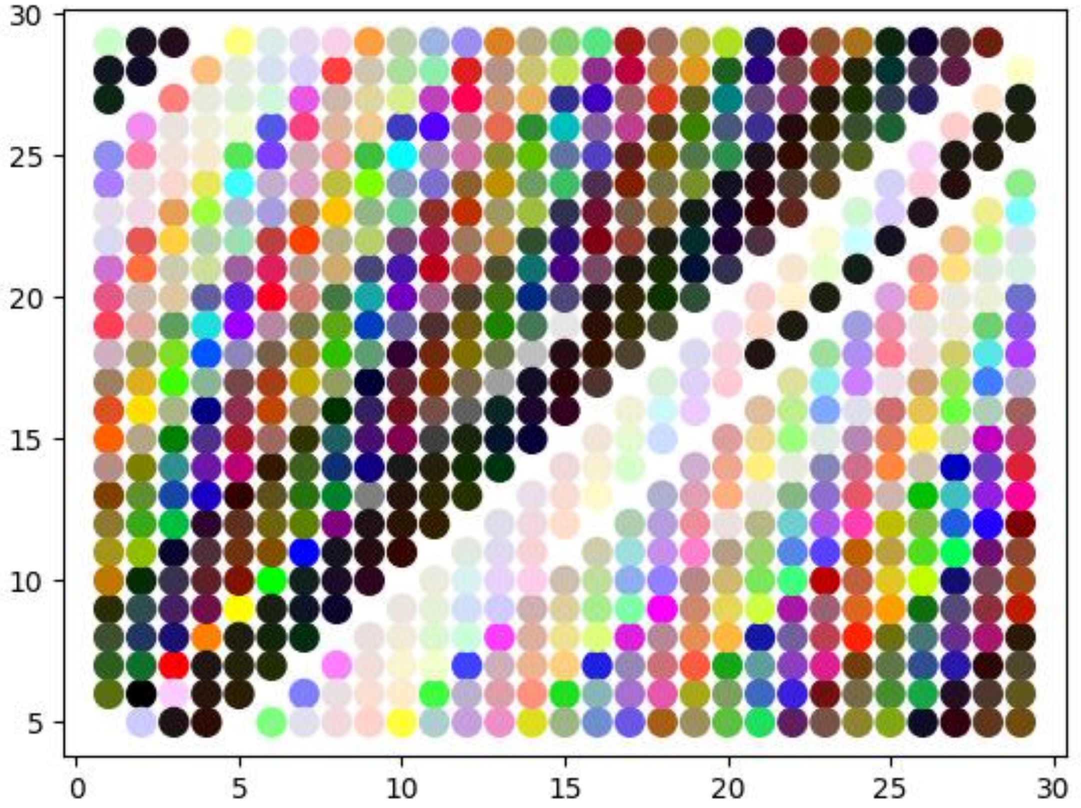
Generation of Color Code using Color Naming System

Prakhar Srivastava
Department of Electrical and Computer Engineering,
Stony Brook University, NY, USA
Emails: prakhar.sri4@gmail.com, prakhar.srivastava@stonybrook.edu



Acknowledgement

I would like to express my special thanks of gratitude to my professor ***Dr. Yanhong Annie Liu*** *who gave me the golden opportunity to do this wonderful project on the topic "Generation of Color Code using Color Naming System", which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful.*

Secondly I would also like to thank ***Dr. Arie E. Kaufman*** *for his research on "A New Color-Naming System for Graphics Languages", which helped me a lot in finalizing the project within the limited time frame***.**

Contents

# Project: gen_colorcode

# Objective:

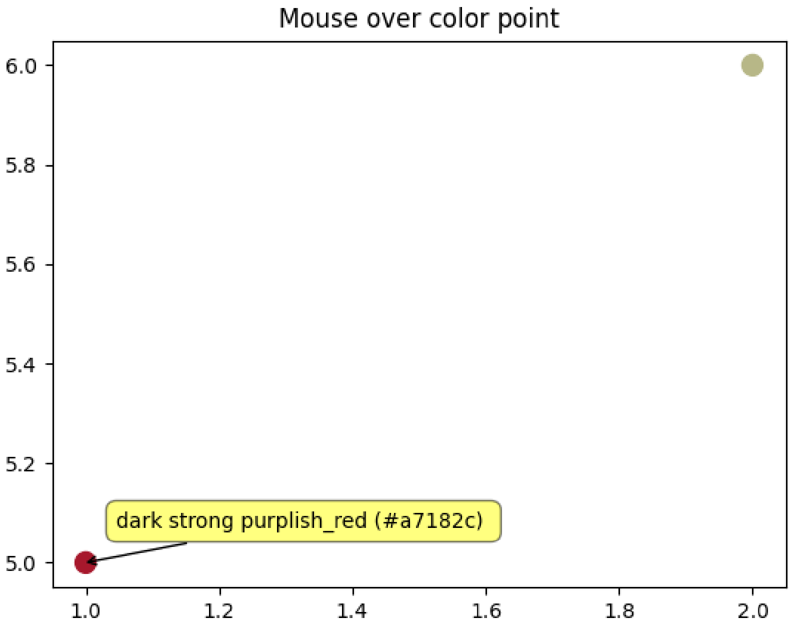
To reduce the redundancy in selection of color for website making, project, blogging, newsletter, online graphics etc. Here the program takes Input as a color name and give the output in color code. Also, we can see colors with the respective code in the plot generated after the execution of the program. The program is working perfectly and generates RGB code, HSL code and Hex code. See below: how to run the program.

For example:

```
Do wanna enter color names or see all colors. (?custom/all):   custom
Enter the color name:  dark strong purplish red
Do wanna enter more enter more color names. (?y/n): y
Enter the color name:  light bit yellow
Do wanna enter more enter more color names. (?y/n): n

dark strong purplish_red, rgb=> [167, 24, 44] , hsl=> [351.4286, 75, 37.5] , hex=> #a7182c
light bit yellow, rgb=> [183, 183, 135] , hsl=> [60.0, 25, 62.5] , hex=>  #b7b787
  Total number of different color codes generated:  2
  Total number of all color codes generated:  2
 Total time before graph display:  1.3265 seconds
 Execution time:  92.1879 seconds
```

Note: 1.) Total time = Time taken while entering names + code generated
2.) Execution time is high due to the graph. It will be shown after the color window is closed. Execution time = Time taken while entering color names + color window opened.



# State of arts:

The CNS (color naming system) is the nice method that uses dictionaries of color and follows the ISCC-NBS lexicon. This will help to reduce time and work load required for creating colors. It takes color name as input and returns the color code. It's like writing English. Program also runs in DistAlgo, check the most interesting things below.

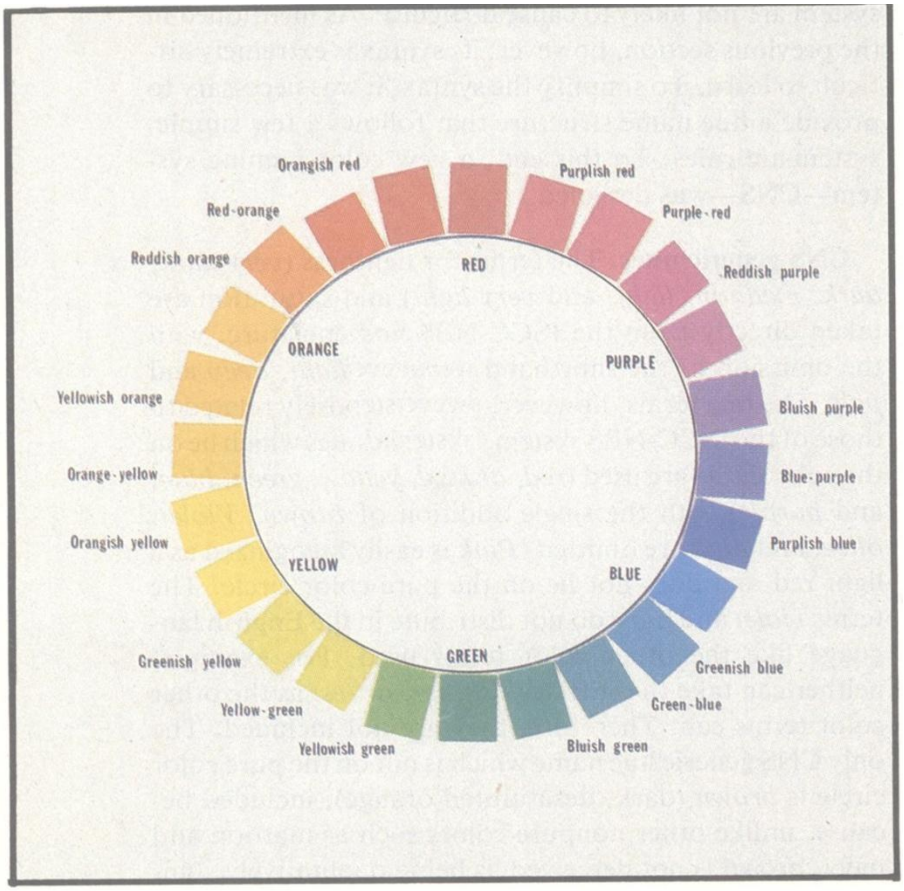
# How to find files:

1.) Lark folder have lark module (library).
2.) Color_code.py file contain python code of the project.
3.) Color_code.da file contain DistAlgo code of the project.
4.) Format.jpg holds the picture that demonstrates a tree of color.
5.) The third file Output file, it has output of the entire multiple color names with code. However, program prints and display graph in the output after running it in jupyter/python/pycharm.
6.) Report.pdf, this file contains all the informations.
7.) Readme.md file have running and some other stuff.

# Main libraries required for the project:

1.) Lark module, for parsing name using context-free grammar. Lark is used instead of tpg because Lark is user friendly, efficient and handle ambiguity gracefully. [For more details, visit: https://github.com/lark-parser/lark].
2.) Enum, for specifying the color value.
3.) Time, used to calculate the execution time.
4.) Itertools, used to make multiple names for testing. [For creating input names].
5.) Matplotlib, used to create scattered colors in the graph.
6.) Mplcursors, used to display hovering in the graph with color name and code.

# How to create basic colors [1]:



1.) Took Red [255, 0, 0], Green [0, 255, 0] and Blue [0, 0, 255].
2.) Calculate the middle of all the above colors.
3.) This will generate different patterns of colors. [1]

# Equations and Formulae:

1.) **Convert the Hue(H), Saturation(S) and Lightness(L) into RGB system: [9-12]**

RGB = Basic/Hue color code

Lightness calculation:

Divide $R/255$, $G/255$, $B/255$. (RGB value range [0,1])

RGB color wheel: $L = [0, 1]$

$L = (1 / 2) \times (\text{Max}(RGB) + \text{Min}(RGB)) * 100$ (Multiply with 100 to get value of lightness in percentage)

Hue Calculation:

RGB color wheel: $H = [0^\circ, 360^\circ]$

(A) If $R \geq G \geq B \mid H = 60^\circ \times [(G-B)/(R-B)]$

(B) If $G > R \geq B \mid H = 60^\circ \times [2 - (R-B)/(G-B)]$

(C) If $G \geq B > R \mid H = 60^\circ \times [2 + (B-R)/(G-R)]$

(D) If $B > G > R \mid H = 60^\circ \times [4 - (G-R)/(B-R)]$

(E) If $B > R \geq G \mid H = 60^\circ \times [4 + (R-G)/(B-G)]$

(F) If $R \geq B > G \mid H = 60^\circ \times [6 - (B-G)/(R-G)]$

Saturation Calculation:

RGB color wheel: $S = [0, 1]$

(A.) If $L < 1 \mid S = (\text{Max}(RGB) - \text{Min}(RGB)) / (1 - |2L - 1|) * 100$ (Multiply with 100 to get value of saturation in percentage)

(B.) If $L = 1 \mid S = 0$

2.) **Convert RGB system to Hex system [10][15]:**

The RGB color is a combination of **R**ed, **G**reen and **B**lue colors. The red, green and blue use 8 bits each, which have integer values from 0 to 255. Hex color code is a 6 digits hexadecimal (base 16) number.

$RRGGBB_{16}$

The 2 left digits represent the red color.

The 2 middle digits represent the green color.

The 2 right digits represent the blue color.

RGB to hex conversion

1. Convert the red, green and blue color values from decimal to hex.
2. Concatenate the 3 hex values of the red, green and blue together: RRGGBB.

# Methodology of implementation:

1.) Import all required modules.

2.) Create list of color name and write their grammars. Grammar concept taken from "A New Color-Naming System for Graphics Languages" research paper [1] and "Color Naming System" from Wikipedia [2]. The paper and Wikipedia, they mentioned the methods of color model.

3.) Defined functions of hue, lightness, saturation, general-hue, half-hue (e.g. red_orange), quarter-hue (e.g. reddish_orange) and ish-form (e.g. bluish). These functions initialize the basic color name with their respective color code. The color code taken from w3schools website [6] and The ISCC-NBS Color System [7].

4.) Defined colorTransformer class that transform the color name and generate color code.

5.) After all the above function, our program read and prints the entered color value.

6.) Now, a function called hue-call is defined to calculate the hue, this is used to determine the HSL code. Here, mathematical approach of color conversion is used instead of built-in module (colorsys). Hue calculation formula taken from Donat Bali Papp, "Let's Colormath" website [9], research paper "Experimental investigation of HSL color model in error diffusion" [11].

7.) Defined a function of HSL to RGB code conversion, in this function parameters are saturation, lightness and hue, which we get from hue-call function. The hsl_to_rgb(h,s,l) function converts the HSL code into RGB code. Here, mathematical approach of color conversion is used instead of built-in module (colorsys). Color conversion equation taken from "Adaptive image enhancement method for correcting low-illumination images"[12], RapidTables, "HSL to RGB color conversion" [13] and Wikipedia, "HSL and HSV" [14].

8.) Defined a function of RGB to hex code conversion, in this function parameter is rgb code. The function converts RGB code into HEX code. This is same as above step, mathematical approach of color conversion is used instead of built-in module (colorsys). Color conversion equation taken from RapidTables, " How to convert RGB to hex color" [10].

9.) In the end, a main is defined and 'for' loop is used to generate all colors name using above basic color. After that scattering graph and hovering are defined. Also, prints the time and number of color code generated.

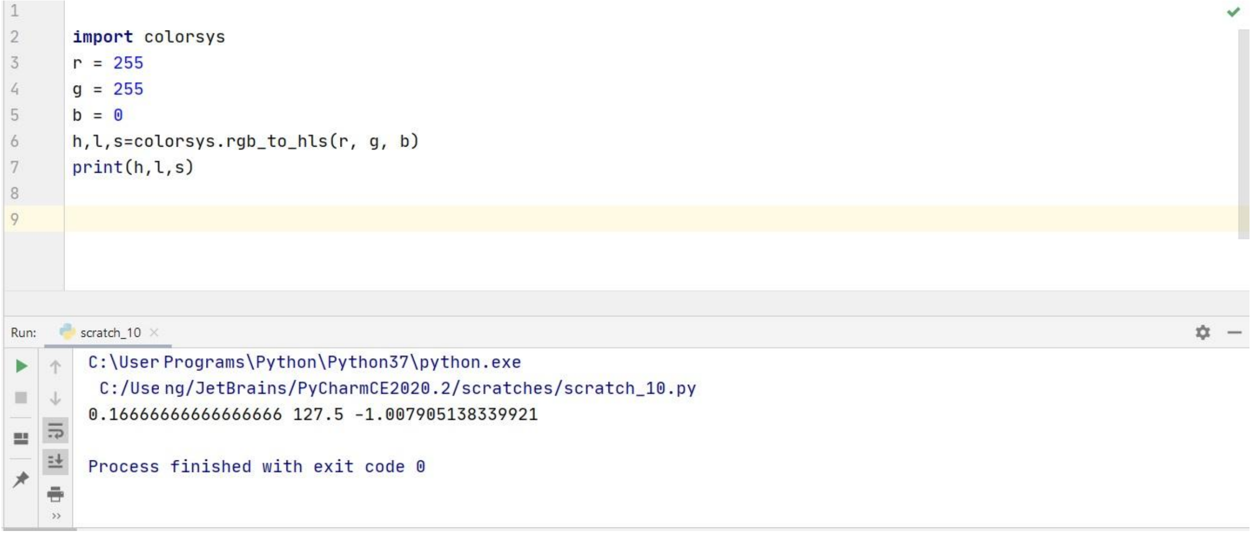
# Problems faced in Built-in Library:

1.) Colorsys library (Module) not used because it generate wrong values for RGB code.
2.) Conversion in Colorsys, also give wrong result.

For example:

RGB = [255, 255, 0] => HSL = [0.1662, 127, -1.0079].

Generating negative value which cannot used because it should be either [0, 1] or [0,255]. Correct value: RGB = [255, 255, 0] => HSL = [60, 100%, 50%].



```
import colorsys
r = 255
g = 255
b = 0
h,l,s=colorsys.rgb_to_hls(r, g, b)
print(h,l,s)
```

```
C:\User Programs\Python\Python37\python.exe
 C:/Use ng/JetBrains/PyCharmCE2020.2/scratches/scratch_10.py
0.16666666666666666 127.5 -1.007905138339921

Process finished with exit code 0
```

# How to Enter color name:

First:

Lightness –

extreme light = [90%]

very light = [75%]

light = [62.5%]

medium = [50%]

dark = [37.5%]

very dark = [25%]

extreme dark = [10%]

Second:

Saturation –

vivid = [100%]

strong = [75%]

moderate = [50%]

bit = [25%]

grayish = [0]

Third (Basic color names):

General Color name –

red

orange

yellow

green

blue

purple

Half hue –

red_orange

orange_yellow

yellow_green

green_blue

blue_purple

purple_red

Quarter hur –

orangish_red

reddish_orange

yellowish_orange

orangish_yellow

greenish_yellow

yellowish_green

bluish_green

greenish_green

purplish_blue

bluish_purple

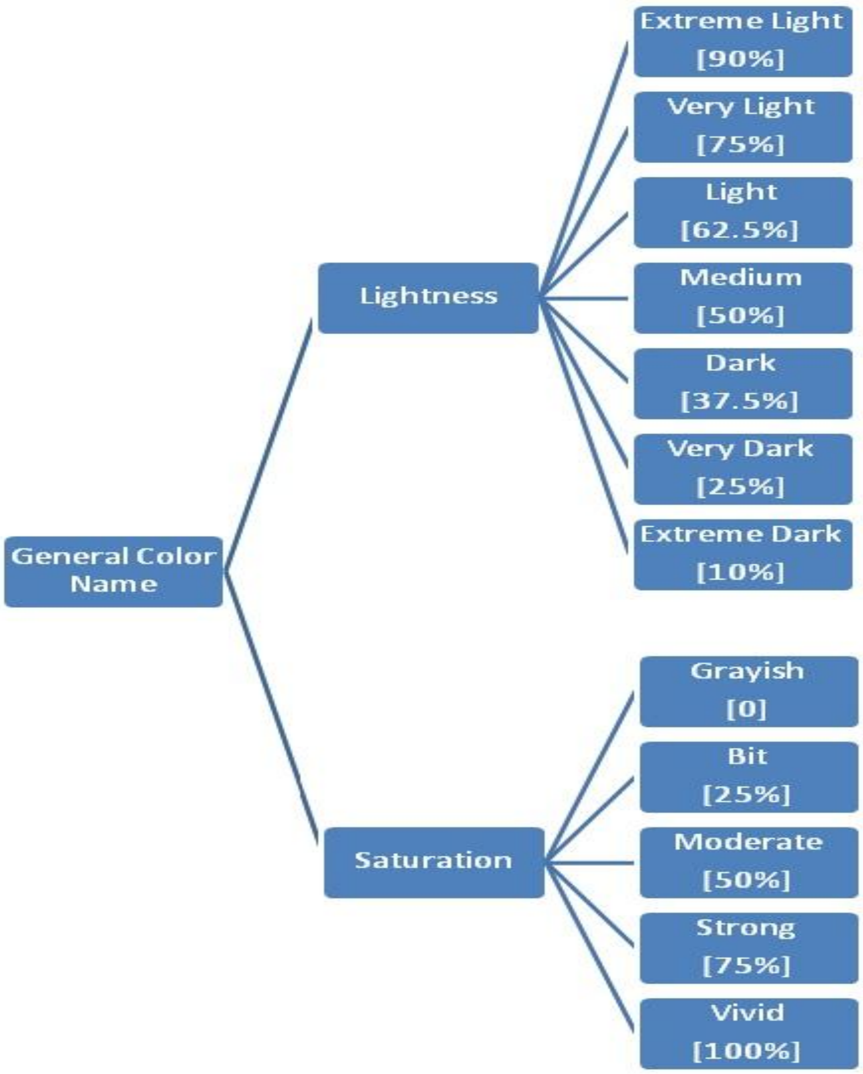
reddish_purple

purplish_red

1.) All names should be in small letters.
2.) First enter the lightness word.
3.) After that enter the saturation condition word, which is mentioned above.
4.) At last, write basic color name from three lists of color pattern:
    a.) General color name.
    b.) Half hue color name.
    c.) Quarter hue color name.
5.) Press ENTER.

# Format of Color:

- In the paper, “A New Color-Naming System for Graphics Languages” there are five parts of lightness and four parts of saturation. To increase more variants of colors, we added two more lightness name such as ‘extreme light’ and ‘extreme dark’ as well as in saturation ‘bit’ is added.
- To reduce the redundancy felt by the user while selecting colors. We fixed lightness and saturation percentage as shown below.



# Some color names:

1.) Pink = ‘medium vivid purple’ and ‘light vivid reddish_purple’.
2.) Brown = ‘dark strong red orange’ and ‘dark strong orangish red’.
3.) Medium gray = gray

Note: These above are some names that look similar.

# How to run the program:

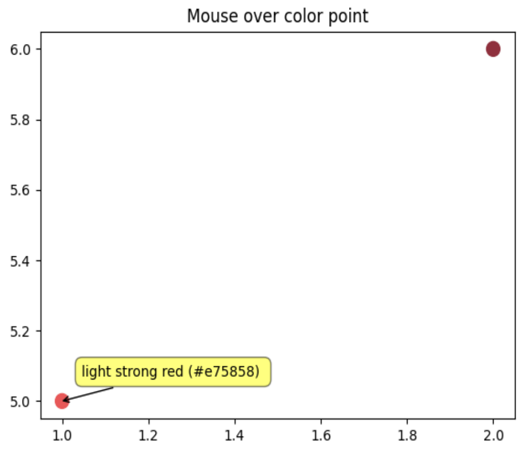
1.) Use jupyter/pycharm/python to run the program.
2.) Install all module mentioned above especially Lark module.
3.) Currently, program is creating multiple names automatically but we can customize it. These names are used as input color names.
4.) Run the program in the IDE.
5.) Then it will ask, all or custom:
    a.) If entered all, each and every possible colors presents in the grammar will print.
    b.) If entered custom, then it will ask to enter color names.
    c.) After that if ask for more entry of color name.
6.) When the search result complete, color code generated after parsing and analyzing the color names. Color names are present in the output along with RGB code, HSL code and Hex code. Also, after few second colors will display in the graph with their names and code (When mouse hover on then).
7.) It will also display the total number of color code generated and execution time.

# Results:

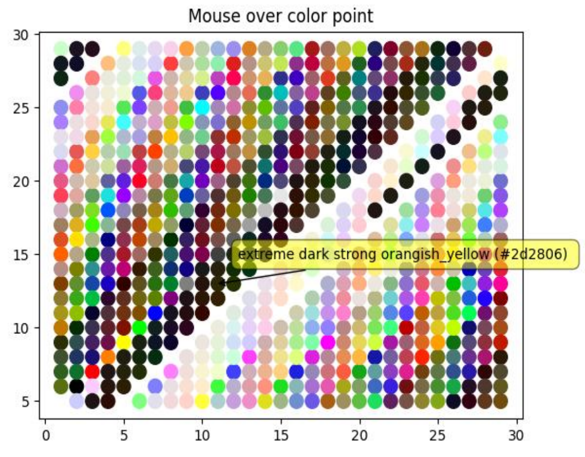
```
Do wanna enter color names or see all colors. (?custom/all):   custom
Enter the color name:  light strong red
Do wanna enter more enter more color names. (?y/n): yes
Enter the color name:  dark moderate purplish red
Do wanna enter more enter more color names. (?y/n): no

light strong red, rgb=> [231, 88, 88] , hsl=> [0.0, 75, 62.5] , hex=>  #e75858
dark moderate purplish_red, rgb=> [143, 48, 61] , hsl=> [351.4286, 50, 37.5] , hex=>
#8f303d
  Total number of different color codes generated:  2
  Total number of all color codes generated:  2
 Total time before graph display:  1.8587  seconds
 Execution time:  34.517  seconds
```

Note: 1.) Total time = Time taken while entering names + code generated 2.) Execution time is high due to the graph. It will be shown after the color window is closed. Execution time = Time taken while entering color names + color window opened.



(a.) Two Enter Color

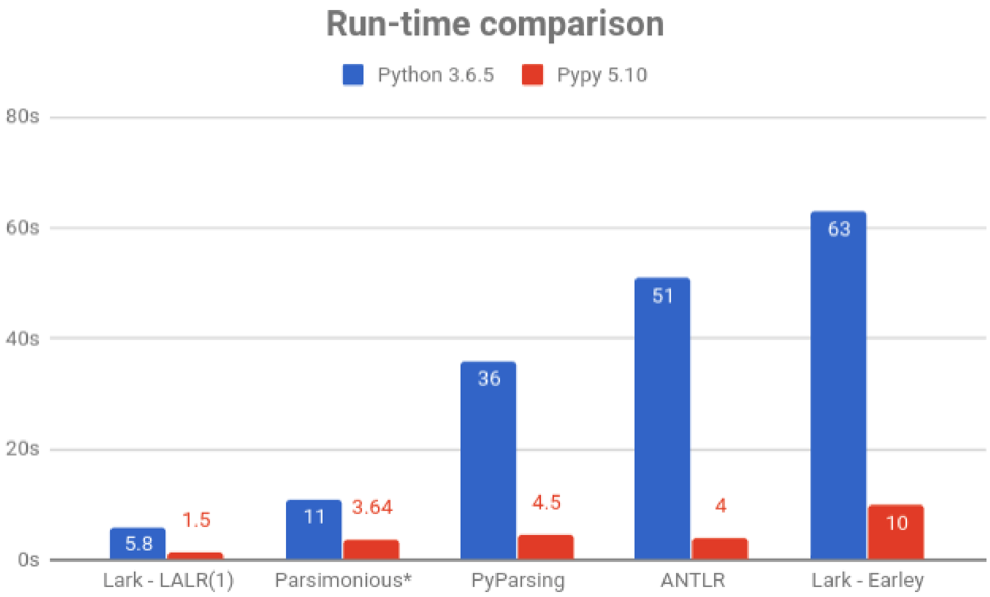


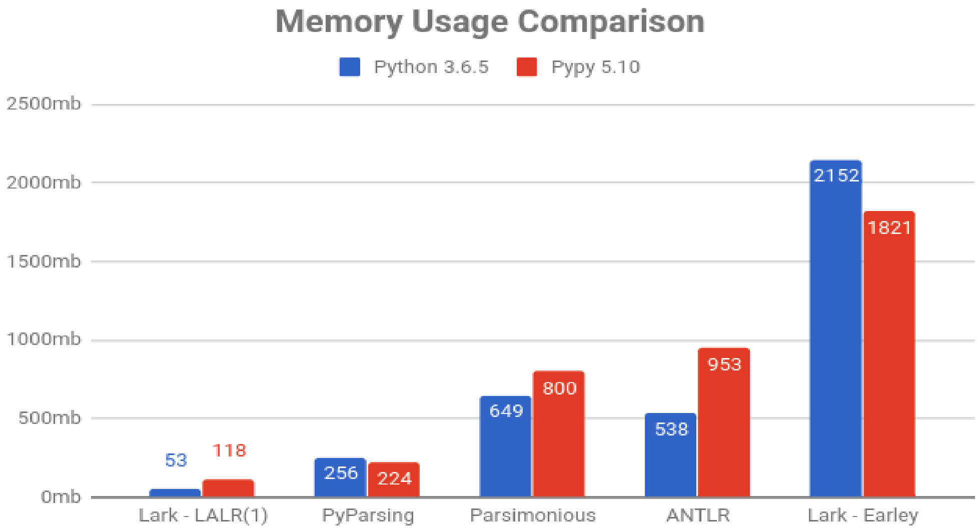
(b.) All colors

# Why Lark:

Lark can parse all context-free languages. It is very friendly and no matter how complicated or ambiguous the grammar, it work so efficiently. Build an annotated parse-tree automatically, no construction code required. Run on every Python interpreter. Most importantly, Lark will save time and prevent you from getting parsing headaches [16].

Performance comparison: Lark is the fastest and lightest (lower is better)





# Most Interesting thing:

1.) This program also runs in DistAlgo.
2.) The file color_code.da has the program and in the terminal writes "python -m da color_code.da" then presses enter.
3.) Program will generate the color code and graph same as in python.

# Conclusion:

Get better results from the HSL system and are used to determine the RGB and HEX systems. Even greater accuracy is obtained with the English-based color-code generator. It not only uses the measurement as an HSL system, but it also supports feedback based on life experience with English color terms. It is believed that the ease of learning by naive users is a very important feature for any system. Therefore, this project was a valid approach of comparing other systems. The main advantage of the English-language-based color-code generator is that users do not need to refer to printed tables of color-magnitude to decide on appropriate first attempts at the selection of colors to be displayed. However, users will experience less trial and error with the method than with one of the quantitative systems in a continuous effort to converge to an acceptable color.

# References:

1.) T. Berk, A. Kaufman and L. Brownston, "A New Color-Namiing System for Graphics Languages" in IEEE Computer Graphics and Applications, vol. 2, no. 03, pp. 37-44, 1982.
2.) "Color Naming System", (2021). Wikipedia, Available at https://en.wikipedia.org/wiki/Color_Naming_System. (Accessed: 20 March 2021).
3.) Tab Atkins-Bittner, “The CNS Color Naming System”, https://www.xanthir.com/blog/b4JF0, June 4 2012. (Accessed: 20 March 2021).
4.) ColorDesigner, "Color Mixer", https://colordesigner.io/color-mixer. (Accessed: 12 April 2021).
5.) Benjamin D. Gray (May 2002), "New CSS3 Color Names", (Accessed: 12 April 2021).

6.) w3schools, "HTML Color Picker", https://www.w3schools.com/colors/colors_picker.asp. (Accessed: 21 April 2021).

7.) Paul Centore, "The ISCC-NBS Colour System" (2016), https://www.munsellcolourscienceforpainters.com/ISCCNBS/ISCCNBSSystem.html. (Accessed: 21 April 2021).

8.) w3schools, "Colors - ISCC/NBS", https://www.w3schools.com/colors/colors_nbs.asp. (Accessed: 22 April 2021).

9.) Donat Bali Papp, "Let's Colormath" (Sep 20, 2017), https://donatbalipapp.medium.com/colours-maths-90346fb5abda. (Accessed: 22 April 2021).

10.) RapidTables, "How to convert RGB to hex color", https://www.rapidtables.com/convert/color/how-rgb-to-hex.html. (Accessed: 23 April 2021).

11.) T. Lin, B. Liao, S. Hsu and J. Wang, "Experimental investigation of HSL color model in error diffusion," *2015 8th International Conference on Ubi-Media Computing (UMEDIA)*, 2015, pp. 268-272, doi: 10.1109/UMEDIA.2015.7297467.

12.) Wencheng Wang, Zhenxue Chen, Xiaohui Yuan, Xiaojin Wu, "Adaptive image enhancement method for correcting low-illumination images", Information Sciences, Volume 496, 2019, Pages 25-41, ISSN 0020-0255.

13.) "HSL and HSV", (2021). Wikipedia, Available at https://en.wikipedia.org/wiki/HSL_and_HSV. (Accessed: 23 April 2021).

14.) RapidTables, "HSL to RGB color conversion", https://www.rapidtables.com/convert/color/hsl-to-rgb.html. (Accessed: 23 April 2021).

15.) Developintelligence, "RGB to Hex: Understanding the Major Web Color Codes", https://www.developintelligence.com/blog/2017/02/rgb-to-hex-understanding-the-major-web-color-codes/#:~:text=Take%20the%20first%20number%2C%20220,code%20is%2012%2C%20or%20C. (Accessed: 24 April 2021).

16.) Erez Shinan, "Lark-a parsing toolkit for Python", GitHub repository, https://github.com/lark-parser/lark/blob/master/README.md. (Accessed: 24 April 2021)

17.) RapidTables, "RGB to HSL color conversion", https://www.rapidtables.com/convert/color/rgb-to-hsl.html. (Accessed: 24 April 2021).