Day - 18

**Turtle & the GUI**

Turtle graphics, Tuples and Importing modules

**18.1 Turtle Graphics with its Documentation**

**from** turtle **import** Screen, Turtle

**import** turtle

turtle\_tim = **Turtle**()

turtle\_tim**.shape**("turtle")

turtle\_tim**.color**("red", "green")

turtle\_tim**.forward**(100)

turtle\_tim**.right**(90)

scRen = **Screen**()

scRen**.exitonclick**()

#*python turtle\_intro.py*

* TK color specification string (TKINTER): Python - Tkinter Colors. Tkinter represents colors with strings. There are two general ways to specify colors in Tkinter −
* You can use a string specifying the proportion of red, green and blue in hexadecimal digits. For example, "#fff" is white, "#000000" is black, "#000fff000" is pure green, and "#00ffff" is pure cyan (green plus blue).
* You can also use any locally defined standard color name. The colors "white", "black", "red", "green", "blue", "cyan", "yellow", and "magenta" will always be available.
* Exercise 18.1: Turtle Challenge 1 - Draw a Square.

**from** turtle **import** Screen, Turtle

timmy = **Turtle**()

timmy**.shape**("turtle")

timmy**.color**("red", "green")

**for** i **in** **range**(0, 4):

    timmy**.forward**(100)

    timmy**.right**(90)

scRen = **Screen**()

scRen**.exitonclick**()

#*python turtle\_square.py*

* Exercise 18.2: Turtle Challenge 1 - Draw a Dashed Line.

|  |  |
| --- | --- |
| Practice version | Instructors solution |
| **from** turtle **import** Screen, Turtle, forward  timmy = **Turtle**()  timmy**.shape**("turtle")  timmy**.color**("red", "green")  **for** i **in** **range**(0, 50):  **if** i%2 **==** 0:          timmy**.pendown**()          timmy**.forward**(10)  **else**:          timmy**.penup**()          timmy**.forward**(10)  scRen = **Screen**()  scRen**.exitonclick**()  #*python turtle\_square.py* | **from** turtle **import** Screen, Turtle, forward  timmy = **Turtle**()  timmy**.shape**("turtle")  timmy**.color**("red", "green")  **for** i **in** **range**(0, 50):          timmy**.pendown**()          timmy**.forward**(5)          timmy**.penup**()          timmy**.forward**(5)  scRen = **Screen**()  scRen**.exitonclick**()  #*python turtle\_dash\_instructor.py* |

**18.2 Importing Modules**

* To install packages, we use ***pip*** or ***pypi.org*** to download the file.
* Importing everything: use "**\***". For example,

**from** turtle **import** \*

* Note: Using this way is kind of destructure of the code. Some methods then could not be separable from the imported modules. The good idea is using

**import** turtle

* then accessing the methods can be tracked by the object name and the '.' operator. Eg:

**import** turtle

timmy = turtle**.Turtle**()

timmy**.shape**("turtle")

* Avoid using asterisk \*,as to its origin, it's very confusing
* If you are going to use the module multiple times:

**from** turtle **import** Turtle()

tom = Turtle()

* Clear expressive: If it's for just once, specify the module name, which contains the class when making a new object. To make things *clear*, *expressive*

**import** turtle

tim = turtle.Turtle()

**18.3 Alias for imported modules/classes**

* Aliasing Modules: In following, turtle has an alia (other name) '**t**'. It is good for very long module name. Use the "**as**" keyword

**from** turtle **as** t

terry = t.Turtle()

* Exercise 18.3: Drawing Different Shapes.

|  |  |
| --- | --- |
| Practice version | Instructors solution |
| **from** turtle **import** Screen, Turtle, forward  timmy = **Turtle**()  timmy**.shape**("turtle")  timmy**.color**("red", "green")  **for** i **in** **range**(1, 11):  **if** i%2 **==** 0:          timmy**.pencolor**("red")  **elif** i%3 **==** 0:          timmy**.pencolor**("green")  **for** j **in** **range**(0,i):          timmy**.forward**(50)          timmy**.right**(360/i)  scRen = **Screen**()  scRen**.exitonclick**()  #*python turtle\_shapes.py* | **import** random  **from** turtle **import** Screen, Turtle, forward  **import** turtle  timmy = **Turtle**()  timmy**.shape**("turtle")  timmy**.color**("red", "green")  coLR = ["red", "green", "blue", "black", "violet"]  **def** **shape**(sides):      anGle = 360/sides  **for** i **in** **range**(0, sides):          timmy**.forward**(50)          timmy**.right**(anGle)  **for** shape\_side **in** **range**(1, 11):      timmy**.pencolor**(random**.choice**(coLR))  **shape**(shape\_side)  scRen = **Screen**()  scRen**.exitonclick**()  #*python turtle\_shapes\_soln.py* |

* Exercise 18.4: Generate a Random Walk.mp4

|  |  |
| --- | --- |
| Practice version | Instructors solution |
| **import** random  **from** turtle **import** Screen, Turtle  **from** tk\_color\_list **import** COLORS  timmy = **Turtle**()  **for** i **in** **range**(0, 200):      path = random**.randint**(30, 60)      angle = random**.choice**([0, 45, 90, 135, 180, 225, 270, 315, 360])      timmy**.forward**(path)      timmy**.right**(random**.choice**([angle, -angle]))      timmy**.width**(5)      timmy**.speed**(9)      timmy**.color**(random**.choice**(COLORS))  scRen = **Screen**()  scRen**.exitonclick**()  #*python turtle\_ran\_wlk.py* | **import** random  **from** turtle **import** Screen, Turtle  #*from tk\_color\_list import COLORS*  colours = ["CornflowerBlue", "DarkOrchid", "IndianRed", "DeepSkyBlue", "LightSeaGreen", "wheat", "SlateGray", "SeaGreen"]  timmy = **Turtle**()  directions = [0, 90, 180, 270]  **for** \_ **in** **range**(200):      timmy**.forward**(30)      timmy**.setheading**(random**.choice**(directions))      timmy**.width**(15)      timmy**.speed**("fastest")      timmy**.color**(random**.choice**(colours))  scRen = **Screen**()  scRen**.exitonclick**()  #*python turtle\_ran\_wlk.py* |

**18.4 Python Tuples and Generate Random RGB Colours**

* Python Collections (Arrays): There are four collection data types in the Python programming language:

1. **List** is a collection which is *ordered* and *changeable*. *Allows* *duplicate* members.
2. **Tuple** is a collection which is *ordered* and *unchangeable*. *Allows* *duplicate* members.
3. **Set** is a collection which is *unordered*, *unchangeable*\*, and *unindexed*. *No* *duplicate* members.
4. **Dictionary** is a collection which is *ordered*\*\* and *changeable*. *No* *duplicate* members.

* Tuple: *Tuples* and *List* are similar but **Lists** are changeable but **Tuples** are unchangeable. Hence "tuples" are *immutable*. Touples are denoted as (a, b, c, .. ).
* ***Tuples*** are used to store *multiple* *items* in a *single* *variable*.
* ***Tuple*** is one of 4 built-in data types in Python used to store collections of data, the other 3 are ***List***, ***Set***, and ***Dictionary***, all with different qualities and usage.
* A ***tuple*** is a collection which is ORDERED and UNCHANGEABLE.
* ***Tuples*** are written with *round* *brackets* "**()"**.

**My\_tuple = (1, 2, 3)**

* Accessing tuple items: ***Tuple*** ***items*** are ***indexed***, the first item has index ***[0]***, the second item has index ***[1]*** etc.

**My\_tuple[1]**

* One member tuple: (12,)
* Converting tuples into lists: **list(tuple\_name)**
* ***colormode():*** turtle.colormode()
* Parameter: cmode is one of the values ***1.0*** or ***255***
* Return the colormode or set it to 1.0 or 255. Subsequently r, g, b values of color triples have to be in the range 0..cmode.

screen.colormode(1)

or

screen.colormode(255)

* Exercise 18.5: Generate a Random color with ***Tuple***.

|  |  |
| --- | --- |
| Practice version | Instructors solution |
| Here colormode is (0, 1)  **import** random  **from** turtle **import** Screen, Turtle  **def** **tUp**(rE, gR, bL):  **return** (rE, gR, bL)  timmy = **Turtle**()  directions = [0, 90, 180, 270]  **for** \_ **in** **range**(200):      r = random**.random**()      g = random**.random**()      b = random**.random**()  **print**(f"R={r} G={g}, B={b}")      timmy**.forward**(30)      timmy**.setheading**(random**.choice**(directions))      timmy**.width**(15)      timmy**.speed**("fastest")      timmy**.pencolor**(**tUp**(r, g, b))  scRen = **Screen**()  scRen**.exitonclick**()  #*python turtle\_ran\_wlk.py* | **import** random  **from** turtle **import** Screen, Turtle  timmy = **Turtle**()  scRen = **Screen**()  scRen**.colormode**(255)  **def** **rand\_color**():      r = random**.randint**(0, 255)      g = random**.randint**(0, 255)      b = random**.randint**(0, 255)  **return** (r, g, b)  directions = [0, 90, 180, 270]  **for** \_ **in** **range**(200):      timmy**.forward**(30)      timmy**.setheading**(random**.choice**(directions))      timmy**.width**(15)      timmy**.speed**("fastest")      timmy**.pencolor**(**rand\_color**())  scRen**.exitonclick**()  #*python turtle\_ran\_wlk.py* |

* Exercise 18.6: Generate a SPIROGRAPH.

|  |  |
| --- | --- |
| Practice version | Instructors solution |
| **import** random  **from** turtle **import** Screen, Turtle  timmy = **Turtle**()  scRen = **Screen**()  scRen**.colormode**(255)  **def** **rand\_color**():      r = random**.randint**(0, 255)      g = random**.randint**(0, 255)      b = random**.randint**(0, 255)  **return** (r, g, b)  **for** i **in** **range**(33):      timmy**.speed**("fastest")      timmy**.circle**(40)      timmy**.right**(10)      timmy**.width**(3)      timmy**.pencolor**(**rand\_color**())  scRen**.exitonclick**()  #*python turtle\_spirograph.py* | **import** random  **from** turtle **import** Screen, Turtle  timmy = **Turtle**()  scRen = **Screen**()  scRen**.colormode**(255)  **def** **rand\_color**():      r = random**.randint**(0, 255)      g = random**.randint**(0, 255)      b = random**.randint**(0, 255)  **return** (r, g, b)  timmy**.speed**("fastest")  **def** **draw\_spirograph**(size\_of\_gap):  **for** i **in** **range**(**int**(360/size\_of\_gap)+1):          timmy**.circle**(100)          timmy**.setheading**(timmy**.heading**() + size\_of\_gap)          timmy**.pencolor**(**rand\_color**())  **draw\_spirograph**(30)  scRen**.exitonclick**()  #*python turtle\_spirograph.py* |

**18.5 COLORGRAM – project and PILLOW**

**colorgram.py** is a Python library that lets you *extract colors from images*. Compared to other libraries, the colorgram algorithm’s results are more intense.

* ***colorgram.py*** is a port of [colorgram.js](https://github.com/darosh/colorgram-js), a JavaScript library written by GitHub user [@darosh](https://github.com/darosh).
* Time-wise, an extraction of a 512x512 image takes about 0.66s (another popular color extraction library, Color Thief, takes about 1.05s).
* Installation: You can install colorgram.py with pip, as following:

pip install colorgram.py

* How to use: Using colorgram.py is simple. Mainly there’s only one function you’ll need to use - colorgram.extract. Example:

**import** colorgram

#*Extract 6 colors from an image.*

colors = colorgram**.extract**('sweet\_pic.jpg', 6)

#*colorgram.extract returns Color objects, which let you access*

#*RGB, HSL, and what proportion of the image was that color.*

first\_color = colors[0]

rgb = first\_color**.**rgb #*e.g. (255, 151, 210)*

hsl = first\_color**.**hsl #*e.g. (230, 255, 203)*

proportion  = first\_color**.**proportion #*e.g. 0.34*

#*RGB and HSL are named tuples, so values can be accessed as properties.*

#*These all work just as well:*

red = rgb[0]

red = rgb**.**r

saturation = hsl[1]

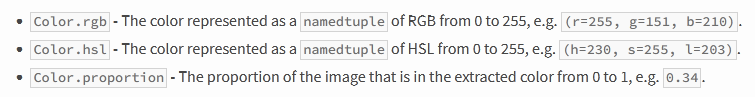
saturation = hsl**.**s

colorgram**.extract**(image, number\_of\_colors)

* Extract colors from an ***image***. image may be either a ***path*** to a ***file***, a ***file-like object***, or a ***Pillow Image object***. The function will return a list of ***number\_of\_colors*** ***Color objects***.

**colorgram.Color**

* A ***Color*** extracted from an image. Its properties are:



* Pillow: Python Imaging Library. Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors. As of 2019, Pillow development is supported by Tidelift.

Overview

* The Python Imaging Library adds image processing capabilities to your Python interpreter.
* This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
* The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

pip install pillow

pip install colorgram.py

**18.6 File I/O in Python**

Following are the various File Modes in Python:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Mode** | **Description** | | **‘r’** | This is the default mode. It Opens file for reading. | | **‘w’** | This Mode Opens file for writing. If file does not exist, it creates a new file. If file exists it truncates the file. | | **‘x’** | Creates a new file. If file already exists, the operation fails. | | **‘a’** | Open file in append mode. If file does not exist, it creates a new file. | | **‘t’** | This is the default mode. It opens in text mode. | | **‘b’** | This opens in binary mode. | | **‘+’** | This will open a file for reading and writing (updating)  **'a+'** or **'w+'** | | Step 1) **Open the .txt file**  f= **open**("guru99.txt","w+")  Step 2) **Enter data into the file**  for i in range(10):  f.**write**("This is line %d\r\n" % (i+1))  or, f.**read**()  Step 3) **Close the file instance**  f.**close**() |

* Exercise 18.7: Final Project. Damien Hurst's Dots.

Practice version

1. The Hirst Painting Project Part 1: - How to Extract RGB Values from Images.

"""

# ----------- Extractiing colors from image -------------

import colorgram

# Extract 6 colors from an image.

colors = colorgram.extract('Hirst\_dots.JPG', 50)

# print(colors[0].Rgb)

# colorgram.extract returns Color objects, which let you access

# RGB, HSL, and what proportion of the image was that color.

first\_color = colors[0]

rgb = first\_color.rgb # e.g. (255, 151, 210)

hsl = first\_color.hsl # e.g. (230, 255, 203)

proportion  = first\_color.proportion # e.g. 0.34

"""

#*create a list of rgb from extracted color and store the list in a .txt file*

"""

cloRs\_rgb = []

for i in range(0, len(colors)):

    cloRs\_rgb.append(colors[i].rgb)

fiLe = open("extracted\_color.txt", "a+")

fiLe.write(str(cloRs\_rgb))

"""

#*Final list from the extracted\_color.txt file*

**def** **Rgb**(r, g, b):

    tuP = (r, g, b)

**return** tuP

extracted\_color = [**Rgb**(r=239, g=236, b=238), **Rgb**(r=239, g=237, b=235), **Rgb**(r=235, g=237, b=242), **Rgb**(r=229, g=238, b=232), **Rgb**(r=26, g=109, b=164), **Rgb**(r=191, g=39, b=81), **Rgb**(r=234, g=160, b=54), **Rgb**(r=233, g=214, b=88), **Rgb**(r=220, g=137, b=175), **Rgb**(r=142, g=108, b=57), **Rgb**(r=105, g=195, b=217), **Rgb**(r=204, g=165, b=30), **Rgb**(r=20, g=57, b=131), **Rgb**(r=212, g=75, b=92), **Rgb**(r=235, g=89, b=51), **Rgb**(r=119, g=190, b=141), **Rgb**(r=140, g=28, b=72), **Rgb**(r=141, g=207, b=225), **Rgb**(r=105, g=107, b=195), **Rgb**(r=7, g=157, b=87), **Rgb**(r=7, g=184, b=175), **Rgb**(r=97, g=50, b=36), **Rgb**(r=21, g=162, b=203), **Rgb**(r=228, g=169, b=188), **Rgb**(r=30, g=91, b=94), **Rgb**(r=85, g=46, b=34), **Rgb**(r=31, g=44, b=85), **Rgb**(r=174, g=185, b=224), **Rgb**(r=150, g=214, b=193), **Rgb**(r=234, g=173, b=162), **Rgb**(r=29, g=94, b=93), **Rgb**(r=241, g=200, b=5), **Rgb**(r=97, g=22, b=52)]

**print**(f"extracted\_color : {extracted\_color}, and length : {len(extracted\_color)}")

1. The Hirst Painting Project Part 2: - Drawing the Dots

**from** turtle **import** Screen, Turtle

**import** random

tim = **Turtle**()

scr = **Screen**()

scr**.colormode**(255)

dot\_size = 20

step\_size = 40

tim**.penup**()

tim**.hideturtle**()

tim**.speed**("fastest")

tim**.left**(225)

tim**.forward**(198)

tim**.right**(225)

**for** i **in** **range**(1, 50):

    #*tim.pencolor((23, 45, 67))*

    tim**.dot**(dot\_size, random**.choice**(extracted\_color))

    tim**.forward**(step\_size)

**if** i%7 **==** 0:

        tim**.left**(90)

        tim**.forward**(step\_size)

        tim**.left**(90)

        tim**.forward**(7\*step\_size)

        tim**.left**(180)

scr**.exitonclick**()

#*python hirsts\_colordots.py*

* I downloaded an image of "Dots" by ***Damien*** ***Hurst***. By using ***colorgram***, which helps me to extract colours used frequently in the image.
* I extracted RGB values and made it into a tuple format.
* I looped through coloured dots randomly to create a masterpiece.

Instructors solution

"""

# ----------- Extractiing colors from image -------------

import colorgram

rgb\_colosr = []

colors = colorgram.extract('Hirst\_dots.JPG', 50)

for clrs in colors:

    r = clrs.rgb.r

    g = clrs.rgb.g

    b = clrs.rgb.b

    new\_colour = (r, g, b)

    rgb\_colosr.append(new\_colour)

print(rgb\_colosr)

"""

rgb\_colors = [(239, 236, 238), (239, 237, 235), (235, 237, 242), (229, 238, 232), (26, 109, 164), (191, 39, 81), (234, 160, 54), (233, 214, 88), (220, 137, 175), (142, 108, 57), (105, 195, 217), (204, 165, 30), (20, 57, 131), (212, 75, 92), (235, 89, 51), (119, 190, 141), (140, 28, 72), (141, 207, 225), (105, 107, 195), (7, 157, 87), (7, 184, 175), (97, 50, 36), (21, 162, 203), (228, 169, 188), (30, 91, 94), (85, 46, 34), (31, 44, 85), (174, 185, 224), (150, 214, 193), (234, 173, 162), (29, 94, 93), (241, 200, 5), (97, 22, 52)]

**import** turtle

**import** random

tim = turtle**.Turtle**()

scr = turtle**.Screen**()

scr**.colormode**(255)

tim**.penup**()

tim**.hideturtle**()

tim**.speed**("fastest")

tim**.setheading**(225)

tim**.forward**(300)

tim**.setheading**(0)

number\_of\_dots = 100

**for** dot\_count **in** **range**(1, number\_of\_dots + 1):

    tim**.dot**(20, random**.choice**(rgb\_colors))

    tim**.forward**(50)

**if** dot\_count%10 **==** 0:

        tim**.setheading**(90)

        tim**.forward**(50)

        tim**.setheading**(180)

        tim**.forward**(500)

        tim**.setheading**(0)

scr**.exitonclick**()

#*python hirsts\_colordots\_ins\_soln.py*

Tkinter color list:

COLORS  =['snow', 'ghost white', 'white smoke', 'gainsboro', 'floral white', 'old lace',

    'linen', 'antique white', 'papaya whip', 'blanched almond', 'bisque', 'peach puff',

    'navajo white', 'lemon chiffon', 'mint cream', 'azure', 'alice blue', 'lavender',

    'lavender blush', 'misty rose', 'dark slate gray', 'dim gray', 'slate gray',

    'light slate gray', 'gray', 'light grey', 'midnight blue', 'navy', 'cornflower blue', 'dark slate blue',

    'slate blue', 'medium slate blue', 'light slate blue', 'medium blue', 'royal blue',  'blue',

    'dodger blue', 'deep sky blue', 'sky blue', 'light sky blue', 'steel blue', 'light steel blue',

    'light blue', 'powder blue', 'pale turquoise', 'dark turquoise', 'medium turquoise', 'turquoise',

    'cyan', 'light cyan', 'cadet blue', 'medium aquamarine', 'aquamarine', 'dark green', 'dark olive green',

    'dark sea green', 'sea green', 'medium sea green', 'light sea green', 'pale green', 'spring green',

    'lawn green', 'medium spring green', 'green yellow', 'lime green', 'yellow green',

    'forest green', 'olive drab', 'dark khaki', 'khaki', 'pale goldenrod', 'light goldenrod yellow',

    'light yellow', 'yellow', 'gold', 'light goldenrod', 'goldenrod', 'dark goldenrod', 'rosy brown',

    'indian red', 'saddle brown', 'sandy brown',

    'dark salmon', 'salmon', 'light salmon', 'orange', 'dark orange',

    'coral', 'light coral', 'tomato', 'orange red', 'red', 'hot pink', 'deep pink', 'pink', 'light pink',

    'pale violet red', 'maroon', 'medium violet red', 'violet red',

    'medium orchid', 'dark orchid', 'dark violet', 'blue violet', 'purple', 'medium purple',

    'thistle', 'snow2', 'snow3',

    'snow4', 'seashell2', 'seashell3', 'seashell4', 'AntiqueWhite1', 'AntiqueWhite2',

    'AntiqueWhite3', 'AntiqueWhite4', 'bisque2', 'bisque3', 'bisque4', 'PeachPuff2',

    'PeachPuff3', 'PeachPuff4', 'NavajoWhite2', 'NavajoWhite3', 'NavajoWhite4',

    'LemonChiffon2', 'LemonChiffon3', 'LemonChiffon4', 'cornsilk2', 'cornsilk3',

    'cornsilk4', 'ivory2', 'ivory3', 'ivory4', 'honeydew2', 'honeydew3', 'honeydew4',

    'LavenderBlush2', 'LavenderBlush3', 'LavenderBlush4', 'MistyRose2', 'MistyRose3',

    'MistyRose4', 'azure2', 'azure3', 'azure4', 'SlateBlue1', 'SlateBlue2', 'SlateBlue3',

    'SlateBlue4', 'RoyalBlue1', 'RoyalBlue2', 'RoyalBlue3', 'RoyalBlue4', 'blue2', 'blue4',

    'DodgerBlue2', 'DodgerBlue3', 'DodgerBlue4', 'SteelBlue1', 'SteelBlue2',

    'SteelBlue3', 'SteelBlue4', 'DeepSkyBlue2', 'DeepSkyBlue3', 'DeepSkyBlue4',

    'SkyBlue1', 'SkyBlue2', 'SkyBlue3', 'SkyBlue4', 'LightSkyBlue1', 'LightSkyBlue2',

    'LightSkyBlue3', 'LightSkyBlue4', 'SlateGray1', 'SlateGray2', 'SlateGray3',

    'SlateGray4', 'LightSteelBlue1', 'LightSteelBlue2', 'LightSteelBlue3',

    'LightSteelBlue4', 'LightBlue1', 'LightBlue2', 'LightBlue3', 'LightBlue4',

    'LightCyan2', 'LightCyan3', 'LightCyan4', 'PaleTurquoise1', 'PaleTurquoise2',

    'PaleTurquoise3', 'PaleTurquoise4', 'CadetBlue1', 'CadetBlue2', 'CadetBlue3',

    'CadetBlue4', 'turquoise1', 'turquoise2', 'turquoise3', 'turquoise4', 'cyan2', 'cyan3',

    'cyan4', 'DarkSlateGray1', 'DarkSlateGray2', 'DarkSlateGray3', 'DarkSlateGray4',

    'aquamarine2', 'aquamarine4', 'DarkSeaGreen1', 'DarkSeaGreen2', 'DarkSeaGreen3',

    'DarkSeaGreen4', 'SeaGreen1', 'SeaGreen2', 'SeaGreen3', 'PaleGreen1', 'PaleGreen2',

    'PaleGreen3', 'PaleGreen4', 'SpringGreen2', 'SpringGreen3', 'SpringGreen4',

    'green2', 'green3', 'green4', 'chartreuse2', 'chartreuse3', 'chartreuse4',

    'OliveDrab1', 'OliveDrab2', 'OliveDrab4', 'DarkOliveGreen1', 'DarkOliveGreen2',

    'DarkOliveGreen3', 'DarkOliveGreen4', 'khaki1', 'khaki2', 'khaki3', 'khaki4',

    'LightGoldenrod1', 'LightGoldenrod2', 'LightGoldenrod3', 'LightGoldenrod4',

    'LightYellow2', 'LightYellow3', 'LightYellow4', 'yellow2', 'yellow3', 'yellow4',

    'gold2', 'gold3', 'gold4', 'goldenrod1', 'goldenrod2', 'goldenrod3', 'goldenrod4',

    'DarkGoldenrod1', 'DarkGoldenrod2', 'DarkGoldenrod3', 'DarkGoldenrod4',

    'RosyBrown1', 'RosyBrown2', 'RosyBrown3', 'RosyBrown4', 'IndianRed1', 'IndianRed2',

    'IndianRed3', 'IndianRed4', 'sienna1', 'sienna2', 'sienna3', 'sienna4', 'burlywood1',

    'burlywood2', 'burlywood3', 'burlywood4', 'wheat1', 'wheat2', 'wheat3', 'wheat4', 'tan1',

    'tan2', 'tan4', 'chocolate1', 'chocolate2', 'chocolate3', 'firebrick1', 'firebrick2',

    'firebrick3', 'firebrick4', 'brown1', 'brown2', 'brown3', 'brown4', 'salmon1', 'salmon2',

    'salmon3', 'salmon4', 'LightSalmon2', 'LightSalmon3', 'LightSalmon4', 'orange2',

    'orange3', 'orange4', 'DarkOrange1', 'DarkOrange2', 'DarkOrange3', 'DarkOrange4',

    'coral1', 'coral2', 'coral3', 'coral4', 'tomato2', 'tomato3', 'tomato4', 'OrangeRed2',

    'OrangeRed3', 'OrangeRed4', 'red2', 'red3', 'red4', 'DeepPink2', 'DeepPink3', 'DeepPink4',

    'HotPink1', 'HotPink2', 'HotPink3', 'HotPink4', 'pink1', 'pink2', 'pink3', 'pink4',

    'LightPink1', 'LightPink2', 'LightPink3', 'LightPink4', 'PaleVioletRed1',

    'PaleVioletRed2', 'PaleVioletRed3', 'PaleVioletRed4', 'maroon1', 'maroon2',

    'maroon3', 'maroon4', 'VioletRed1', 'VioletRed2', 'VioletRed3', 'VioletRed4',

    'magenta2', 'magenta3', 'magenta4', 'orchid1', 'orchid2', 'orchid3', 'orchid4', 'plum1',

    'plum2', 'plum3', 'plum4', 'MediumOrchid1', 'MediumOrchid2', 'MediumOrchid3',

    'MediumOrchid4', 'DarkOrchid1', 'DarkOrchid2', 'DarkOrchid3', 'DarkOrchid4',

    'purple1', 'purple2', 'purple3', 'purple4', 'MediumPurple1', 'MediumPurple2',

    'MediumPurple3', 'MediumPurple4', 'thistle1', 'thistle2', 'thistle3', 'thistle4',

    'gray1', 'gray2', 'gray3', 'gray4', 'gray5', 'gray6', 'gray7', 'gray8', 'gray9', 'gray10',

    'gray11', 'gray12', 'gray13', 'gray14', 'gray15', 'gray16', 'gray17', 'gray18', 'gray19',

    'gray20', 'gray21', 'gray22', 'gray23', 'gray24', 'gray25', 'gray26', 'gray27', 'gray28',

    'gray29', 'gray30', 'gray31', 'gray32', 'gray33', 'gray34', 'gray35', 'gray36', 'gray37',

    'gray38', 'gray39', 'gray40', 'gray42', 'gray43', 'gray44', 'gray45', 'gray46', 'gray47',

    'gray48', 'gray49', 'gray50', 'gray51', 'gray52', 'gray53', 'gray54', 'gray55', 'gray56',

    'gray57', 'gray58', 'gray59', 'gray60', 'gray61', 'gray62', 'gray63', 'gray64', 'gray65',

    'gray66', 'gray67', 'gray68', 'gray69', 'gray70', 'gray71', 'gray72', 'gray73', 'gray74',

    'gray75', 'gray76', 'gray77', 'gray78', 'gray79', 'gray80', 'gray81', 'gray82', 'gray83',

    'gray84', 'gray85', 'gray86', 'gray87', 'gray88', 'gray89', 'gray90', 'gray91', 'gray92',

    'gray93', 'gray94', 'gray95', 'gray97', 'gray98', 'gray99']

#*python tk\_color\_list.py*