

```

1  """
2      Contributors: Jesus A. Bernal Lopez jebernal@csumb.edu
3                   Paul Whipp pwhipp@csumb.edu
4
5      Class: CST-205
6
7      Lab 3: Color Dictionaries
8
9      Last Modified: 02/02/2019
10
11     Due Date: 02/04/2019
12  """
13
14  # Dictionary declaration for task 1
15  color_dictionary = {
16      "red": (255, 0, 0),
17      "green": (0, 255, 0),
18      "blue": (0, 0, 255),
19      "magenta": (255, 0, 255),
20      "cyan": (0, 255, 255),
21      "yellow": (255, 255, 0),
22      "purple": (92, 40, 136),
23      "lime": (9, 234, 79),
24      "black": (0, 0, 0),
25      "white": (255, 255, 255),
26      "lemon": (255, 252, 81),
27      "gold": (206, 203, 10),
28      "gray": (165, 165, 145),
29  }
30
31  sentence_fragments = [
32      "The red channel of",
33      "The green channel of",
34      "The blue channel of",
35  ]
36
37  # I think "is" makes more sense than "has value" but hey I am not the
38  # instructor
39  has_value = "has value"
40
41  tineye_sample = {
42      "status": "ok",
43      "error": [],
44      "method": "extract_collection_colors",
45      "result": [
46          {
47              "color": (141, 125, 83),
48              "weight": 76.37,
49              "name": "Clay Creek",
50              "rank": 1,
51              "class": "Grey"
52          },
53          {
54              "color": (35, 22, 19),
55              "weight": 23.63,

```

```

55         "name": "Seal Brown",
56         "rank": 2,
57         "class": "Black"
58     }
59 ]
60 }
61
62
63 def print_dictionary(dict):
64     statement = "{\n"
65     for key in dict:
66         statement += f"\t{key}: {dict[key]},\n"
67     statement += "\n}"
68     print(statement)
69
70
71 """
72     Description: We got bored and decided to print the dictionary in
73     a pretty way, the dictionary is declared up top
74     Driver: Jesus
75     Navigator: Paul
76 """
77
78 def task1():
79     print_dictionary(color_dictionary)
80
81
82 """
83     Description: We defined sentence fragment as a global list and we
84     formed a sentence in the function asking for
85     the channel and color as arguments.
86     Driver: Jesus
87     Navigator: Paul
88 """
89
90 def task2_1(channel, color):
91     # channel values:
92     # red = 0
93     # green = 1
94     # blue = 2
95     print(f"{sentence_fragments[channel]} {color} {has_value} {
96     color_dictionary[color][channel]}")
97
98 """
99     Description: We were debating if the desired output was all the
100     values with the second letter being 'e' (task2_2)
101     or running the color through a check to see if the
102     second letter was 'e' then outputting the whole
103     tuple if it was (task2_3) and just the channel if it
104     was not.
105     Driver: Paul
106     Navigator: Jesus

```

```

104 """
105
106
107 def task2_2(dict):
108     for color in dict:
109         if color[1] == 'e':
110             print(f"{color} {has_value} {color_dictionary[color]}")
111
112
113 def task2_3(channel, color):
114     if color[1] == 'e':
115         print(f"The tuple of {color} {has_value} {color_dictionary[
116 color]}")
117     else:
118         task2_1(channel, color)
119
120 """
121     Description: We approached it from a couple different angles.
122     1. We printed out the desired result by just navigating the
123     dictionary in a hard coded way.
124     2. We made a more generalized function that allowed the
125     function to take the name, channel and dictionary
126     as arguments and output the result in a sentence as in
127     task 2.
128     Driver: Jesus
129     Navigator: Paul
130 """
131
132 def task3(name_color, channel, dictionary):
133
134     #####
135     #####
136     # This will do exactly what you asked for but below is a more
137     # general way if the dictionary was bigger and we did not know
138     # the location of it.
139
140     #####
141     #####
142     hard_coded_way_1 = tineye_sample["result"][0]["color"][0]
143     hard_coded_way_2 = tineye_sample["result"][1]["color"][2]
144     print(hard_coded_way_1)
145     print(hard_coded_way_2)
146
147     # result = dictionary["result"]
148     # for val in result:
149     #     if val["name"] == name_color:
150     #         print(f"{sentence_fragments[channel]} {name_color} {
151     # has_value} {val['color'][channel]}")
152
153
154 def main():
155     # task1()
156
157

```

```

149     # task2_1(2, 'magenta')
150     # task2_1(1, 'yellow')
151     # task2_1(0, 'cyan')
152
153     # task2_2(color_dictionary)
154
155     # task2_3(2, 'red')
156     # task2_3(1, 'blue')
157     # task2_3(2, 'lemon')
158
159     task3("Don't matter", "just to lazy to make another without", "
arguments")
160     # task3("Clay Creek", 0, tineye_sample)
161     # task3("Seal Brown", 2, tineye_sample)
162
163
164
165 if __name__ == "__main__":
166     main()
167
168
169 """
170 ****
171 *                               Summary                               *
172 ****
173
174 Overall we got good at navigating through dictionaries. A problem we
faced was debating what was asked for in task 2
175 but ultimately we overcame that by implementing both solutions.
176
177
178 ****
179 *                               Task 1                               *
180 ****
181
182
183 Function call: task1()
184 Output:
185 {
186     red: (255, 0, 0),
187     green: (0, 255, 0),
188     blue: (0, 0, 255),
189     magenta: (255, 0, 255),
190     cyan: (0, 255, 255),
191     yellow: (255, 255, 0),
192     purple: (92, 40, 136),
193     lime: (9, 234, 79),
194     black: (0, 0, 0),
195     white: (255, 255, 255),
196     lemon: (255, 252, 81),
197     gold: (206, 203, 10),
198     gray: (165, 165, 145),
199 }
200
201

```

```
202 *****
203 *                               Task 2                               *
204 *****
205
206 *****
207 *               Part 1               *
208 *****
209
210
211 Function calls: task2_1(2, 'magenta')
212                  task2_1(1, 'yellow')
213                  task2_1(0, 'cyan')
214 Output:
215 The blue channel of magenta has value 255
216 The green channel of yellow has value 255
217 The red channel of cyan has value 0
218
219
220 *****
221 *               Part 2               *
222 *****
223
224
225 Function call: task2_2(color_dictionary)
226 Output:
227 red has value (255, 0, 0)
228 yellow has value (255, 255, 0)
229 lemon has value (255, 252, 81)
230
231
232 *****
233 *               Part 3               *
234 *****
235
236
237 Function calls: task2_3(2, 'red')
238                  task2_3(1, 'blue')
239                  task2_3(2, 'lemon')
240 Output:
241 The tuple of red has value (255, 0, 0)
242 The green channel of blue has value 0
243 The tuple of lemon has value (255, 252, 81)
244
245
246 *****
247 *                               Task 3                               *
248 *****
249
250
251 Function calls: task3("Clay Creek", 0, tineye_sample)
252                  task3("Seal Brown", 2, tineye_sample)
253 Output:
254 The red channel of Clay Creek has value 141
255 The blue channel of Seal Brown has value 19
256
```

```
257 Function call(hard coded way): task3("Don't matter", "just to lazy to  
    make another without", "arguments")  
258 141  
259 19  
260  
261  
262 .....
```