Working with Images

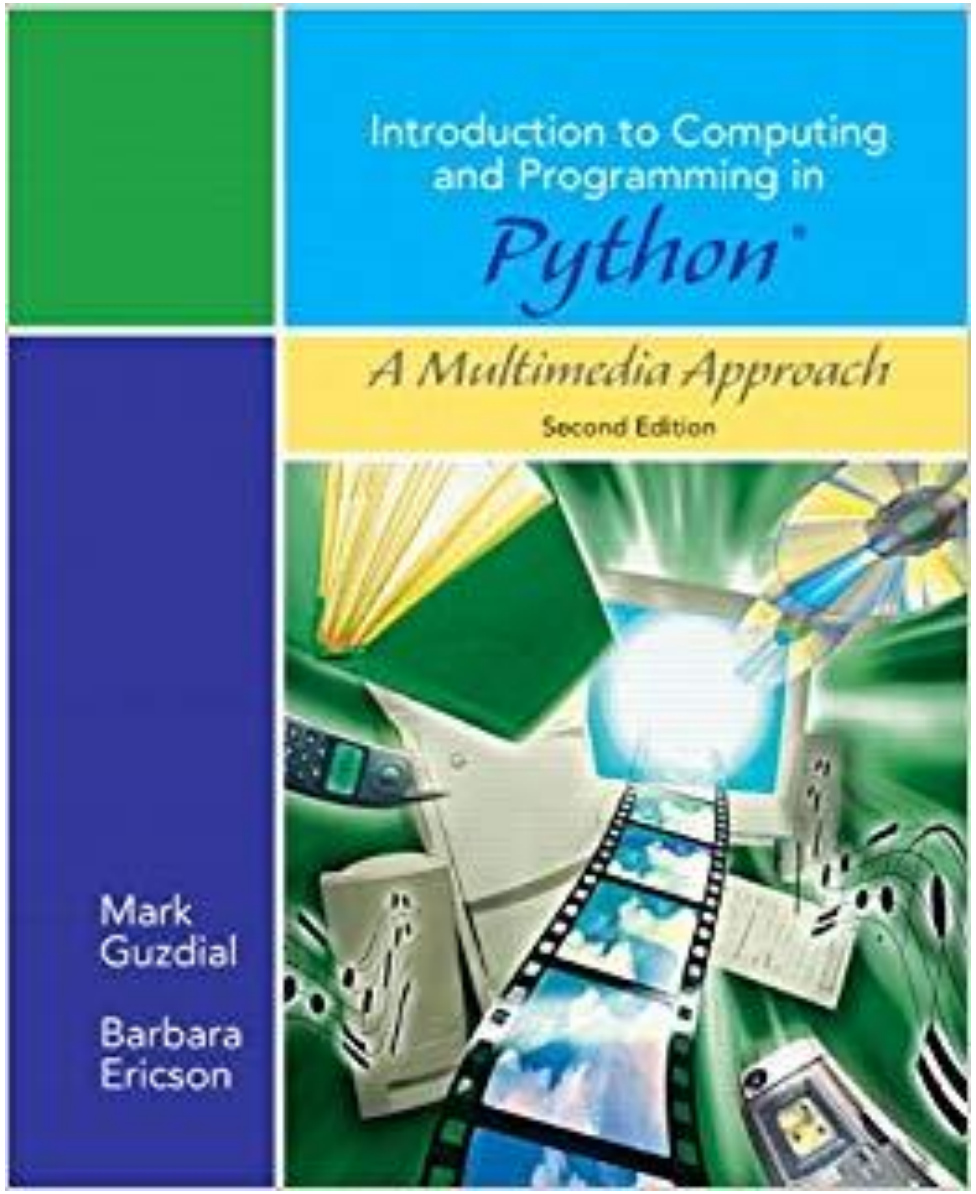
CMM201 – Programming Concepts for Business Analytics

Week 9

Dr Carlos Moreno-Garcia

Why?

- Images (and multimedia in general) are easy to understand analogies of data!
- Allow you to understand data input/output in an easier way.
- Is more visual!



How does an image look “digitally”?

- As you may know, there are many image file **extensions:**
  - jpeg
  - png
  - tiff
  - etc.
- These are just some of the many **compression algorithms** that exist to convert an image into a file.

Compression

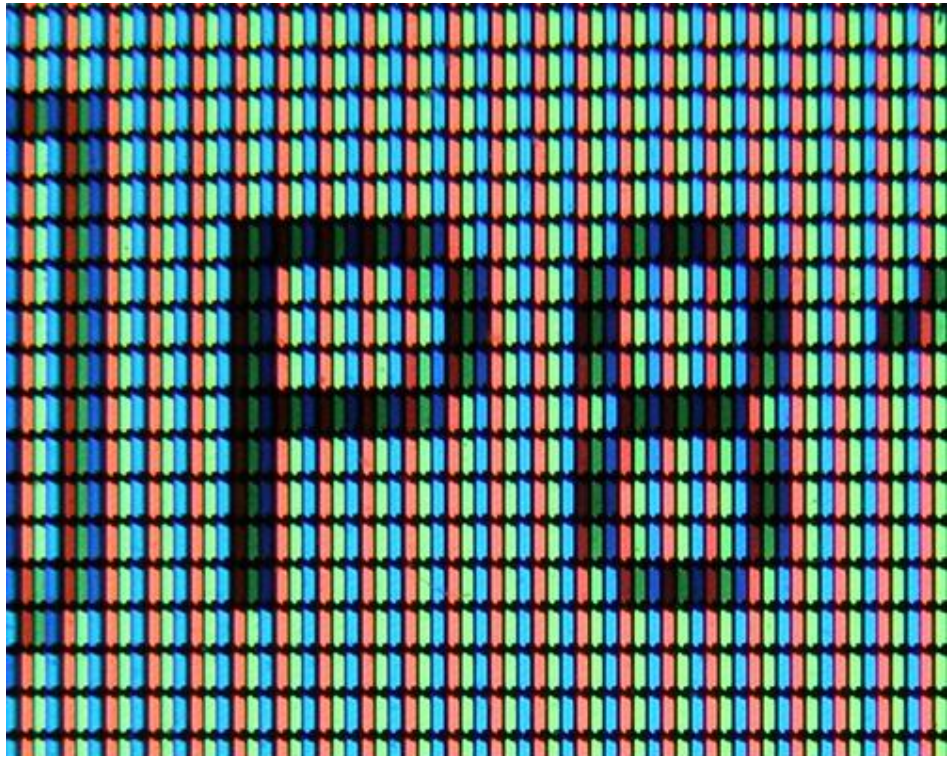
- The “art” of **compression algorithms** is in quantization!
  - The best algorithms are the ones who achieve the best visual quality of the image once this is not analogue anymore.
- However, that is NOT our problem!
  - We just want to work with the image as a pixel array.
- We are going to work with images in a simpler, **uncompressed** way…

8.9M

68.34K

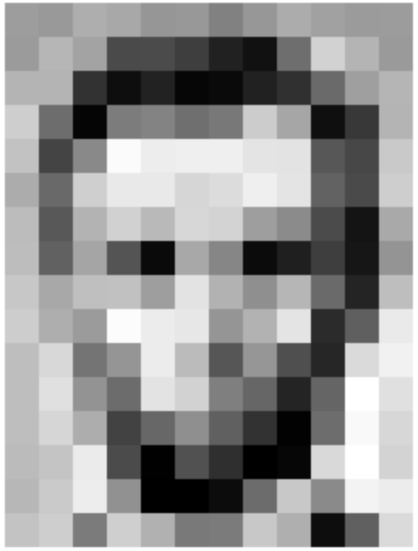
As numpy arrays!

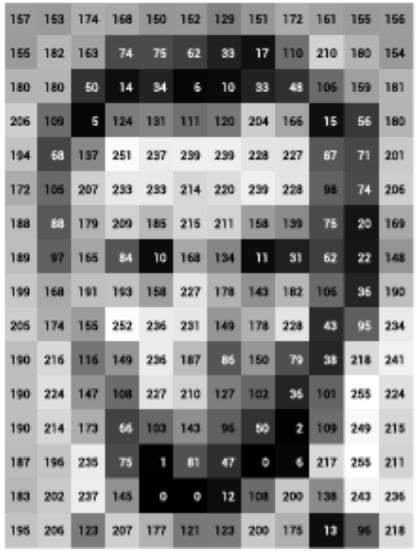
- Well, as arrays/matrices (in a more general sense…)
- When we visualise an image, in reality what we are looking at is a bunch of **pixels** arranged in our screen.
- Each of these pixels has a different **pixel value,** which indicates the colour at that given positions.



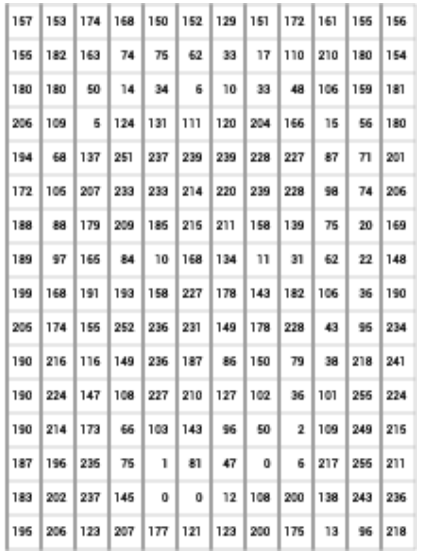
Grayscale images

- A 2D-grid of pixels.
- Two ways to represent grayscale:
  - **Standard**: from 0 (black) to 255 (white).
  - **Normalised**: from 0 (black) to 1 (white).

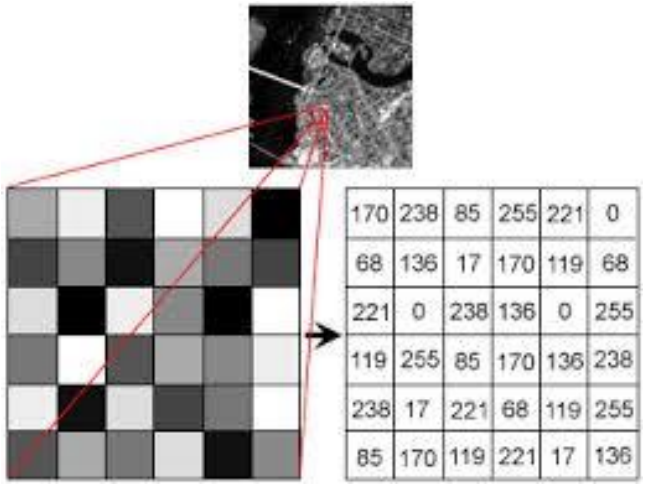




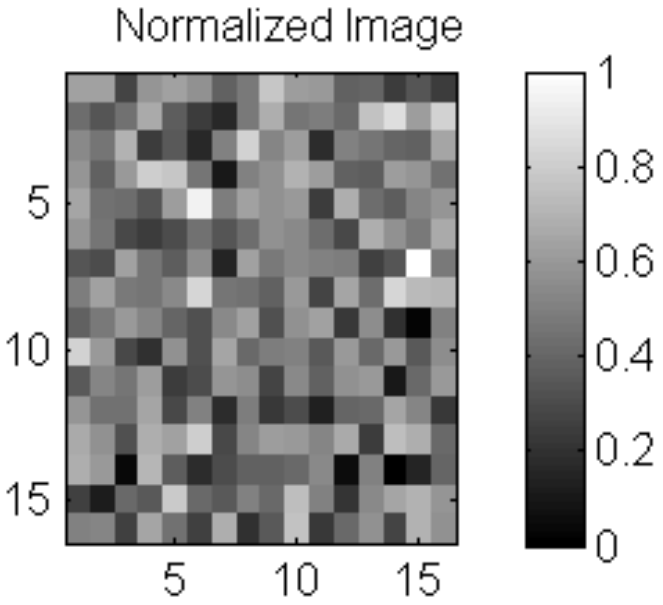
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

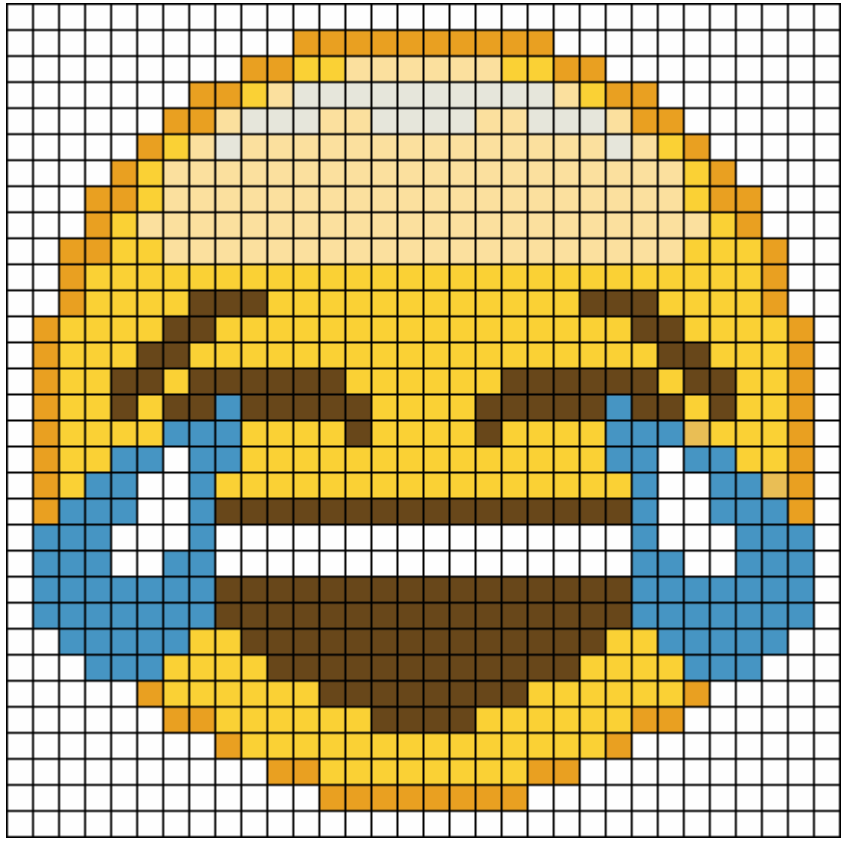


170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136



Can you convert between standard↔normalised?

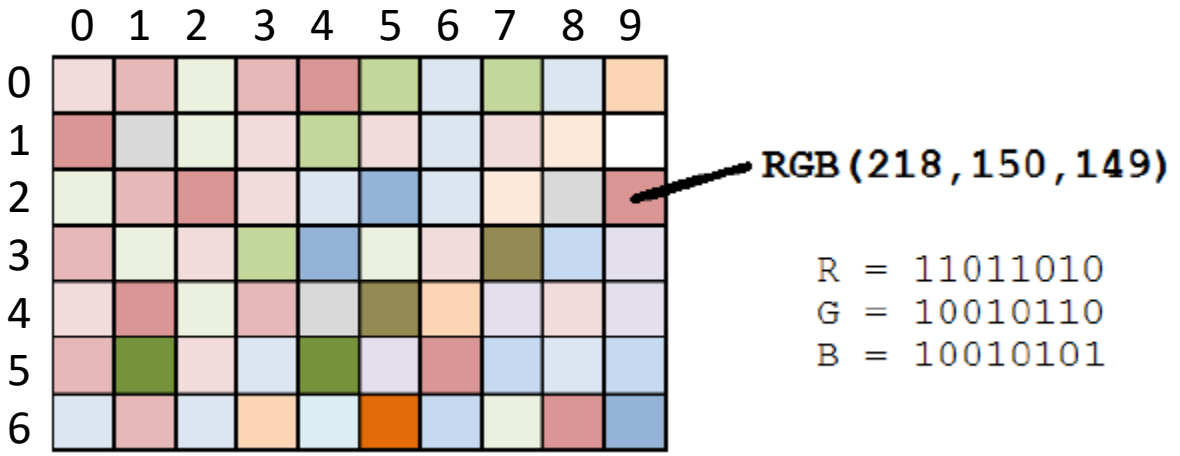
What about colour images?



Do you know who she is?

Colour Images

- Each pixel has three values (also called channels):
  - Red
  - Green
  - Blue
- That's why images with colour are often called RGB images.
- If this image is imported in Python, the pixel in row 2-column 9 has a value of 218 in the red channel, 150 in the green channel and 149 in the blue channel.
- This value would be stored in a **tuple**.



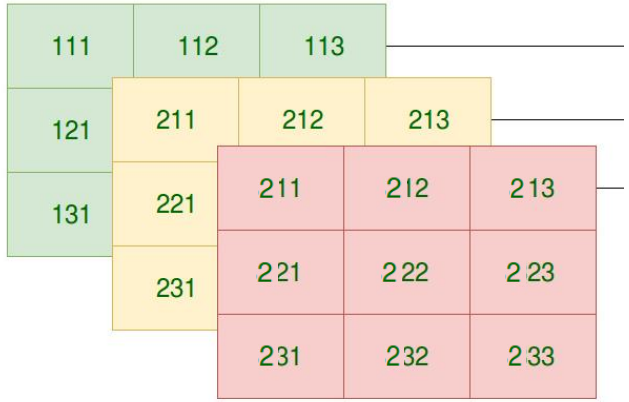
How many colours can be represented using the RGB standard?

Is there another standard that can represent more colours?

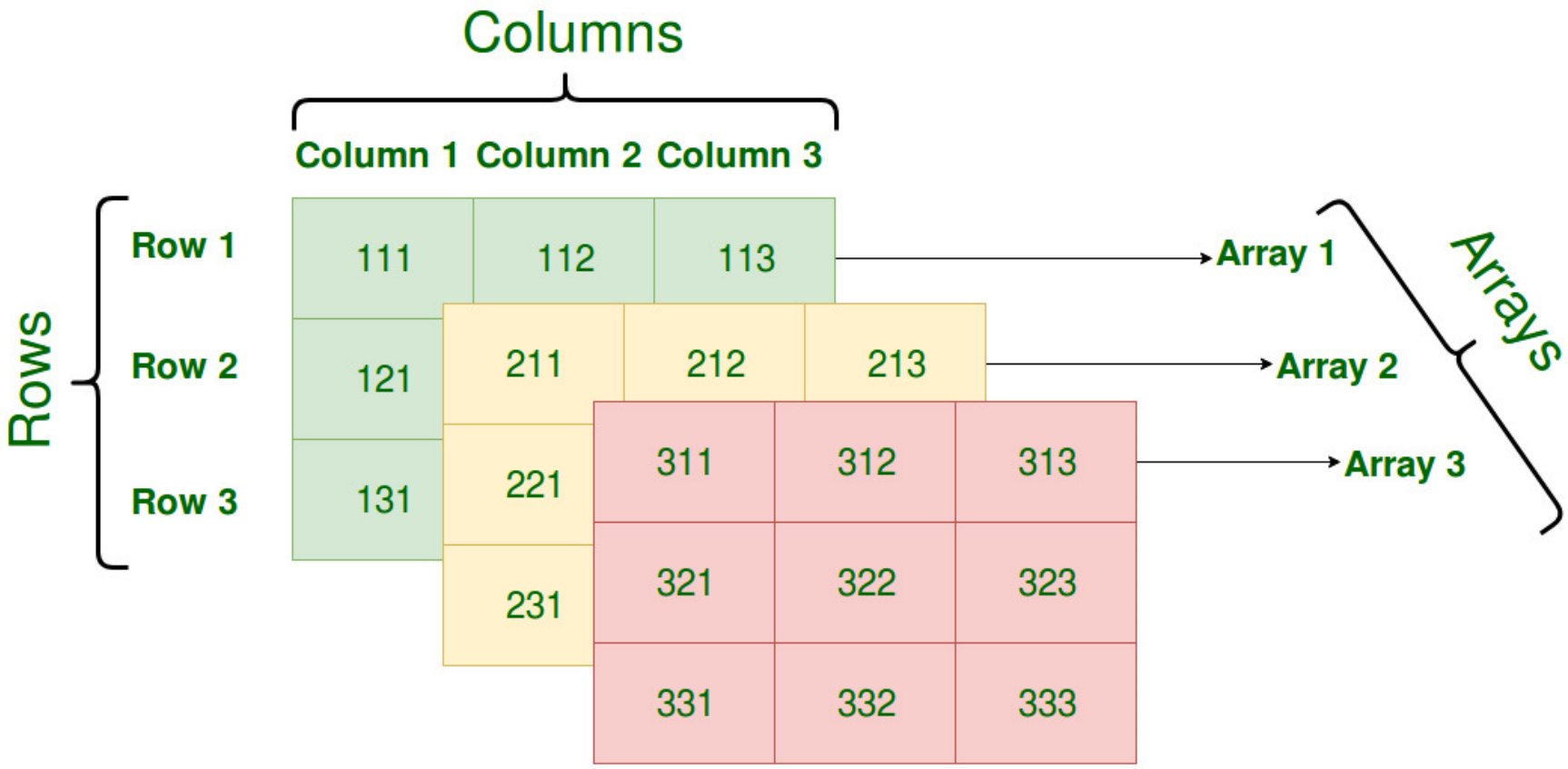
Colour Image Representation in numpy

- To import an colour image in a numpy array, we would have two options:
  - A two dimensional array where each position contains a tuple.
    - This is possible! Last time we stored strings in numpy arrays.
    - Problem: Calculations would be harder to do.
  - A three dimensional array with layers.

(211,211,111)	(212,212,112)	(213,213,113)
(221,121,121)	(222,222,122)	(223,223,123)
(231,231,131)	(232,232,132)	(233,233,133)



How would you “query” a certain pixel using numpy array terminology?



Note: These are **not** the values, these are the positions!