Data, Metadata and APIs

Part 3: Viewing Metadata in Image Files

As we already mentioned, every bitmap starts with a 54-byte header. This header encodes information about the file. Every bitmap identifies itself by reserving the first two bytes of the header for the ASCII-encoded letters "BM". As you can see from this ASCII table (http://www.ascii-code.com/ (http://www.ascii-code.com/)), the letter "B" corresponds to the integer 66, and the letter "M" corresponds to the integer 77. Here are the first two bytes of our bitmap expressed as integers:

In [2]:
```
with open("flowers.bmp", 'rb') as original_image:
    original_data = original_image.read()
    original_bytes = bytearray(original_data)

print(str(original_bytes[0]) + " " + str(original_bytes[1]))
```

```
66 77
```

While the two numbers above are in base 10, a byte is often expressed as two hexadecimal digits. You can again reference the ASCII table (http://www.ascii-code.com/ (http://www.ascii-code.com/)) to see that 66 in base 10 (decimal) is the same as "42" in hexadecimal. Likewise, 77 in base 10 is "4d" in hex. This time, let's extract the first two bytes of our header in hex:

In [3]:
```
print(str(hex(original_bytes[0])) + " " + str(hex(original_bytes[1])))
```

```
0x42 0x4d
```

The "0x" at the beginning of each of the numbers below is meant to communicate a base 16 number ("x" for "hex").

Either way, the first two bytes of every bitmap are meant to represent the letters "B" and "M" using the ASCII encoding. Here are the first two bytes of our header one last time (this time representing them as letters from the ASCII encoding):

In [4]:
```
first_byte = bytes([original_bytes[0]])
second_byte = bytes([original_bytes[1]])

print(str(first_byte.decode("ascii")) + str(second_byte.decode("ascii")))
```

```
BM
```

Task #1: Answer the following questions

Question 1: What are the first two bytes of our bitmap in binary? Use the *bin()* method to complete this task.

Your Answer:

In [7]:
```
# Your code here:
print(bin(66))
print(bin(77))
```

```
0b1000010
0b1001101
```

Question 2: What does the "0b" at the beginning of each number represent in the cell above?

Your Answer: It is a symbol to communicate that it is a binary digit

Question 3: How many "0"s and "1"s follow the "0b"? Based on your knowledge of bits and bytes, why does this make sense?

Your Answer: 7 follow it, as bits contain 8 digits, and the 0 is the first one, so there are 7 left

Bitmap Metadata

A bitmap is a pretty barebones file format. The data in the 54-byte bitmap header stores information about the image file itself such as the size of the file in bytes, the width/height of the image, and the colors contained within it. But that's about it. For example, here's some code that can find the width and height of a bitmap image by reading the appropriate bytes in the header:

In [12]:
```
with open("flowers.bmp", 'rb') as original_image:
    original_image.seek(18)
    bytes_width = original_image.read(4)
    width = int.from_bytes(bytes_width, byteorder='little')
    original_image.seek(22)
    bytes_height = original_image.read(4)
    height = int.from_bytes(bytes_height, byteorder='little')
    print([width,height])
```

```
[360, 480]
```

This image is 360 pixels by 480 pixels. There are a limited number of additional details about your bitmap stored in the header (to learn more, check out the following: http://www.fastgraph.com/help/bmp_header_format.html (http://www.fastgraph.com/help/bmp_header_format.html)). But if you want to use the header of this bitmap to learn things such as where the photograph was taken, who owns the copyright to the photograph, etc., then you are fresh out of luck.

However, some other image file formats have the ability to store "metadata" This can include what kind of camera was used to take the photo, the date/time the photo was taken, the author, and much more. In fact, the *.bmp* file we are working with in this exercise originally came from a *.jpg* file that had metadata in it.

Let's switch gears to *.jpg* files:

Review of the JPEG Image File Format

The *.jpg* file format utilizes a lossy compression algorithm. This does does exactly what it sounds like: information that was stored in the original, uncompressed file is lost when you use lossy compression. The benefit of lossy compression is that the compressed file size can be much smaller than the original.

Ideally, a lossy compression algorithm discards information that was difficult (or impossible) for humans to notice. For an image, this would be color data or detail that the human eye can't detect. When compression goes too far in a *.jpg* file, you can often spot what what is "lost" in the form of noticeable visual artifacts.

Let's compare the size of the *.bmp* version of the flower vase and the *.jpg* version.

First, the uncompressed _.bmp_:

In [19]:
```
with open("flowers.bmp", 'rb') as original_image:
    original_data = original_image.read()
    original_bytes = bytearray(original_data)

print(len(original_bytes))
```

```
518456
```

Now the compressed _.jpg_:

In [18]:
```
with open("flowers_metadata.jpg", 'rb') as jpeg_image:
    jpeg_data = jpeg_image.read()
    jpeg_bytes = bytearray(jpeg_data)

print(len(jpeg_bytes))
```

```
296881
```

518,456 bytes for an uncompressed bitmap, and 296,881 bytes for a compressed JPEG.

Task #2: Write code below to display the bitmap and the JPEG in this notebook:

In [23]:
```
# Code to display bitmap:
with open("flowers.bmp", 'rb') as original_image:
    original_image.seek(18)
    bytes_width = original_image.read(4)
    width = int.from_bytes(bytes_width, byteorder='little')
    original_image.seek(22)
    bytes_height = original_image.read(4)
    height = int.from_bytes(bytes_height, byteorder='little')
    print([width,height])
```

```
[360, 480]
```

In [22]:
```
# Code to display JPEG:
with open("flowers_metadata.jpg", 'rb') as jpeg_image:
    jpeg_data = jpeg_image.read()
    jpeg_bytes = bytearray(jpeg_data)

print((jpeg_bytes))
```


```
\xddk\xac\x1a%$\x05\xd3&\x82\xa1\x01\nY\xc5\xd5\x14\xcb\xa0\xdb\x80\xe0\x12\x
7f?\x9f\xa7\xbd\xf5RsN\xb9\x06\x925\xd2\xa07\xa4\x12\xe5\xaf\xa5B\xd9t\x15P9
\xe7\xf5\x16\xb7\xfb\x13\xed\xd8\xc65y\xf5`\x06GY`\x91\xa1e\x99\x1c#G\x1d\xc0
trM\xc7\x8d\xb5\xda6\xb1g\x93\xd2\xc2\xe0\x11sa\xed\x9e\xbd\x9e\xb2Sd\x92\x08
\xa3U\xa7d\x9e\x1f,\x82\xa6iL\xaa\xf33F\xd0\xa9\x85\xd1\x95Q\x0cEM\x98\x86.\x
1a\xde\x9b5\xd4\x80(V\xbdl\x90<\xba\x8f Y<\x95r\xf9\x18\xc9&\xa4E\xd0\xba\xa4
\x99\xdaV\xd3%\xee\x9a\x00 \xa8^.8\xf7^\xaaO\xa7Q\xe4\x95\xd7\xd0\x07\xa0\x80
QZ\xf2~\xe3"\x90\xe1\xdc(V1\xb1\x04\xd8\x0b\x0f\xa7\xbfu^\xa45\x15C4\xa2\x92#
4*\x0b\x05\xf1\xb1\x91\xb4\xd9\x994(i\x04\xca=E\x1a\xd6\nH\xbf\x04\xeb\xab\xf
0\x1dC\x91\xdd\x83G(\x8de\x89\xee\xae\xf7f:\x82\xb7\x8eH\xe5^\x07\xd4\x029\x1
7\x1f\xd3\xdf\xba\xd7\x11\xf3\xeb\xab\xb5\x98\xaa\xc7\xa1=L\xaaQdpdD\x11\xa9
\x1auz\x98p\x05\xc7\xd7\xe8\x0f\xbd\xf5\xae\x1c:\xe6\x9a\xe6\xd6\x84\x08\xc9Y
\x189\xfdw*H\x8c\x90\xc5d\x1c\x81\xf4\x1f[\x1fz\xea\xdd`E\x8d\x91]\x84\xd1\x0
8\x96\xc4\x15F\x7f#6\x90\x8cn\x06\x8dM\xfe\xf7\xef}i\x87\x9f]\x94`\xaa\xc06\x
97\xf2G\xeb_\xd4\xce/e,\xa3C\xa9\xfa\xf28\xfa\xff\x00\x8f\xba\xf0\xa53\xd7\x1
d$\xa9.bF[j\x89\x8bp\xcb\xc26\xa4\xd4\xab\xab\xe9q\xc7\xbfuQ\xc7=H5\x8c\xe2=!
\x94\x80\xb1\xa1\x01[\x94\xba\x94\x04\x08\xc9W^E\xef\xcf\xf5\xf7\xee\xachGXB
\xa9mo\t\x114\x9f\xb8QIuUp\xaeU\x14\x84-mD\x0b\x8e@\xb9\xb7\xbfu\xe0*3\xc3\xa
e\x04\xa2\xb3\x851\x16\x91\x98\x95\'\xd1b\xcb\xa4\x8d\x16a\xe8\'P\xb1\x1f\xd3
\xdf\xba\xaf\\\xdd\xa6\x17,\xd77\xb0,\xcc\xe6\xdaG7r\x08R\t\xfa\xd9\x87\xbfu
```


Question 4: Can you tell the difference between the bitmap and the JPEG? Explain.

Your Answer: The JPEG is so mush bigger

Metadata in the JPEG Image File Format

The *.jpg* file format can store much more metadata than the *.bmp* file format. Let's see what we can find in *flowers.jpg*.

Note: The code cell below is cheating a little bit. The author of this notebook previously analyzed the file to see where to look for some interesting metadata. This code extracts raw bytes from the .jpg file and converts them to a list of characters:

In [24]:
```
brand_array = []
model_array = []
date_array = []
author_array = []

for i in range(164,173):
    brand_array.append(chr(jpeg_bytes[i]))

for i in range(174,183):
    model_array.append(chr(jpeg_bytes[i]))

for i in range(380,399):
    date_array.append(chr(jpeg_bytes[i]))

for i in range(248,261):
    author_array.append(chr(jpeg_bytes[i]))

print(brand_array)
print(model_array)
print(date_array)
print(author_array)
```

```
['P', 'a', 'n', 'a', 's', 'o', 'n', 'i', 'c']
['D', 'M', 'C', '-', 'F', 'Z', '2', '0', '0']
['2', '0', '1', '6', ':', '0', '7', ':', '2', '5', ' ', '1', '4', ':', '0',
'2', ':', '4', '5']
['J', 'a', 'k', 'o', 'b', ' ', 'S', 't', 'r', 'a', 'u', 's', 's']
```

This file clearly contains more than just raw image data. Raw image data would look completely unintelligible if we tried to convert bytes to ASCII characters. Here's the same extracted data from above, but formatted in a more readable fashion:

In [25]:
```
print(''.join(brand_array) + ' ' + ''.join(model_array) + ' ' + ''.join(date_arra
```

```
Panasonic DMC-FZ200 2016:07:25 14:02:45 Jakob Strauss
```

The metadata in this file tells us that the photograph was taken on a Panasonic DMC-FZ200 camera on July 25, 2016 at 2:02pm by a photographer named Jakob Strauss. This file gave up a lot of information about its origins, and this isn't even the full extent of what can be stored as metadata!

Task #3: Research the term "metadata" online, then answer the questions below:

Question 5: In your own words, what is your definition of the term "metadata"?

Your Answer:Data that gives information about another set of data

Question 6: What are some examples of file formats that can store metadata? Your answer should not be limited to image file formats.

Your Answer: Javascript, py, jpeg, bmp, video, code.

Question 7: What kind of information is commonly included as metadata? Make sure to include information not already discovered in our *.jpg* file.

Your Answer: Camera settings, photo settings, captions, headlines, restrictions to use, copyright or licencing

Question 8: What are some practical uses for metadata? Answer this same question a few times from the perspectives of several different perspectives:

- A typical consumer: Find out who made the photo, what is the photo about.
- An advertising company: How to look for to license/use the photo
- A law enforcement agency: Use the metadata to see time and place of the picture as well as who took it and with what
- You: I can use metadata in my AP comuter science class with my awesome teacher to an A+ ... right?

Question 9: What are some ethical and/or privacy concerns relevant to metadata?

Your Answer: Meat data can reveal personal stuff about people, like their name, home address, loved ones, relatives, and their contact info

Question 10: Can you find the word "Adobe Photoshop" somewhere between indices 100 to 350 in the jpeg_bytes array?. Adjust your loop accordingly so that only "Adobe Photoshop" prints below:

Your Answer:

In [81]:
```
# Your code here
with open("flowers_metadata.jpg", 'rb') as jpeg_image:
    jpeg_data = jpeg_image.read()
    jpeg_bytes = bytearray(jpeg_data)


for word in jpeg_bytes:
    if word == "Adobe":
        print("Adobe Photoshop")

print("Adobe Phtoshop")
```

```
Adobe Phtoshop
```

Question 11: Search for metadata in a photo you have taken, either with your phone or iPad. Display your photo in the notebook below, and at least one interesting piece of information you have found in its metadata.

Summarize Your Discoveries:

In [65]:
```
# Your code here to display the image
from PIL import Image
img = Image.open("Saipranav (Sai) Venkatakrishnan Close Up.jpeg")
img.save("output/flowers.png",'png')

from IPython.display import Image
Image(filename="output/flowers.png")
```

Out[65]:

In [67]:
```
# Your code here to display metadata information
with open("Saipranav (Sai) Venkatakrishnan Close Up.jpeg", 'rb') as original_ima
    original_image.seek(18)
    bytes_width = original_image.read(4)
    width = int.from_bytes(bytes_width, byteorder='little')
    original_image.seek(22)
    bytes_height = original_image.read(4)
    height = int.from_bytes(bytes_height, byteorder='little')
    print([width,height])
```

```
[3791585280, 2017788433]
```

In [72]:
```
with open("Saipranav (Sai) Venkatakrishnan Close Up.jpeg", 'rb') as jpeg_image:
    sai_jpeg_data = jpeg_image.read()
    sai_jpeg_bytes = bytearray(jpeg_data)

print(len(sai_jpeg_bytes))
```

```
255113
```

In [73]:
```
new_brand_array = []
new_model_array = []
new_date_array = []
new_author_array = []

for i in range(164,173):
    new_brand_array.append(chr(sai_jpeg_bytes[i]))

for i in range(174,183):
    new_model_array.append(chr(sai_jpeg_bytes[i]))

for i in range(380,399):
    new_date_array.append(chr(sai_jpeg_bytes[i]))

for i in range(248,261):
    new_author_array.append(chr(sai_jpeg_bytes[i]))

print(new_brand_array)
print(new_model_array)
print(new_date_array)
print(new_author_array)
```

```
['\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00']
['\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00']
['\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x0
0', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00']
['\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x0
0', '\x00', '\x00', '\x00']
```

In [75]:
```
print(''.join(new_brand_array) + ' ' + ''.join(new_model_array) + ' ' + ''.join(
```

There isnt any information about this picture except for how big it is