Binary Files

Not all files contain text. Some of them contain pictures of my dog.

```
an_image = open('beauregard.jpg', mode='rb')                    #①
print (an_image.mode)                                           #②
#rb

print (an_image.name)                                           #③
#'beauregard.jpg'

print (an_image.encoding)                                       #④
#Traceback (most recent call last):
#   File "__ed_file.py", line 8, in <module>
# print (an_image.encoding) #\u2463
#AttributeError: '_io.BufferedReader' object has no attribute 'encoding'
```

① Opening a file in binary mode is simple but subtle. The only difference from opening it in text mode is that the `mode` parameter contains a `'b'` character.

② The stream object you get from opening a file in binary mode has many of the same attributes, including `mode`, which reflects the mode parameter you passed into the `open()` function.

③ Binary stream objects also have a `name` attribute, just like text stream objects.

④ Here's one difference, though: a binary stream object has no `encoding` attribute. That makes sense, right? You're reading (or writing) bytes, not strings, so there's no conversion for Python to do. What you get out of a binary file is exactly what you put into it, no conversion necessary.

Did I mention you're reading bytes? Oh yes you are.

```
# continued from the previous example
```

```
an_image = open('beauregard.jpg', mode='rb')
print (an_image.tell())
#0


data = an_image.read(3)          #①
print (data)
#b'\xff\xd8\xff'

print (type(data) )              #②
#<class 'bytes'>

print (an_image.tell())          #③
#3

print (an_image.seek(0))
#0

data = an_image.read()
print (len(data))
#3150
```

① Like text files, you can read binary files a little bit at a time. But there's a crucial difference…

② …you're reading bytes, not strings. Since you opened the file in binary mode, the `read()` method takes *the number of bytes to read*, not the number of characters.

③ That means that there's never an unexpected mismatch between the number you passed into the `read()` method and the position index you get out of the `tell()` method. The `read()` method reads bytes, and the `seek()` and `tell()` methods track the number of bytes read. For binary files, they'll always agree.