Each of those tags can be accessed with the `[]` operator on the metadata, much like a python dictionary:

```
>>> tag = metadata['Exif.Image.DateTime']
```

The value of an `ExifTag` object can be accessed in two different ways: with the *`raw_value`* and with the `value` attributes:

```
>>> tag.raw_value
'2004-07-13T21:23:44Z'

>>> tag.value
datetime.datetime(2004, 7, 13, 21, 23, 44)
```

The raw value is always a byte string, this is how the value is stored in the file. The value is lazily computed from the raw value depending on the EXIF type of the tag, and is represented as a convenient python object to allow easy manipulation.

Note that querying the value of a tag may raise an `ExifValueError` if the format of the raw value is invalid according to the EXIF specification (may happen if it was written by other software that implements the specification in a broken manner), or if pyexiv2 doesn't know how to convert it to a convenient python object.

Accessing the value of a tag as a python object allows easy manipulation and formatting:

```
>>> tag.value.strftime('%A %d %B %Y, %H:%M:%S')
'Tuesday 13 July 2004, 21:23:44'
```

Now let's modify the value of the tag and write it back to the file:

```
>>> import datetime
>>> tag.value = datetime.datetime.today()

>>> metadata.write()
```

Similarly to reading the value of a tag, one can set either the *`raw_value`* or the `value` (which will be automatically converted to a correctly formatted byte string by pyexiv2).

You can also add new tags to the metadata by providing a valid key and value pair (see exiv2's documentation for a list of valid EXIF tags):

```
>>> key = 'Exif.Photo.UserComment'
>>> value = 'This is a useful comment.'
>>> metadata[key] = pyexiv2.ExifTag(key, value)
```

As a handy shortcut, you can always assign a value for a given key regardless of whether it's already present in the metadata. If a tag was present, its value is overwritten. If the tag was not present, one is created and its value is set:

```
>>> metadata[key] = value
```

The EXIF data may optionally embed a thumbnail in the JPEG or TIFF format. The thumbnail can be accessed, set from a JPEG file or buffer, saved to disk and erased:

```
>>> thumb = metadata.exif_thumbnail
>>> thumb.set_from_file('/tmp/thumbnail.jpg')
>>> thumb.write_to_file('/tmp/copy')
>>> thumb.erase()
>>> metadata.write()
```