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

```
>>> metadata[key] = values
```

The IPTC metadata in an image may embed an optional character set for its encoding. This is defined by the `Iptc.Envelope.CharacterSet` tag. The `ImageMetadata` class has an *`iptc_charset`* property that allows to easily get, set and delete this value:

```
>>> metadata.iptc_charset
'ascii'

>>> metadata.iptc_charset = 'utf-8'

>>> del metadata.iptc_charset
```

Note that at the moment, the only supported charset that can be assigned to the property is `utf-8`. Also note that even if the charset is not explicitly set, its value may be inferred from the contents of the image. If not, it will be `None`.

2.3 Reading and writing XMP tags

Reading and writing XMP tags works pretty much the same way as with EXIF tags. Let's retrieve the list of all available XMP tags in the image:

```
>>> metadata.xmp_keys
['Xmp.dc.creator',
 'Xmp.dc.description',
 'Xmp.dc.rights',
 'Xmp.dc.source',
 'Xmp.dc.subject',
 'Xmp.dc.title',
 'Xmp.xmp.CreateDate',
 'Xmp.xmp.ModifyDate']
```

Each of those tags can be accessed with the `[]` operator on the metadata:

```
>>> tag = metadata['Xmp.xmp.ModifyDate']
```

As with EXIF tags, the value of an `XmpTag` object can be accessed in two different ways: with the *`raw_value`* and with the `value` attributes:

```
>>> tag.raw_value
'2002-07-19T13:28:10'

>>> tag.value
datetime.datetime(2002, 7, 19, 13, 28, 10)
```

Note that querying the value of a tag may raise an `XmpValueError` if the format of the raw value is invalid according to the XMP 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.

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