

2.2.1. Source Code Encoding

It is possible to use encodings different than ASCII in Python source files. The best way to do it is to put one more special comment line right after the `#!` line to define the source file encoding:

```
# -*- coding: encoding -*-
```

With that declaration, all characters in the source file will be treated as having the encoding *encoding*, and it will be possible to directly write Unicode string literals in the selected encoding. The list of possible encodings can be found in the Python Library Reference, in the section on [codecs](#).

For example, to write Unicode literals including the Euro currency symbol, the ISO-8859-15 encoding can be used, with the Euro symbol having the ordinal value 164. This script, when saved in the ISO-8859-15 encoding, will print the value 8364 (the Unicode code point corresponding to the Euro symbol) and then exit:

```
# -*- coding: iso-8859-15 -*-

currency = u"€"
print ord(currency)
```

If your editor supports saving files as UTF-8 with a UTF-8 *byte order mark* (aka BOM), you can use that instead of an encoding declaration. IDLE supports this capability if `Options/General/Default SourceEncoding/UTF-8` is set. Notice that this signature is not understood in older Python releases (2.2 and earlier), and also not understood by the operating system for script files with `#!` lines (only used on Unix systems).

By using UTF-8 (either through the signature or an encoding declaration), characters of most languages in the world can be used simultaneously in string literals and comments. Using non-ASCII characters in identifiers is not supported. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.