

LEARNING encoding

Free unaffiliated eBook created from **Stack Overflow contributors.**

#encoding

Table of Contents

About	1
Chapter 1: Getting started with encoding	2
Remarks	2
Examples	3
Installation or Setup	3
How to detect the encoding of a text file with Python?	3
Credits	4

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: encoding

It is an unofficial and free encoding ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official encoding.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with encoding

Remarks

What is an encoding and how it works?

A computer can't store letters or anything else - it stores bits. Bit can be either 0 or 1 ("yes"/"no", "true"/"false" - these formats are called binary therefore). To use these bits some rules are required, to convert the bits into some content. There rules are called *encodings*, where sequences of 1/0 bits stand for certain characters. A sequence of 8 bits is called *byte*.

Encodings work like tables, where each character is related to a specific byte. To encode something in ASCII encoding, one should follow the entries from right to left, searching for bits related to characters. To decode a string of bits into characters, one substitutes bits for letters from left to right.

Bytes can be represented in different formats: for example 10011111 in binary is 237 in octal, 159 in decimal and 9F in hexadecimal formats.

What is the difference between different encodings?

First character encoding like ASCII from the pre-8-bit era used only 7 bits from 8. ASCII was used to encode English language with all the 26 letters in lower und upper case form, numbers and plenty of punctuation signs. ASCII could not cover other European languages with all the ö-ß-é-å letters - so encodings were developed that used the 8-th bit of a byte to cover another 128 characters.

But one byte is not enough to represent languages with more than 256 characters - for example Chinese. Using two bytes (16 bits) enables encoding of 65,536 distinct values. Such encodings as BIG-5 separate a string of bits into blocks of 16 bits (2 bytes) to encode characters. Multi-byte encodings have the advantage to be space-efficient, but the downside that operations such as finding substrings, comparisons, etc. all have to decode the characters to unicode code points before such operations can be performed (there are some shortcuts, though).

Another type of encoding are such with variable number of bytes per character - such as UTF standards. These standards have some unit size, which for UTF-8 is 8 bits, for UTF-16 is 16 bits, and for UTF-32 is 32 bits. And then the standard defines some of the bits as flags: if they're set, then the next unit in a sequence of units is to be considered part of the same character. If they're not set, this unit represents only one character fully (for example English occupies only one byte, and thats why ASCII encoding maps fully to UTF-8).

What is the Unicode?

Unicode if a huge character set (saying in a more understandable way - a table) with 1,114,112 code points, each of them stands for specific letter, symbol or another character. Using Unicode, you can write a document which contains theoretically any language used by people.

Unicode is not an encoding - it is a set of code points. And there are several ways to encode Unicode code points into bits - such as UTF-8, -16 and -32.

Examples

Installation or Setup

Detailed instructions on getting encoding set up or installed.

How to detect the encoding of a text file with Python?

There is a useful package in Python - chardet, which helps to detect the encoding used in your file. Actually there is no program that can say with 100% confidence which encoding was used - that's why chardet gives the encoding with the highest probability the file was encoded with. Chardet can detect following encodings:

- ASCII, UTF-8, UTF-16 (2 variants), UTF-32 (4 variants)
- Big5, GB2312, EUC-TW, HZ-GB-2312, ISO-2022-CN (Traditional and Simplified Chinese)
- EUC-JP, SHIFT_JIS, CP932, ISO-2022-JP (Japanese)
- EUC-KR, ISO-2022-KR (Korean)
- KOI8-R, MacCyrillic, IBM855, IBM866, ISO-8859-5, windows-1251 (Cyrillic)
- ISO-8859-2, windows-1250 (Hungarian)
- ISO-8859-5, windows-1251 (Bulgarian)
- windows-1252 (English)
- ISO-8859-7, windows-1253 (Greek)
- ISO-8859-8, windows-1255 (Visual and Logical Hebrew)
- TIS-620 (Thai)

You can install chardet with a pip command:

```
pip install chardet
```

Afterward you can use chardet either in the command line:

```
% chardetect somefile someotherfile somefile: windows-1252 with confidence 0.5 someotherfile: ascii with confidence 1.0
```

or in python:

```
import chardet
rawdata = open(file, "r").read()
result = chardet.detect(rawdata)
charenc = result['encoding']
```

Read Getting started with encoding online: https://riptutorial.com/encoding/topic/6755/getting-started-with-encoding

Credits

S. No	Chapters	Contributors
1	Getting started with encoding	Community, vlad.rad