Text File Encoding

As explained by someone that never wanted to know about this mess

Text files are always easier to process (and other lies I've been told)

01

It’s just text

02

Why can’t I open this file?

03

Where did the rest of the file go?

04

What the heck is \040?

05

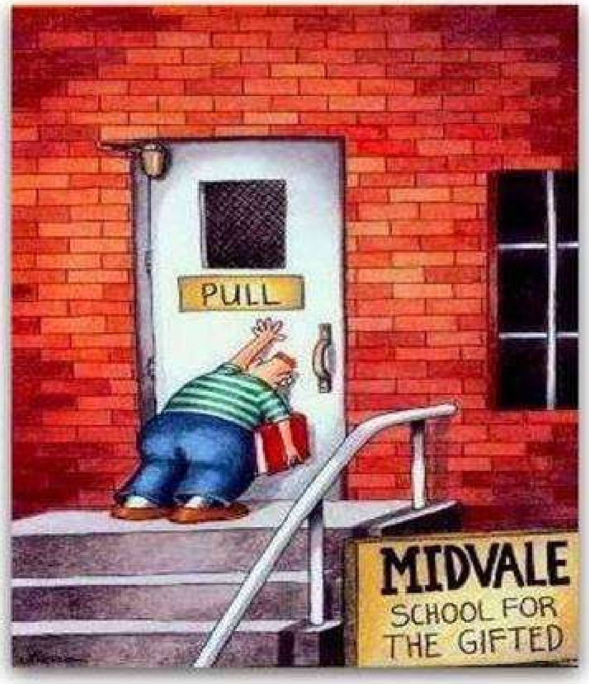
It looks fine on one computer, but not on this one

It’s just text

- Text files have no enforced rules
- Humans can and will:
  - Edit the original output. Raw source file is better than “helpful cleanup”
  - Break formatting
  - Accidentally change the encoding
  - Paste invalid data

Why can’t I open this file?

- The file header was damaged (humans can’t normally see this)
- My text-editor or application only understands ASCII or UTF-8
- My application incorrectly guessed the encoding type, and or it was tricked by some special characters near the top of the file.



Source: The Far Side by Gary Larson

Where did the rest of the file go?

- My application ran into some special characters, silently triggered an exception, and stopped processing the rest of the file.
  - Whenever processing files, do a sanity check. If the file is 10 MB, do you see about 10MB worth of data after importing?
  - Were you expecting 1,000 lines for 1,000 items, or was it 20,000 items?
- The process that exported this data to a text file was interrupted. I really don't have all the data that I requested.

What the heck is \040 ?

- Your current encoding scheme doesn't have a displayable character to represent that byte sequence, so instead you get an octal placeholder.
- If you only see this once or twice in a large dataset, it's probably safe to remove. It can be caused by cut and pasting text from another system into a non-sanitized input field.
- If you see a lot of these along with other gibberish:
  - You are opening the file using the wrong encoding type
  - The source database has some old data that survived a change to the database encoding type.

```
SOA server1.ad.treehealth.net. jim\040smith.example.org. (
                            2088          ; serial number
                            900           ; refresh
                            600           ; retry
                            86400         ; expire
                            3600        ) ; default TTL
```

```
5         PTR     lab1.example.com\040\040.
49   900  PTR     car\040prototype.abc.car.com.
```

It looks fine on one computer, but not on this one

- User 1 is using a different encoding type (and probably an old operating system)
  - The encoding type is probably old enough that it didn't identify itself in the file header.
- I'm using a modern OS, and it doesn't support that encoding type.
  - Ask for the user to save the file using UTF-8 encoding using a modern text editor
    - Window - Notepad++, MS Visual Studio
    - Mac - BBEdit (and Text Wrangler?), iconv
    - Linux - vim or iconv

Let's Break Stuff

- https://gitlab.com/scottwed/text-preflight
- breakout_text.py
- corrupt_text_file.py
- basic_scan.py