Encodings and Representing Text in a Computer: Takeaways

Syntax

- Encode a string:

```
'Data Quest'.encode(encoding='ascii')
```

- Change the way errors are handled:

```
'Data Quest'.encode(encoding='ascii', errors='replace')
```

- Creating a bytes object from hexadecimal:

```
b = bytes.fromhex("FF 00 A0 42")
```

- Accessing a specific byte of a bytes object:

```
b = bytes.fromhex("FF 00 A0 42")

print(b[0])
```

- Trying to detect the encoding of a string:

```
import chardet

chardet.detect(encoded)
```

- Decoding a string:

```
decoded = encoded.decode(encoding="UTF-8")
```

- Detecting and decoding a string:

```
import chardet

encoding = chardet.detect(encoded)

decoded = encoded.decode(encoding=encoding)
```

Concepts

- ASCII is the encoding upon which most encodings are build. It is able to represent a total of 128 characters.
- Some characters are meant to be read by a computer only and thus are not displayable on a screen. There are called **control characters** and used to convey information to the computer such that indicate the start and end of the text.
- Most encodings are compatible with ASCII meaning that ASCII characters are encoded in the same way in most encodings.
- All textual data is represented as a sequence of bytes in a computer. In order to read it we need to know which encodings was used. This is a hard task but `chardet.detect()` can help us with that.
- Unicode is an attempt to stop the encoding madness and have a single reference table for all existing symbols. There are several encodings that implement it, namely, UTF-8, UTF-16 and UTF-32.
- UTF-8 is the default encoding used in Python.

Resources

- ASCII
- ASCII table
- Unicode
- Unicode charts
- UTF-8