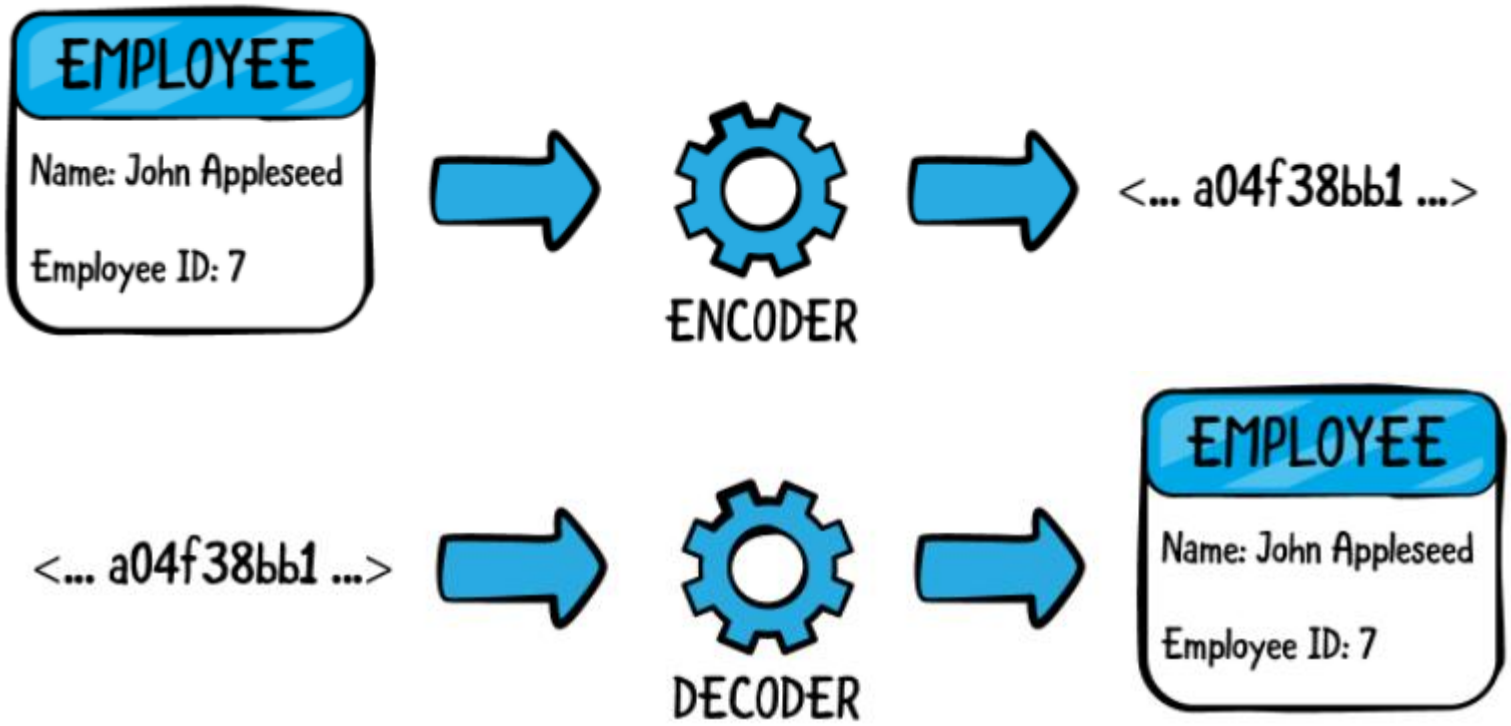
Character Encoding

Character Encoding

- Encoding
  - A way to convert data from one format to another.



[ref : https://www.raywenderlich.com/books/swift-apprentice/v6.0/chapters/22-encoding-decoding-types]

Character Encoding

- A character encoding
  - **A way to convert text data into binary numbers.**
  - Assigning unique numeric values to specific characters and converting those numbers in binary language.
  - Why?
    - To either **store** it inside a computer (machine) or **transfer** over a digital network

Character Encoding

- Terminologies
  - **Character Set**
    - A table of different characters like letters, numbers and other symbols.

D	0100
E	0101
F	0110
G	0111
H	1000
I	1001

Character Encoding

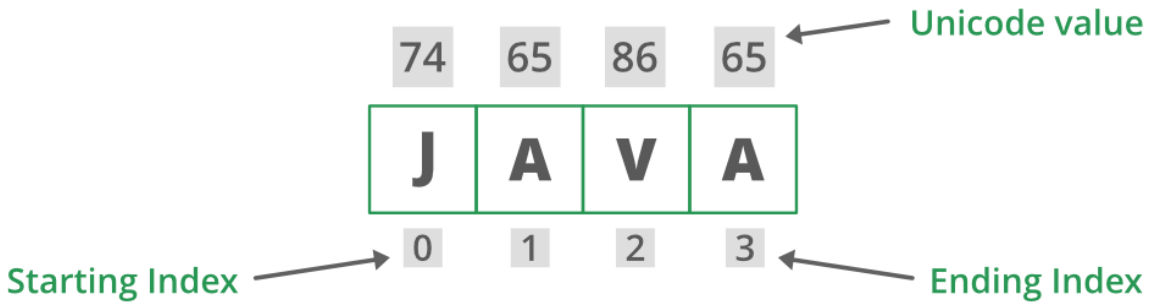
- Terminologies
  - **Encoding Scheme**
    - **A way to represent a character in binary.**
    - An encoding must follow a specific character set.
    - The value of character A in the UTF character set is decimal 65

character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010

[ref : https://stackoverflow.com/questions/2241348/what-is-unicode-utf-8-utf-16]

Character Encoding

- Terminologies
  - **Code Point**
    - **A decimal value associated with a character in a character set.**
      - **The atomic unit of information.**
    - E.g. the code point of character A in the UTF character set is 65.
    - **Text is a sequence of code points.**



[ref : https://www.geeksforgeeks.org/java-program-to-determine-the-unicode-code-point-at-given-index-in-string/]

Character Encoding

- ASCII Encoding
  - **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
  - An encoding and charset developed by USA in the **1960s**.
  - Mainly developed for **electronic communications** in the United States.
  - **Encoding English characters, numbers and other symbols** used **generally in the US only**
  - A total of **128 characters**
    - A unique value between 0 and 127.
  - **8 bit encoding** for computer storage
    - **The first bit 0 (MSB)**

Character Encoding

- ASCII Encoding
  - Examples
    - **0**1000001 → 41_{16} → 65_{10} → A
    - **0**1100001 → 61_{16} → 91_{10} → a
    - **0**0100000 → 20_{16} → 32_{10} → (*space*)
  - Pros and Cons
    - One of the simplest encodings schemes
    - Small text file size
    - Easier to read and write.
    - Applicable only to English language data.

Character Encoding

- Unicode Consortium and UTF encodings
  - **A universally accepted character set and encoding**
    - Applicable to every language
  - **The Unicode Consortium**
    - A non-profit organization that maintains the Unicode standard.
    - Unicode (an abbreviation for **Uni**versally **Code**d Character Set).
    - The Unicode Consortium also maintains the standard for UTF (**U**nicode **T**ransformation **F**ormat) encodings.
  - **Unicode characters are most commonly referred by their 4-digit hexadecimal representations (0000 to FFFF)**

Character Encoding

- Unicode Consortium and UTF encodings
  - Unicode
    - A coded character set
    - A set of characters and **a mapping between the characters and integer code points r**epresenting them
    - However, "Unicode" is unfortunately used in various different ways, depending on the context.
  - Both the **UCS standards and the UTF standards** encode the code points as defined in Unicode.
    - **These encodings were made to encode Unicode code points.**

Character Encoding

- Unicode Consortium and UTF encodings
  - **UCS Encodings**
    - Universal Coded Character Set
    - **16-bit and 32-bit fixed-width encoding** schemes to support characters from basic languages used across the world.
    - **UCS-2**
      - **Now obsolete**
    - **UCS-4**
      - **Identical to UTF-32**

Character Encoding

- UTF Encodings
  - **A variable-width encoding in the unit of byte**
  - Multiple encoding schemes, both fixed-width and variable-width.
    - **UTF-8, UTF-16, and UTF-32.**

Character Encoding

- UTF-8
  - **The encoding of the codepoints.**
    - one possible encoding scheme for **Unicode** text.
  - An **8-bit variable-length** encoding scheme **designed to be compatible with ASCII encoding.**
  - A variable length from **1 up to 4 bytes**
  - Using the UTF character set for character code points.
  - The main idea to **encode all the characters** that could possibly exist on the planet but at the same time **support ASCII encoding.**
    - **An ASCII encoded character will look exactly similar in UTF-8.**

Character Encoding

- UTF-8
  - The starting bits of the code unit for byte length

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

  - **The default encoding of a HTML document in HTML5**

Character Encoding

- UTF-8
  - Pros and Cons
    - **Compatibility with ASCII**
      - Any ASCII encoded document is a valid UTF-8 document.
    - **Memory efficient encoding**
    - Self-synchronizing.
      - **Easy to locate the start of encoding with random jump**
      - Critical to any good character encoding.
    - **The de facto standard for encoding in web and internet.**
      - Information about its encoding in Content-Type header
        - » Content-Type: <MIME Type>; charset=<encoding>.
        - » Content-Type: text/html; **charset=UTF-8**

Character Encoding

- UTF-16
  - **A 16-bit variable length encoding scheme**
    - Represented in **1 or 2 code units.**
    - **16 or 32 bits of memory** based on its code point.
  - **The initial 6 bits of the code unit for position** and length
    - Leaving only 10 bits to encode the code point of a character per code unit.
    - 20 bits of the memory for encoding for 2 code units
  - **The default character encoding scheme in Java and JavaScript**

Korean Character Encoding in Python

- Encoding
  - Converting character into integers
  - UTF-8 for Korean
    - **Encoding each character into 3 bytes**
  - How to find default encoding in python
    - Import sys
    - sys,stdin.encoding

```
import sys
sys.stdin.encoding
```

```
'UTF-8'
```

```
: '한국어'.encode('utf-8')
: b'\xed\x95\x9c\xea\xb5\xad\xec\x96\xb4'
```

Korean Character Encoding in Python [3]

- Encoding for Korean
  - Unicode
    - A standard coded character set to represent characters from almost all languages.
    - Every Unicode character is encoded using a unique integer code point between 0 and 0x10FFFF.
    - The range for Korean
      - **"AC00 ~ D7AF"**

Korean Character Encoding in Python

- Encoding for Korean
  - **2) UTF8 (Unicode Transformation Set – 8 bit)**
    - A variable-width character encoding used for electronic communi cation
    - Defined by the Unicode Standard
    - Basic unit of 8 bits
    - 3 byte for Korean while 1~4 byte depending on language
    - Unicode : "AC00 ~ D7AF"
    - **Representing byte length with prefix in blue color**
      - 0 → 1byte, 110 → 2byte, 1110 → 3byte, 11110 → 4byte
    - **Inserting unicode in x position**

Korean Character Encoding in Python

- Encoding for Korean
  - **2) UTF8 (Unicode Transformation Set – 8 bit)**

Unicode		UTF8		
U+0000~U+007F	0 - 127	0xxxxxxx	(1byte)	ASCII
U+0080~U+07FF	128 - 2047	110xxxxx 10xxxxxx	(2byte)	C280 - DFDF
U+0800~U+FFFF	2,048 - 65,535	1110xxxx 10xxxxxx 10xxxxxx	(3byte)	E0A080 - EFBFBF
U+10000~U+10FFFF	65,536 - 1,114,111	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	(4byte)	F0908080 - F090BD9F

- "안"
  - Unicode : "U+C548"
    - » C:1100/5:0101/4:0100/8:1000
  - 이며, UTF8 인코딩 값은 "0xEC9588" 입니다.

```
'안'.encode('utf8')
```

```
b'₩xec₩x95₩x88'
```

Binary 11101100 10010101 10001000
hexa e c 9 5 8 8

Korean Character Encoding in Python

- Encoding for Korean

```
a='자연어'.encode('UTF-8')
a
```

```
b'₩xec₩x9e₩x90₩xec₩x97₩xb0₩xec₩x96₩xb4'
```

```
print('₩xec₩x9e₩x90₩xec₩x97₩xb0₩xec₩x96₩xb4')
print(b'₩xec₩x9e₩x90₩xec₩x97₩xb0₩xec₩x96₩xb4')
print(a)
print(a.decode('UTF-8'))
print(b'₩xec₩x9e₩x90'.decode('UTF-8'))
print(b'₩xec₩x97₩xb0'.decode('UTF-8'))
print(b'₩xec₩x96₩xb4'.decode('UTF-8'))
```

```
ìì—° ì- ´
b'₩xec₩x9e₩x90₩xec₩x97₩xb0₩xec₩x96₩xb4'
b'₩xec₩x9e₩x90₩xec₩x97₩xb0₩xec₩x96₩xb4'
자연어
자
연
어
```

References

- [1] https://guzene.tistory.com/150
- [2] https://bytes.com/topic/python/answers/701188-reload-sys
- [3] https://redscreen.tistory.com/m/163
- [4] https://financedata.github.io/posts/faq_crawling_data_encoding.html
- [5] https://financedata.github.io/posts/faq_crawling_data_encoding.html
- [6] https://medium.com/jspoint/introduction-to-character-encoding-3b9735f265a6