4-2. 텍스트 데이터를 문자열로 저장한다는 것 (1) 인코딩과 디코딩

텍스트 데이터를 문자열로 저장한다는 것

우리 주변에는 수많은 텍스트 데이터가 있습니다.

텍스트 데이터는 과연 어떻게 나타낼까요? 바로 **문자열**(string)로 표현합니다. 파이썬에서 문자열 리터럴(literal)은 작은따옴표 `('')` 혹은 큰따옴표 `("")` 로 묶인 일련의 문자라고 볼 수 있습니다.

자, 일단 한번 텍스트 데이터를 변수에 저장해 출력해볼까요?

[Input]

```
my_str = 'Welcome!'
ur_str = "You're welcome."

print(my_str)
print(ur_str)

#- 아래 주석을 제거해 각 변수의 자료형을 확인해보세요 :)
type(my_str)
#type(ur_str)
```

실행

[Output]

우리는 지금까지 데이터를 변수에 저장했습니다. 리스트를 사용해 배열로 저장하기도 하고, 따옴표를 이용해 문자열 형태로 저장해보기도 했습니다.

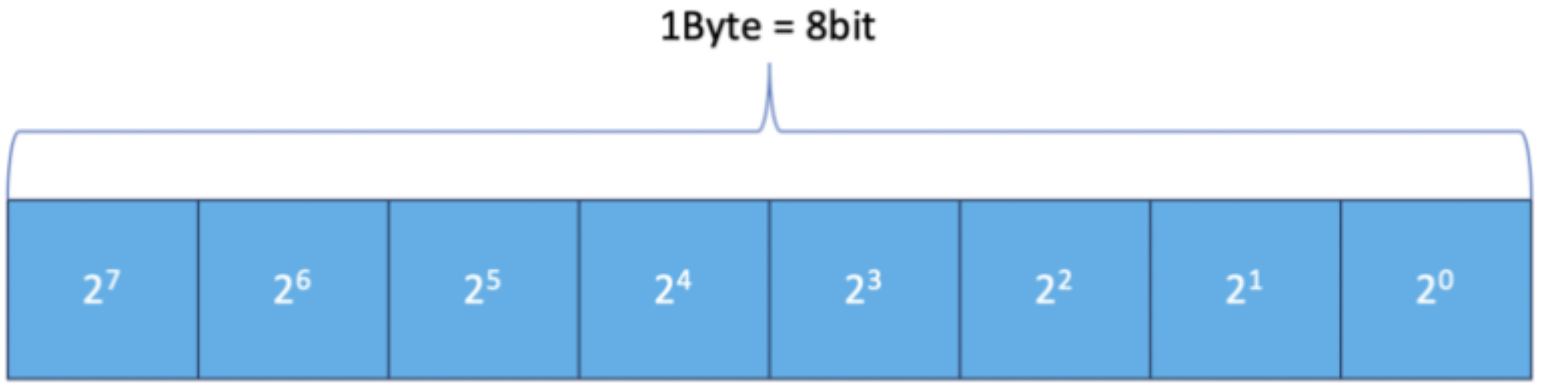
이렇게 변수에 데이터를 할당하면 이 데이터들은 컴퓨터의 주기억장치인 메모리(RAM)에 저장됩니다.

인코딩과 디코딩

이 문자열 데이터가 메모리에 저장될 때 컴퓨터 내부에서는 어떤 일이 일어날까요?

컴퓨터 세상은 온통 숫자 0과 1 즉, 이진 데이터(binary data)로 표현되기 때문에 데이터도 마찬가지로 0과 1로 변환돼 다뤄지게 됩니다.

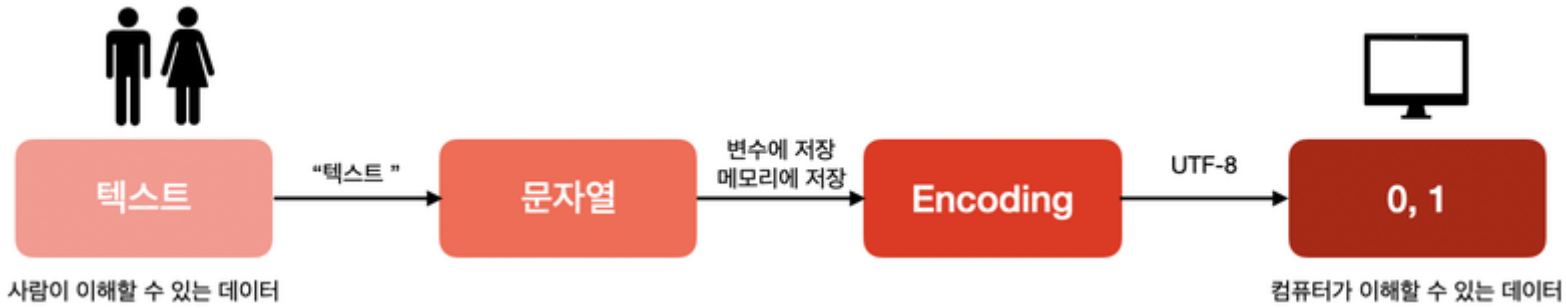
이진 데이터의 최소 단위는 비트(bit)이고, 비트가 8개 모이면 바이트(byte)가 되는데요. 메모리에는 바이트로 저장이 됩니다.



✔ 잠깐, 정리해보아요!

- **바이트**(byte) : 컴퓨터의 기본 저장단위
  - 1바이트(1byte)는 8비트(8bit)이다.
  - 1바이트에는 2의 8승, 즉 256개의 고유한 값을 저장할 수 있다.
- **인코딩** (encoding): 문자열을 바이트로 변환하는 과정
- **디코딩** (decoding) : 바이트를 문자열로 변환하는 과정

텍스트 데이터의 처리 과정



위 그림에서 볼 수 있듯이 사람이 이해할 수 있는 데이터는 텍스트 데이터, 엄밀히 말해 **문자열 데이터**를 의미합니다. 그렇다면 어떻게 문자열을 이진수로 변환(인코딩)할까요? 숫자를 이진수로 표현하는 것은 생각보다 간단합니다. 문자 각각을 숫자에 대응시키고 이 숫자를 이진수로 변환하면 됩니다. 예를 들어 숫자 5를 이진수로 나타내면 101이 되겠죠.

여기서 잠깐 생각해봅시다. 같은 알파벳 a를 미국은 숫자 65로, 한국은 숫자 80으로 표현한다면 어떨까요? 더 나아가 한자를 인코딩하는 규약이 명시되지 않았다면 한자로 쓰인 파일은 다 깨져버릴 것입니다. 문자를 숫자로 대응시키는 방법이 이렇게 제각각이면 텍스트로 정보를 주고받는 과정에서 엄청난 혼란이 있을 것이라 직감할 수 있을 것입니다.

하지만 이 방법을 전 세계가 통일시킨다면 텍스트 데이터를 오류없이 깔끔하게 처리할 수 있겠죠. 그래서 국제표준기구인 ISO(International Standards Organization)는 전세계 문자를 모두 표시할 수 있는 표준코드를 제정하였습니다. 이것이 바로 유니코드(Unicode)입니다.

그렇다면 UTF-8, UTF-16 등은 무엇일까요? 흔히 UTF-8과 유니코드가 같은 것이라고 혼동하는 경우가 많아서, 여러가지 버전의 유니코드가 존재하는 것이 아닌가 오해할 수 있어서 정리합니다.
유니코드는 오직 한가지 버전만 존재합니다. 그리고 UTF-8, UTF-16 등은 유니코드로 정의된 텍스트를 메모리에 인코딩하는 방식들을 말합니다.

좀 더 개념을 정확히 이해하고 싶으시다면 아래 글들을 참고해보세요.

- 유니코드 테이블
- 유니코드와 UTF-8
- UTF-8과 UTF-16

파이썬 내장함수 ord()와 chr()

- `ord()` : 해당 문자에 대응하는 유니코드 숫자를 반환합니다.
- `chr()` : 해당 유니코드 숫자에 대응하는 문자를 반환합니다.

Q1. 유니코드 테이블에서 한글 '가'에 해당하는 코드값은 무엇인가요?

U+AC00

제출

예시답안

U+AC00

[Input]

```
print(ord('a'))
print(ord('A'))
print(chr(97))
print(ord('가'))
print(chr(0xAC00))
print(chr(ord('가')))
#- 0xAC00은 44032의 16진수 표현입니다.
```

실행 ▶

[Output]

파이썬에서 모든 문자열은 유니코드로 표현된다!

문자열은 내부적으로 -엄밀히 말하면 파이썬 3부터- 유니코드 규약에 따릅니다. 이는 외부 데이터 및 데이터베이스로부터 데이터를 읽거나 보낼 때는 인코딩 혹은 디코딩 작업을 거쳐야 하며, 인코딩 규약은 내부적으로 유니코드(UTF-8)임을 의미합니다. 파이썬 2에서 문자열은 인코딩 및 디코딩 여부와 상관없이 (regular) string, unicode string으로 구분되었고, 파이썬 3부터는 인코딩 혹은 디코딩이 되어 있는지로 구분되었습니다. 아래 코드 및 실행 결과를 살펴보시면 이 차이를 금방 이해하실 수 있을 것입니다.

```
#- 파이썬 2 -#
#- string, unicode string으로 구분됩니다.

>>> str1 = b'hello'
>>> str2 = 'hello'
>>> str3 = u'hello'
>>> print(type(str1), type(str2), type(str3))
<type 'str'>, <type 'str'>, <type 'unicode'>
```

```
#- 파이썬 3-#
#- bytes와 string으로 구분됩니다.

>>> str1 = b'hello'
>>> str2 = 'hello'
>>> str3 = u'hello'
>>> print(type(str1), type(str2), type(str3))
<type 'bytes'>, <type 'str'>, <type 'str'>
```

간단히 정리를 해보자면,

- 파이썬 2에서는 인코딩을 한 후에도 아스키(ascii) → 유니코드(unicode) 변환 등의 작업을 거쳐야 했던 반면에,
- 파이썬 3부터는 문자열이 무조건 유니코드로 인코딩되므로 해당 텍스트가 인코딩이 되어 있는지 혹은 디코딩이 되어 있는지만 고려하면 된다는 것이 포인트입니다.