파이썬 프로그래밍
강의 노트 #05

문자열

문자열 조작의 필요성

- 프로그램을 만들다 보면 문자열을 자르고, 붙이고, 검색하고, 치환하는 등의 코드를 자주 작성
- 예: 증권 분석 프로그램에서 회사명, 종목명 등 문자열이 많이 쓰임
- 학교 수강신청 프로그램에서도 과목 이름, 강사 이름, 학생 이름 등의 문자열이 있음
- 인터넷에서 날씨 정보를 읽고 처리하는 것도 문자열 처리가 필요함

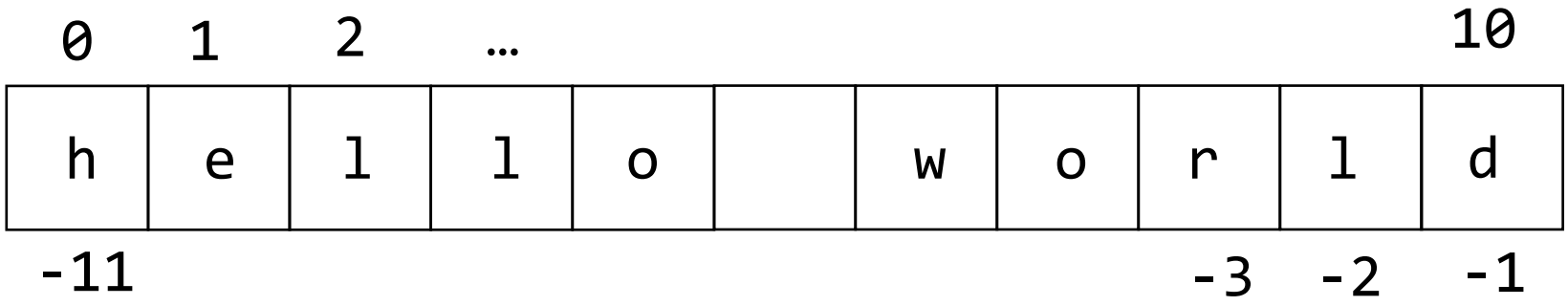
문자열 다루기

- 문자열의 길이 확인
- 문자열의 길이
  - 영문과 한글을 구별하지 않고 문자열에 포함된 글자 개수
  - len() 명령 사용

```
>>> len("hello")
5
>>> s = "문자열의 길이를 확인해요."
>>> len(s)
14
```

문자열의 특정 위치 확인

- 문자열의 특정 문자 확인하기
  - 문자열[인덱스] 사용
    - 문자열의 첫 글자의 인덱스는 0부터 시작
    - 인덱스가 음수이면 문자열의 끝에서부터 시작됨



문자열의 특정 위치 확인

```
>>> s = "Hi, everyone.\n"
>>> s[0] # 'H'
'H'
>>> s[-1] # '\n'
'\n'
>>> s[len(s) - 1]
'\n'
>>> print(s[-2]) # '.'
'.'
>>> print(s[10]) # 'n'
'n'
```

문자열은 수정 불가

- 파이썬의 문자열은 수정할 수 없음
- 문자열의 내용을 변경하려 하면 오류 발생

```
>>> s = "Hi, everyone.\n"
>>> s[0] = 'h'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item
assignment
```

문자열에 포함되어 있는 명령들

- 문자열은 다양한 명령들을 포함하고 있음
- 문자열 명령 사용법
  - 문자열.명령()
- 문자열 검색
  - 문자열이 특정 문자열로 시작하는지 확인
    - 문자열에 포함되어 있는 startswith() 활용

```
s = "Hello World"
s.startswith('H') # s는 H로 시작하므로 True
s.startswith('W') # s는 W로 시작하지 않으므로 False
s.startswith('h') # 결과는 ???
s[0] == 'H'
s[0] == 'h'
```

문자열 검색

- 문자열이 특정 문자열로 끝나는지 확인
  - endswith() 함수 활용

```
s = "Hello World"
s.endswith('d') # s는 H로 시작하므로 True
s.endswith('D') # s는 W로 시작하지 않으므로 False
s[len(s) - 1] == 'd'
s[-1] == 'd'
s[len(s) - 1] == 'D'
```

문자열 검색

- 문자열에 특정 문자열이 포함되었는지 확인
  - in 활용
    - in은 문자열1이 문자열2에 포함되어 있는지 확인

```
문자열1 in 문자열2 # True 또는 False 반환
```

    - 예

```
s = "Hello World"
"Wor" in s   # True
"wor" in s   # False
```

문자열 검색

- index() 명령 활용
  - in은 단순히 문자열이 있는 지만 알려줌
  - index() 명령은 어디에 위치하고 있는지를 알려줌
  - 단 문자열이 없으면 오류 발생

```
s = "Hello World"
s.index("Wor") # 6
s.index("wor") # error 발생
```

- find(str)
  - str로 주어진 문자열을 찾아서 인덱스 반환
  - 만약 주어진 문자열이 없으면 -1 반환

```
s.find("Wor")  # 6
s.find("wor")  # -1
```

문자열 검색

- index()와 find()에서 문자열의 검색할 구간을 지정

```
문자열.index(검색할_문자열, 시작_인덱스, 끝_인덱스)
문자열.find(검색할_문자열, 시작_인덱스, 끝_인덱스)
```

- 예

```
>>> s = "Hello World"
>>> s.index('o', 0, 4)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: substring not found
>>> s.index('o', 0, 5)
4
>>> s.find('o', 0, 4)
-1
>>> s.find('o', 0, 5)
4
```

문자열 검색

- rindex()와 rfind() 활용
  - 검색하는 문자열이 가장 마지막에 나타나는 위치를 반환(오른쪽에서부터 검색)

```
>>> "Bye Bye".rindex("Bye")
4
>>> "Bye Bye".rfind("Bye")
4
```

  - 검색하는 문자열이 없으면 rindex()는 오류 발생
  - 검색하는 문자열이 없으면 rfind()는 -1 반환
  - rindex()와 rfind()에서도 검색할 구간 지정 가능

문자열 추출하기

- 문자열 일부 추출하기
  - s[시작_인덱스:마지막_인덱스]
  - 인덱스 시작_인덱스부터 마지막_인덱스-1까지 추출해서 새로운 문자열을 생성함
  - 인덱스가 비어 있으면 양 끝 중 한 쪽을 의미함

```
s = "Hello World"
s[0:2]      # He
s[1:4]      # ell
s[:-1]      # 처음부터 -2까지. 즉 끝에서 한
            # 문자 빼고 추출
s[:3]       # 처음부터 3-1까지. Hel
s[:]        # 처음부터 끝까지
```

문자열 다루기

- 문자 제거
  - strip()명령
    - 문자열의 양 끝에 있는 공백 문자를 제거

```
" hello \n".strip() # "hello"만 남음
```

  - strip("문자조합")
    - 문자열의 양 끝에 있는 "문자조합"의 모든 문자들을 제거

```
"가hello가".strip('가') # "hello"만 남음
"가hello".strip('가') # "hello"만 남음
"hhelloh".strip('h') # "ello"만 남음
"가helloh가나".strip("가h나") # "ello"만 남음
```

문자열 다루기

- 문자 제거
  - lstrip()명령
    - 문자열의 왼쪽 끝에 있는 공백 문자를 제거

```
" hello \n".lstrip() # "hello \n"가 남음
```

  - lstrip("문자조합")
    - 문자열의 왼쪽 끝에 있는 "문자조합"의 모든 문자들을 제거

```
"가hello가".lstrip('가')
"가hello".lstrip('가')
"hhelloh".lstrip('가')
"가helloh가나".lstrip("가h나")
```

문자열 다루기

- 문자 제거
  - rstrip()함수
    - 문자열의 오른쪽 끝에 있는 공백 문자를 제거

```
" hello \n".rstrip() # " hello"가 남음
```

  - rstrip("문자조합")
    - 문자열의 오른쪽 끝에 있는 "문자조합"의 모든 문자들을 제거

```
"가hello가".rstrip('가')
"가hello".rstrip('가')
"hhelloh".rstrip('가')
"가helloh가나".rstrip("가h나")
```

문자열 일부를 치환해서 새로운 문자열 생성

- replace(oldString, newString) 명령은 문자열에서 oldString을 newString으로 치환해서 새로운 문자열을 생성

```
>>> "뜨거운 커피".replace("뜨거운", "차가운")
'차가운 커피'
```

- 치환할 문자열이 없으면 아무것도 안함

```
>>> "차가운 커피".replace("뜨거운", "차가운")
'차가운 커피'
```

문자열 다루기

- 글자 또는 숫자 확인하기
  - isalnum()
    - 문자열이 알파벳과 숫자로만 구성되어 있는지 확인
  - isalpha()
    - 문자열이 알파벳만으로 구성되어 있는지 확인
  - isdigit(), isnumeric()
    - 문자열의 글자가 숫자로만 구성되어 있는지 확인
  - islower()
    - 문자열이 영문 소문자로만 구성되어 있는지 확인
  - isupper()
    - 문자열이 영문 대문자로만 구성되어 있는지 확인

문자열 다루기

- upper()
  - 문자열을 대문자로 변환된 새로운 문자열 반환
- lower()
  - 문자열을 소문자로 변환된 새로운 문자열 반환
- swapcase()
  - 문자열의 대문자는 소문자로 변환, 소문자는 대문자로 변환된 새로운 문자열 반환
- count(str)
  - 문자열에서 주어진 문자열인 str이 몇 번 나오는지 반환
- count(str, start, end)
  - 문자열에서 주어진 문자열인 str이 몇 번 나오는지 찾는데 start 부터 end – 1 인덱스내에서만 검색

실습문제 1

- 문제
  - "Beautiful.Image.png"파일 이름에서 확장자를 제외한 부분을 화면에 출력
- 요구사항
  - 확장자는 파일 이름의 마지막에 ".확장자" 형태로 주어진다고 가정
  - 파일 이름에 "."가 두 개 이상 있을 수 있음(최소 한 개는 있음)
  - 확장자는 몇 글자인지 정해져 있지 않음

실습문제 2

- 문제
  - 사용자로부터 파일 이름을 입력받고, 확장자를 제외한 부분을 화면에 출력
- 요구사항
  - 확장자는 파일 이름의 마지막에 ".확장자" 형태로 주어진다고 가정
  - 파일 이름에 "."가 없을 수 있음. 확장자가 없다면, 입력된 파일 이름 전체를 출력
  - 파일 이름에 "."가 두 개 이상 있을 수 있음
  - 확장자의 글자 수는 정해지지 않음

실습문제 3

- 문제
  - 사용자로부터 두 개 이상의 단어가 있는 문자열을 입력받고, 첫 번째 단어만 추출해서 출력
- 요구사항
  - 사용자가 반드시 두 개 이상의 단어를 입력한다고 가정
  - 단어는 공백 문자로 분리된다(탭('₩t') 문자나 줄바꿈('₩n') 문자는 검사하지 않음)
  - 문자열의 양 끝에 공백 문자가 있을 수 있음(먼저 제거한 후 단어 추출)

실습문제 4

- 문제
  - 사용자로부터 파일 이름을 입력 받고, 파일 이름의 확장자가 ".png"로 끝나면 ".jpg"로 교체한 후 화면에 출력. 확장자가 ".png"가 아니라면, 파일 이름을 있는 그대로 출력
- 요구사항
  - 파일 이름이 ".png"로 끝나는지 확인
  - replace() 함수를 사용하지 않고 다른 명령들을 이용해서 문제를 해결할 것

문자 인코딩과 디코딩

- 컴퓨터는 디지털 시스템
  - 0과 1의 조합만 이해할 수 있음
- 영어 알파벳 'A'는 영어를 이해하는 사람이 보면 알파벳 중 한 개. 만약 모르는 사람이 본다면 글자 또는 기호에 불과함
- 컴퓨터는 사람이 사용하는 언어를 이해하지 못함. 따라서 'A'라는 글자는 처리할 자료의 한 개이지 의미를 모름
- 컴퓨터는 다음 코드를 어떻게 실행할까?

```
ch = 'A'
```

문자 인코딩과 디코딩

- 메모리에 'A'를 저장해야 하는데, 메모리는 0과 1의 조합만 저장 가능
- 'A'를 0과 1의 조합으로 변환해야 함
  - 문자 변환 표가 필요함
- 문자 변환 표의 예

문자	이진 코드	문자	이진 코드
A	000	B	001
C	010	D	011
가	100	나	101
다	110	라	111

문자 인코딩과 디코딩

- 문자 인코딩(encoding)
  - 문자를 이진 코드로 변환
- 문자 디코딩(decoding)
  - 이진 코드를 문자로 변환

대표적인 문자 변환 표

- ASCII(아스키 코드), EUC-KR, cp949(CP949, MS949), Unicode 등이 있음
- ASCII
  - 1960년대 초반 미국에서 표준화된 문자 변환 표
  - 7비트로 구성되어 128개 글자 표현 가능
  - C/C++, Python 2에서 기본으로 사용

16진수 코드	문자	16진수 코드	문자	16진수 코드	문자	16진수 코드	문자
58	X	59	Y	5A	Z	5B	[
5C	₩	5D	]	5E	^	5F	_
78	x	79	y	7A	z	7B	{

대표적인 문자 변환 표

- EUC-KR
  - 한국에서 표준화시킨 한글 문자 변환 표
  - ASCII 코드를 포함하고 한글 등을 2바이트로 표현

16진수 코드	문자	16진수 코드	문자	16진수 코드	문자	16진수 코드	문자
B0A1	가	B0A2	각	B1A3	간	B1A4	갇
B1A5	갉	B1A6	갊	B1A7	감	B1A8	갑
B2A1	깹	B2A2	깻	B2A3	깼	B2A4	깽

  - 한글을 초성, 중성, 종성으로 조합하는 것이 아니라, 이미 완성되어 있는 글자 형태로 코드화시킴
  - 현대 한글에서 자주 쓰이는 2350글자만 표현 가능
  - 1990년에 MBC의 인기 드라마였던 "똠방각하"를 EUC-KR에서 표현 못함

대표적인 문자 변환 표

- cp949
  - 마이크로소프트사에서 윈도우에서 사용하기 위해 만든 한글 코드 표
  - 초성, 중성, 종성으로 조합할 수 있는 모든 한글을 표에 넣음 → 현대 한글에서 사용할 수 있는 모든 한글을 완성형으로 넣어 확장 완성형이라고도 불림
  - 윈도우에서는 ANSI 인코딩이라고 불리기도 함

대표적인 문자 변환 표

- 유니코드
  - 마이크로소프트사를 비롯한 다국적 소프트웨어 기업들이 전 세계에서 사용하는 대부분의 언어들을 통합하는 문자 변환표를 만든 것
  - 여러 가지 인코딩 방법들이 제공됨
  - 리눅스나 맥 운영체제, 파이썬 3은 UTF-8(utf-8)을 사용함
- utf-8인코딩 방식
  - ASCII 코드에 포함된 128개 문제는 동일함
  - 나머지 문자들은 1~3바이트로 표현

파이썬 코드로 인코딩 다뤄보기

- 문자열을 바이트열(8비트 단위의 숫자 열)로 변환
  - 문자를 이진화된 코드로 구성된 문자열로 바꾸는 것
  - 문자열의 encode() 명령 사용

```
>>> "test".encode()
b'test'
```

  - 출력 결과의 'b'는 이진화된 코드임을 보임
- 한글 "파이썬"을 이진 코드로 변환

```
>>> "파이썬".encode()
b'\xed\x8c\x8c\xec\x9d\xb4\xec\x8d\xac'
```

  - utf-8에서 한글 한 글자는 3바이트로 저장
  - 출력 결과의 ₩x는 16진수임을 보임

파이썬 코드로 인코딩 다뤄보기

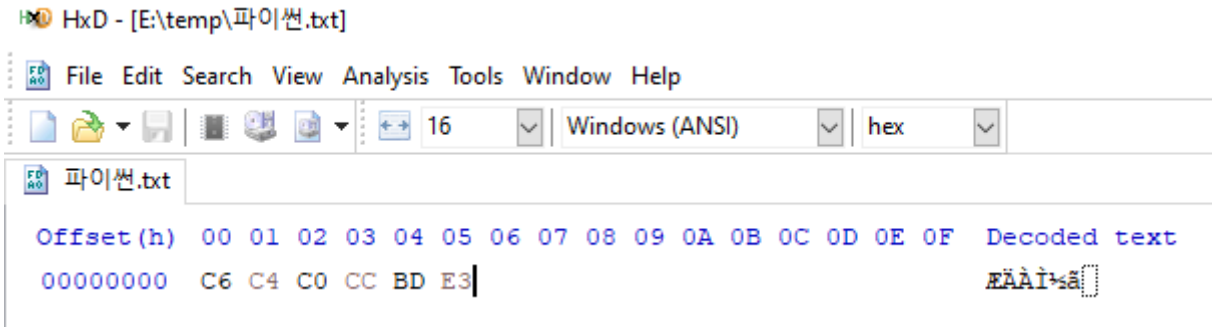
- 다른 인코딩 방식으로 변환하려면 encode() 명령에 변환 방법을 지정
- "파이썬"을 utf-16방식으로 변환

```
>>> "파이썬".encode("utf-16")
b'\xff\xfe\x0c\xd3t\xc7l\xc3'
```

- cp949방식으로 변환

```
>>> "파이썬".encode("cp949")
b'\xc6\xc4\xc0\xcc\xbd\xe3'
```

cp949방식으로
파일에 저장한 내용



파이썬 코드로 인코딩 다뤄보기

- URL (퍼센트) 인코딩이란?

- URL 인코딩은 퍼센트 인코딩이라고도 불리며 URL에 문자를 표현하는 문자 인코딩 방법입니다. 알파벳이나 숫자 등 몇몇 문자를 제외한 나머지는 1바이트 단위로 묶인 16진수로 인코딩하는 방식

- GET 방식을 통해 HTTP 요청을 할 때 쿼리 파라미터가 붙는 경우가 생기는데 URL은 ASCII 코드값만 사용됩니다. 이 쿼리 파라미터에 한글이 포함될 경우, ASCII 코드만으로 표현을 할 수 없어서 인코딩을 진행해야 합니다. 호출하는 API마다 쿼리 파라미터에 한글 문자 그대로를 지원하는 경우도 있지만 그렇지 않은 경우도 있으므로 미리 인코딩을 거친 형식으로 전송하는 것이 바람직하다
- URLLib() 패키지 활용으로 적용한다.

실습문제5

- 문제
- 사용자로부터 문자열을 입력받아 ,
  그 문자열을 역순으로 출력하는 코드를 작성하세요.

- 요구사항

실습문제6

- 문제

- 사용자로부터 문자열을 입력받아 그 문자열을 모두 대문자로 변환하여 출력하는 코드를 작성하세요.

- 요구사항

실습문제7

- 문제
- 사용자로부터 문자열과 제거할 문자를 입력받아, 해당 문자를 모두 제거한 문자열을 출력하는 코드를 작성하세요.

- 요구사항

실습문제8

- 문제
- 사용자로부터 문자열을 입력받아, 문자열의 앞뒤 공백을 제거한 후 출력하는 코드를 작성하세요.

- 요구사항

실습문제9

- 문제
- 사용자로부터 문자열을 입력받아, 해당 문자열에 포함된 단어의 개수를 세어 출력하는 코드를 작성하세요.

- 요구사항

실습문제10

- 문제
- 사용자로부터 문자열과 특정 단어를 입력받아, 해당 문자열에 그 단어가 몇 번 포함되어 있는지 세어 출력하는 코드를 작성하세요.

- 요구사항

실습문제11

- 문제
- 사용자로부터 문자열을 입력받아, 대문자는 소문자로, 소문자는 대문자로 변환하여 출력하는 코드를 작성하세요.

- 요구사항

실습문제12

- 문제
- 사용자로부터 문자열을 입력받아, 문자열의 첫 글자만 대문자로 변환하여 출력하는 코드를 작성하세요.

- 요구사항

실습문제13

- 문제
- 사용자로부터 문자열을 입력받아, 문자열의 각 단어의 첫 글자만 대문자로 변환하여 출력하는 코드를 작성하세요.

- 요구사항

실습문제14

- 문제

- 사용자로부터 문자열을 입력받아, 해당 문자열을 UTF-8로 인코딩하여 바이트 배열을 출력하는 코드를 작성하세요.

- 요구사항
  - encode('utf-8') 활용

실습문제15

- 문제

- 사용자로부터 UTF-8 인코딩된 바이트 배열을 입력받아, 해당 바이트 배열을 문자열로 디코딩하여 출력하는 코드를 작성하세요.

- 요구사항
  - eval(b'') 평가식 활용

실습문제16

- 문제
- 사용자로부터 문자열을 입력받아, 해당 문자열을 URL 인코딩하여 출력하는 코드를 작성하세요.

- 요구사항
  - import urllib.parse 활용
  - urllib.parse.quote('') 활용

실습문제17

- 문제
- 사용자로부터 URL 인코딩된 문자열을 입력받아, 해당 문자열을 디코딩하여 원래의 문자열로 출력하는 코드를 작성하세요.

- 요구사항
  - urllib.parse.unquote() 활용

실습문제18

- 문제
- 사용자로부터 문자열을 입력받아, 해당 문자열을 헥스(hex)로 인코딩하여 출력하는 코드를 작성하세요.

- 요구사항
  - encode().hex()

실습문제19

- 문제
- 사용자로부터 헥스(hex)로 인코딩된 문자열을 입력받아, 해당 문자열을 디코딩하여 원래의 문자열로 출력하는 코드를 작성하세요.

- 요구사항
  - bytes.fromhex('').decode() 활용

Geek & geek