파이썬 프로그래밍

8차시

문자열 다루기

⚠ 학습개요

… 문자열

- 객체와 클래스
- 문자열 길이, 첨자 활용(indexing)
- 슬라이싱(slicing)

… 문자 함수

- chr() ord()

⚠ 학습개요

… 문자 표현

- 이스케이프 시퀀스

… 내장 함수

- min() max()

⚠ 학습목표

… 문자열에서 길이를 알 수 있고 첨자로 하나의 문자를 참조할 수 있다.

… 부분 문자열을 참조할 수 있다.

… 문자의 코드 번호를, 코드로 문자를 알 수 있다.

… 내장 함수 min() max()를 활용할 수 있다.

Chapter 1.

문자열

PYTHON PROGRAMMING

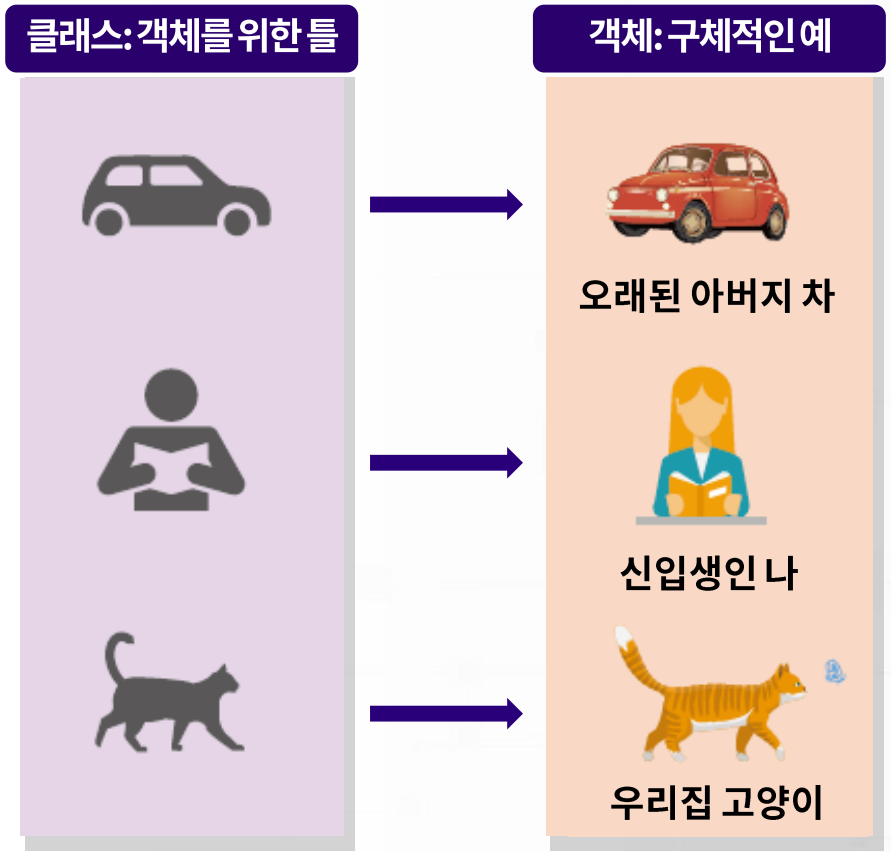
⚠ 클래스와 객체

+ 문자열: '문자의 나열', 텍스트 시퀀스(text sequence)

- 자료형 **class str**
- 'python'과 같은 문자열 상수는 **클래스 str의 객체**

용어를 알아봅시다!

+ 클래스와 객체



[그림8-1] 클래스와 객체

- 일상생활에서 사용하는 입학생, 자동차, 고양이 등의 용어 자체를 클래스(class)라고 생각하자. 여기서 구체적으로 오래된 아버지 차, 신입생인 나, 우리집 고양이 등과 같은 구체적인 사례는 객체다.
  즉, 클래스 str은 문자열의 개념을 정리해놓은 자료형이고,
  여기서 'python', 'java' 등은 클래스 str의 구체적인 객체다.

- 클래스는 객체지향 프로그래밍 언어의 관점에서 객체를 정의하기 위한 틀 또는 모형인 템플릿(templet)이라고 한다.
  객체는 클래스로부터 만들어진 하나의 구체적인 사례인 인스턴스(instance)다. 다시 말해 클래스는 객체를 만들기 위한 정의, 객체는 해당 클래스의 구체적인 자료라고 생각하자.

⚠ 문자열의 다양한 표현

+ 문자열: '문자의 나열', 텍스트 시퀀스(text sequence)

1 작은따옴표

- "'큰" 따옴표 표현'처럼 문자열 내부에 큰따옴표 사용 가능

2 큰따옴표

- "'작은' 따옴표 표현"처럼 문자열 내부에 작은따옴표 사용 가능

3 삼중따옴표

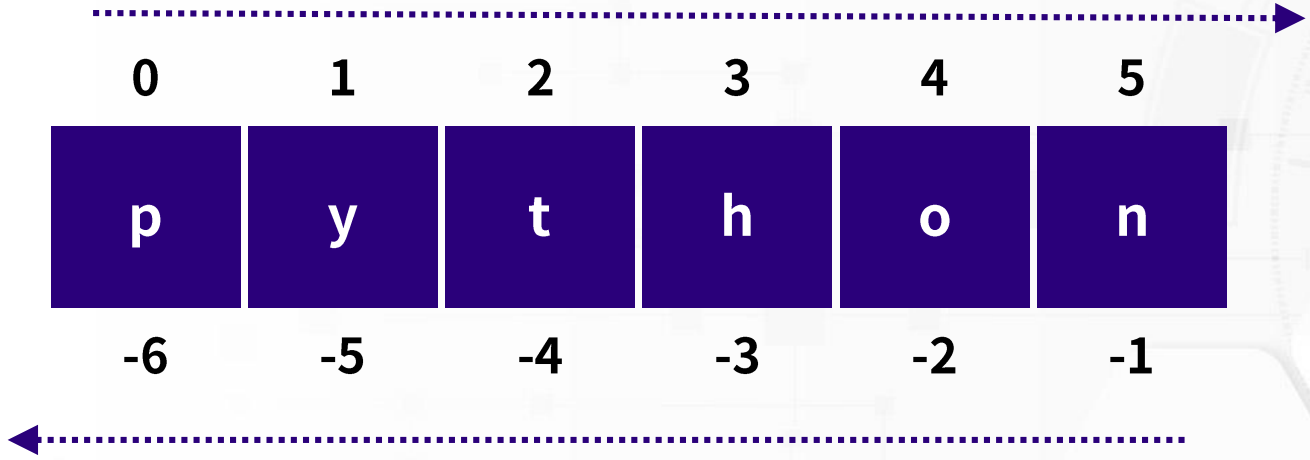
- "'문자열''' 또는 """문자열"""로 여러 줄의 문자열 표현

⚠ 문자열 활용, 길이와 문자 참조

+ 문자열의 길이 반환: 함수 len()

+ 첨자를 사용한 문자열의 문자 참조

오름차순 첨자: 0~[len('python')-1]



내림차순 첨자: [-len('python')~-1]

⚠ 인덱싱(indexing): 문자열 첨자 활용

```
>>> 'python'[0]
'p'
>>> 'python'[3]
'h'
>>> 'python'[-1]
'n'
>>> 'python'[len('python')-1]
'n'
>>> 'python'[-4]
't'
```

```
>>> 'python'[6]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> 'python'[-7]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

⚠ 문자열 활용, 길이와 문자 참조

[코딩실습] 첨자로 문자열의 문자 참조

난이도 기본

```
1. str = 'Hello python!'
2. n = len(str)
3. print('문자열', str, '길이', n)
4. print('첫 문자', str[0], str[-n])
5. print('가운데 문자', str[n//2], str[-n//2])
6. print('마지막 문자', str[n-1], str[-1])
```

결과

```
문자열 Hello python! 길이 13
첫 문자 H H
가운데 문자 p p
마지막 문자 ! !
```

⚠ 슬라이싱: 문자열의 부분 문자열 참조 방식

+ 슬라이싱(slicing)

- **전체에서 일부분을 참조**하는 방법

+ 문자열 슬라이싱

- 기존의 문자열은 절대 수정이 되는 것이 아니라 **원 문자열은 변함이 없고 일부분을 반환**

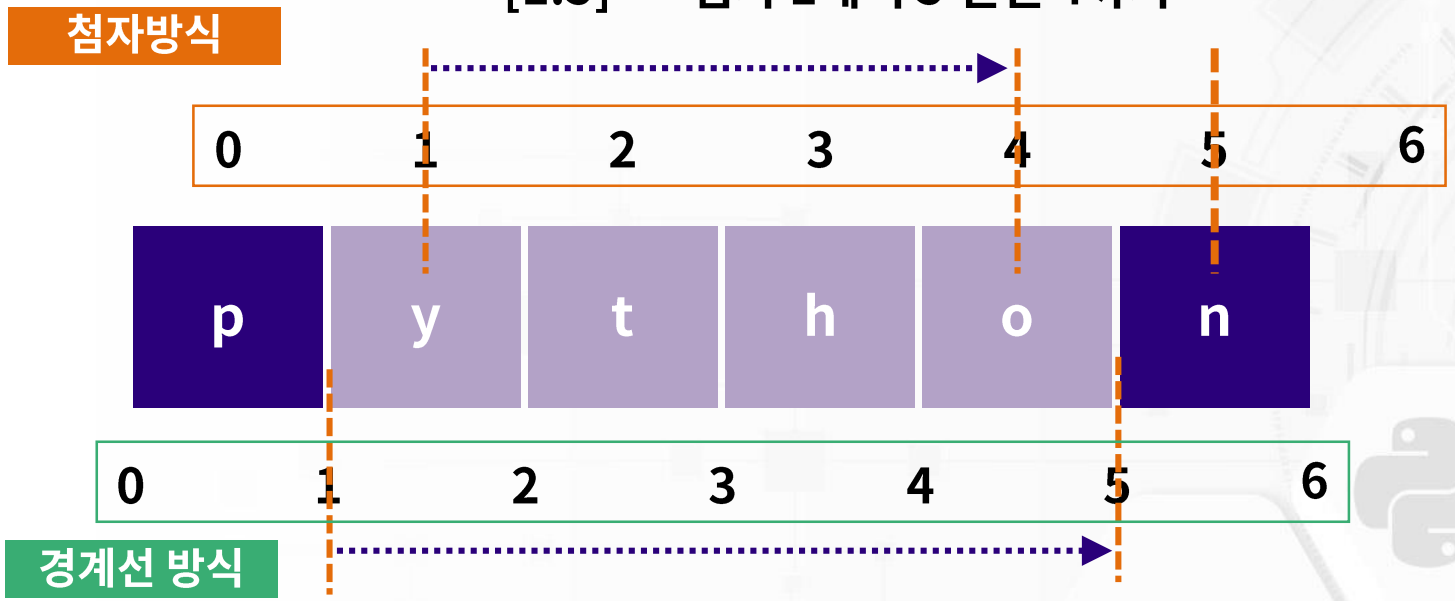
+ str[start:end]

- 문자열 **str에서 start 첨자에서 end-1 첨자까지**의 문자열을 반환

⚠ 슬라이싱: 문자열의 부분 문자열 참조 방식

'python'[1:5] == 'ytho'

[1:5] == 첨자 1에서 5 전인 4까지



[1:5] == 경계선 1에서 5까지

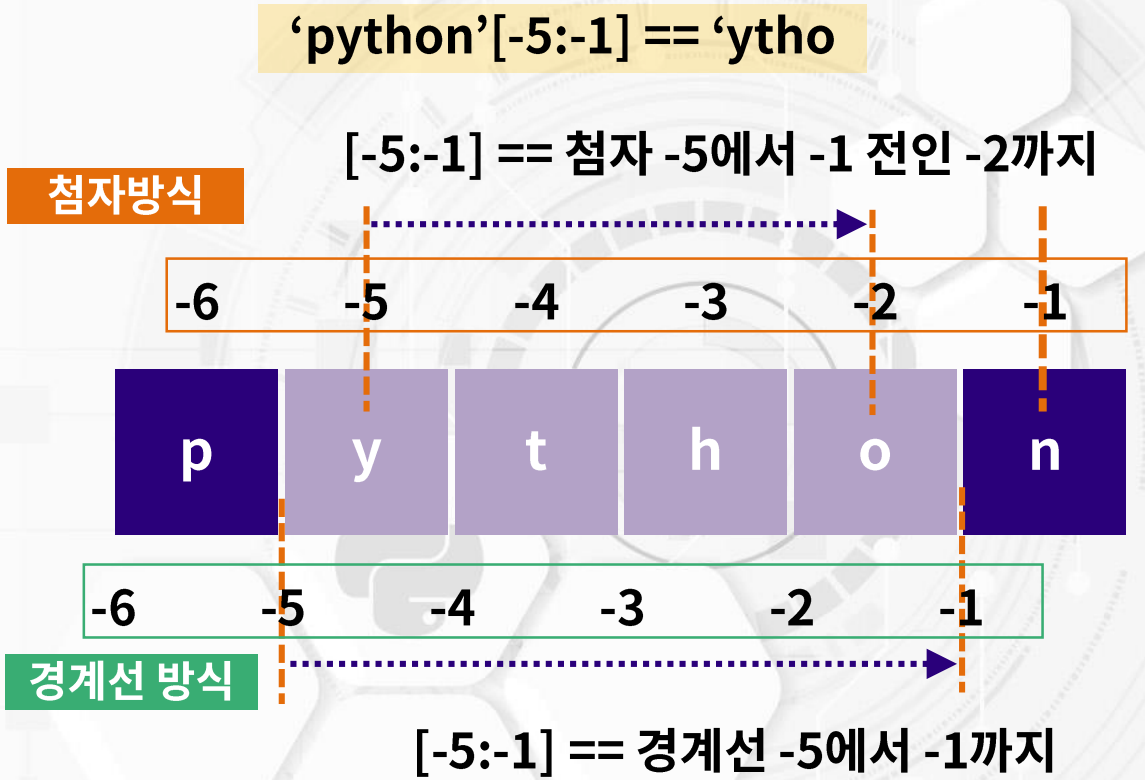
⚠ 슬라이싱: 문자열의 부분 문자열 참조 방식

```
>>> 'python'[1:5]
'ytho'
>>> 'python'[2:4]
'th'
>>> 'python'[0:3]
'pyt'
>>> 'python'[0:6]
'python'
>>> 'python'[0:len('python')]
'python'
```

⚠ 첨자의 음수 사용과 음수와 양수 혼합하여 사용 가능

+ 음수를 이용한 문자열 슬라이싱

```
>>> 'python'[-5:-1]
'ytho'
>>> 'python'[-4:-1]
'tho'
>>> 'python'[-6:-3]
'pyt'
>>> 'python'[-6:-1]
'pytho'
>>> 'python'[-len('python'):-1]
'pytho'
```

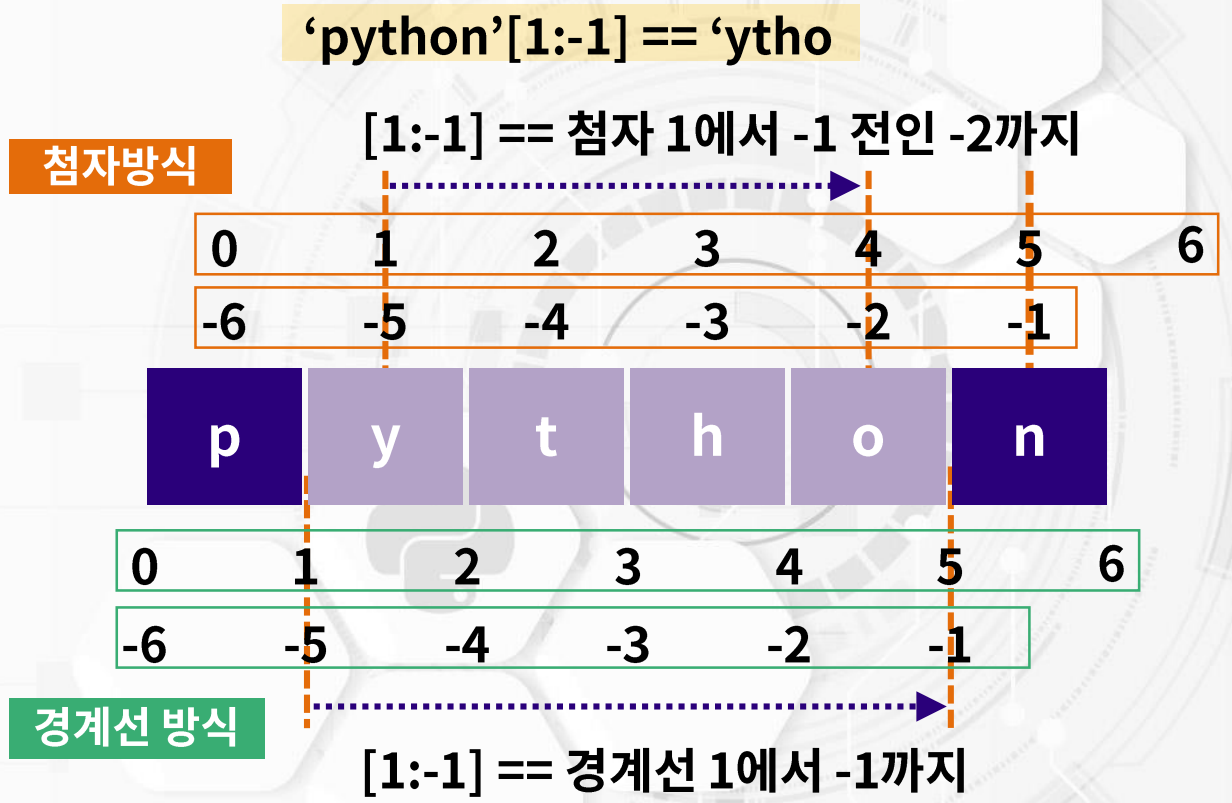


[그림8-2] 음수를 이용한 문자열 슬라이싱

⚠ 첨자의 음수 사용과 음수와 양수 혼합하여 사용 가능

+ 음수와 0, 양수 첨자를 이용한 문자열 슬라이싱

```
>>> 'python'[1:-1]
'ytho'
>>> 'python'[0:-2]
'pyth'
>>> 'python'[0:-1]
'pytho'
>>> 'python'[2:-2]
'th'
>>> 'python'[-5:5]
'ytho'
>>> 'python'[-6:4]
'pyth'
>>> 'python'[-4:6]
'thon'
```



[그림8-2] 음수를 이용한 문자열 슬라이싱

⚠ start와 end를 비우면 '처음부터'와 '끝까지'를 의미

+ start와 end, 생략하면 각각 '처음부터'와 '끝까지'를 의미

+ 앞뒤를 모두 비우면 문자열 전체를 반환

```
>>> 'python'[:3]
'pyt'
>>> 'python'[:4]
'pyth'
>>>
>>> 'python'[1:]
'ython'
>>> 'python'[3:]
'hon'
```

```
>>> 'python'[:] # 전체 반환
'python'
```

⚠ start와 end를 비우면 '처음부터'와 '끝까지'를 의미

[코딩실습] 슬라이싱으로 문자열의 부분 문자열 참조 난이도 기본

```
1. >>> str = 'Monty Python'
2. >>> len(str)
3. 12
4. >>> str[0:5], str[6:], str[6:12]
5. ('Monty', 'Python', 'Python')
6. >>> str[-12:-7], str[-6:], str[-6:0]
7. ('Monty', 'Python', '')
```


- start 첨자는 end 첨자보다 문자열 범위 내에서 왼쪽에 위치한 첨자여야 반환문자열이 존재, 그렇지 않으면 공백문자열 반환(오류는 발생하지 않음)

⚠ 문자 사이의 간격을 step으로 조정 가능

+ str[start:end:step]

- 문자 사이의 간격을 step으로 조정, step을 생략하면 1

+ str[:] == str[::] == str[::1]

- 전체 문자열 반환

+ 간격인 step은 음수도 가능

- start는 end보다 오른쪽에 위치한 첨자여야 함

+ str[::-1]은 역순 문자열 반환

⚠ 문자 사이의 간격을 step으로 조정 가능

```
>>> 'python'[1:5:3]
'yo'
>>> 'python'[::-1] # 역순의 문자열을 반환
'nothyp'
>>> 'python'[5:0:-1]
'nohty'
>>> 'python'[-1:-7:-1]
'nohtyp'
```

Chapter 2.

문자 함수

PYTHON PROGRAMMING

⚠ 문자 함수 ord( )와 chr( )

+ ord(‘문자’)

- ‘문자’의 코드 번호 반환

+ chr(코드번호)

- 해당 코드 번호의 문자 반환

```
>>> ord(‘가’) # 44032
44032
>>> chr(44032) # ‘가’
‘가’
>>> hex(ord(‘가’)) # 44032의 16진수 ac00
‘0xac00’
>>> hex(ord(‘힣’))
‘0xd6a7’
```

끝

Chapter 3.

문자 표현

PYTHON PROGRAMMING

⚠ 이스케이프 시퀀스 문자(escape sequence characters)

✚ 하나의 문자를 역슬래시(\)로 시작하는 조합으로 표현하는 문자

```
>>> '\a' # bell alert sounds
'\x07'
>>> '\b' # backspace(BS) removes previous character
'\x08'
>>> print("ab" + "\b" + "c")
ac
>>> '\n' # newline
'\n'
>>> print("hello\nworld")
hello
World
# Prints a character from the Unicode database
>>> '\N{DAGGER}'
'†'
```

⚠ 이스케이프 시퀀스 문자(escape sequence characters)

이스케이프 시퀀스 문자	설명	이스케이프 시퀀스 문자	설명
\\	역슬래시	\f	폼피드(form feed) (예전 프린터에서 다음 페이지의 첫 줄로 이동)
\’	작은 따옴표	\t	수평 탭
\”	큰 따옴표	\v	수직 탭
\a	벨소리(알람)	\uxxxx	16비트 16진수 코드
\b	백스페이스 (이전 문자 지우기)	\Uxxxxxxxx	32비트 16진수 코드
\n	새 줄	\ooo	8진수의 코드 문자
\N{name}	유니코드의 이름	\xhh	16진수의 코드 문자
\r	동일한 줄의 맨 앞 으로 이동 (파이썬 쉘에서는 다음 줄로)		

끝

Chapter 4.

내장 함수

PYTHON PROGRAMMING

⚠ 내장 함수 min( )과 max( )

+ 인자의 최댓값과 최솟값을 반환하는 함수

1 인자가 문자열 1개

- 문자열을 구성하는 문자에서 코드 값으로 최대와 최소인 문자를 반환

2 인자가 문자열이 2개 이상

- 문자열 중 최대와 최소인 문자열을 반환

3 2개 이상의 숫자

- 최대와 최소 수를 반환

⚠ 내장 함수 min( )과 max( )

+ 내장 함수(built - in function)

- 함수 **len( ), print( ), int( ), float( ), str( ), type( ), min( ), max( )** 등
- 파이썬 라이브러리로 인터프리터에서 아무런 설정 없이
  (나중에 배울 부가적인 설치와 임포트 필요 없이) 바로 사용할 수 있는 함수
- 파이썬 내장 함수 설명
  - docs.python.org/ko/3/library/functions.html

⚠ 내장 함수 min( )과 max( )

```
>>> min('ipython')
'h'
>>> max('ipython')
'y'
>>> min('3259')
'2'
>>> max('3259')
'9'
>>> min(3, 96.4, 13)
3
>>> max(3, 96.4, 13)
96.4
>>> min('ipython', 'java')
'ipython'
>>> max('ipython', 'java')
'java'
```

⚠문자열

… 객체와 클래스
… 문자열 길이: len()
… 첨자(indexing): 문자열[n]
… 슬라이싱(slicing): 문자열[start:end:step]

⚠문자 함수

… ord(): 코드 번호 반환
… chr(): 해당 코드 번호의 문자 반환

⚠문자 표현

… 이스케이프 시퀀스:
하나의 문자를 역슬래시(\)로 시작하는 조합으로 표현하는 문자

⚠내장 함수

… min() max(): 인자의 최댓값과 최솟값을 반환하는 함수