컴퓨터정보과

파이선 프로그래밍

03주차 – 자료형

자료형과 기본 연산 (2장)

그룹	자료형	생성자	리터럴 표현	내용
수치형 (Number)	Integer	int()	1	정수
	Floating point number	float()	1.1	실수
	Complex Number	complex()	2 + 3j	복소수
논리형 (Boolean)	Boolean	bool()	True	참(True)과 거짓(False)만 값을 갖음
군집형 (Collection)	**String**	**str()**	**“1” 혹은 ‘1’**	**문자열 (순서 O, 수정 X, 중복 O, 구성요소한정 O)**
	List	list()	[1,2]	리스트 (순서 O, 수정 O, 중복 O, 구성요소한정 X)
	Tuple	tuple()	(1,2)	튜플 (순서 O, 수정 X, 중복 O, 구성요소한정 X)
	Set	set()	{1,2}	집합, 세트 (순서 X, 수정 O, 중복 X, 구성요소한정 X)
	Dictionary	dict()	{1:”1”, 2:”2}	사전, 딕셔너리 (순서 X, 수정 O, 중복 X, 구성요소한정 X) {key:value, … , key_n:$value_n$}

문자열형

- 문자열형 (string) p.50
  - 문자열: 문자들의 집합
  - 리터럴 표현
    - " ~ " , ' ~ ', """ ~ """, ''' ~ '''
  - 따옴표 안의 따옴표
    - "Inha's dream" , "Inha\" s dream" ,
    - 'Inha" s dream', 'Inha\'s dream'
  - 여러 줄 표현
    - "inha \nuniv"
    - """inha
      univ"""
    - '''inha
      univ'''
  - escape code p.54

문자열 관련 연산자 & 함수

- 문자열 관련 연산 p.55
  - 문자열 연결 연산자 : + (str + str)
    - "a" + "b" ➔ "ab"
  - 문자열 반복 연산자 : * (str * int)
    - "a" * 3 ➔ "aaa"
- 문자열 관련 내장 함수 p.56
  - len() : 문자열 길이
    - len("ab") ➔ 2

인덱싱 - 문자열

- Indexing p.56
  - 복합 형태의 자료형에 [] 대괄호를 사용하여 원하는 위치(인덱스) 번호를 넣어 값을 알아내는 방법
  - 대부분의 컴퓨터 언어는 0에서부터 숫자를 시작한다. (일부 예외가 있음)
  - 문자열은 문자가 순서대로 배치된 복합체이다.
    - 인덱싱 사용 가능
  - 예제 : "I like Python"
    - a = "I like Python"
      - a[0]
      - a[-13]
      - a[12]
      - a[-1]
      - a[len(a)-1]

"I"	" "	"l"	"i"	"k"	"e"	" "	"P"	"y"	"t"	"h"	"o"	"n"
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

  - *문자열의 인덱싱의 결과는 문자열*

슬라이싱 - 문자열

- Slicing p.58
  - 하나의 요소만 뽑는 것이 아니라 범위 값을 이용해서 한 개 이상의 요소를 추출하는 방법
  - 형태 : 문자열[시작인덱스 : 종료인덱스]
    - 범위 : 시작인덱스 <= 문자열 < ***종료인덱스***
  - 예제 : "I like Python"
    - a = "I like Python"
      - a[0:4]
      - a[2:6]
      - a[7:]
      - a[:4]
      - a[:]

I		l	i	k	e		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

  - *문자열의 슬라이싱의 결과는 문자열*
  - (주의) 인덱싱/슬라이싱은 기존의 문자열을 분해하는 것이 아니다. 문자열은 **변경 불가능한(immutable)** 자료형이다.

문자열 포매팅 - 방법 1

- % 포맷팅 p.62
  - old-style string formatting (like C Lang)
  - 예제

```
a = "나는 %d살 입니다." % 20
age = 21
a = "나는 %d살 입니다." % age
b = "내 이름은 %s 입니다." % "인하"
name = "정석"
b = "내 이름은 %s 입니다." % name
c = "내 이름은 %s 이고, %d 살입니다." % (name, age)
```

  - 문자열 포맷 코드
    - 만능 포맷 코드 : %s
  - 정렬과 공백
    - %10s : 전체 길이 10, 오른쪽 정렬
    - %-10s : 전체길이 10, 왼쪽 정렬
  - 소수점 표현 : %10.4f

문자열 포매팅 - 방법 2

- str.format() 메소드 p.67
  - 예제

```
a = "나는 {0}살 입니다. “ .format(20)

age = 21
a = "나는 {0}살 입니다. “ .format(age)
b = "내 이름은 {0} 입니다. “ .format("인하")

name = "정석"
b = "내 이름은 {0} 입니다.".format(name)
c = "내 이름은 {0} 이고, {1} 살입니다.".format(name, age)
c = "내 이름은 {name} 이고, {age} 살입니다.".format(name= "정석", age=21)
c = "내 이름은 {0} 이고, {age} 살입니다.".format("정석", age=21)
```

문자열 포매팅 - 방법 2

- str.format() 메소드 p.67
  - 정렬

```
a = "'{0:<10}'".format("hi") #왼쪽 정렬      'hi        '
a = "'{0:>10}'".format("hi") #오른쪽 정렬     '        hi'
a = "'{0:^10}'".format("hi") #가운데 정렬     '    hi    '
```

  - 공백 채우기

```
a = "'{0:=^10}'".format("hi") # '====hi====‘
a = "'{0:!<10}'".format("hi") # 'hi!!!!!!!! ’
```

  - 소수점

```
"{0:0.4f}".format(3.141592)        # 3.1416
```

문자열 포매팅 - 방법 3

- Formatted String Literals p.71
  - f-string / 문자열보간법
  - python 3.6 버전 이상만 사용 가능, str.format()의 개선 문법
  - 문자열 앞에 f접두사를 붙여서 사용
  - 표현식(변수와 수식을 함께 사용하는 것)을 지원
  - 예제

```
name = "김인하"
age = 20
intro = "내 이름은 {name}입니다. 나이는 {age}살 입니다."
print(intro)
intro = f"내 이름은 {name}입니다. 나이는 {age}살 입니다."
print(intro)
intro = f"내 이름은 {name}입니다. 나이는 {age + 1}살 입니다."
print(intro)
```

  - 정렬/공백 채우기/소수점 표현은 format() 함수와 동일

문자열 관련 메소드

- 문자열 메소드 p.72
  - 메소드 : 자료형이 자체적으로 갖는 함수
    - 문자형 객체 뒤에 '.'을 붙여서 문자열형만 호출할 수 있는 함수
    - 함수 : 특정 자료형에 국한되지 않는 함수
  - count() : 문자 개수 세기
  - find() : 문자열 위치 찾기, 찾지 못하면 -1 반환
  - index() : 문자열 위치 찾기, 찾지 못하면 오류 발생
  - join() : 문자열 삽입
  - upper() : 대문자로 바꾸기
  - lower() : 소문자로 바꾸기
  - strip() / lstrip() / rstrip() : 문자 공백 지우기
  - replace() : 문자열 바꾸기
  - split() : 문자열 나누기
- (중요) 문자열은 불변 자료형이다