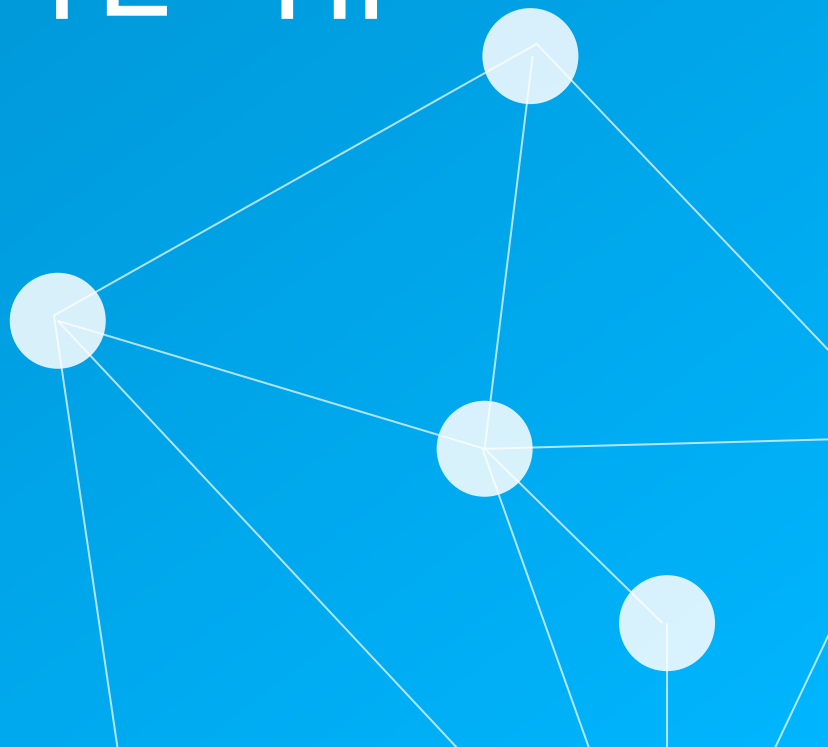
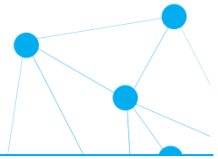
파이썬
자료구조

파이썬 리뷰



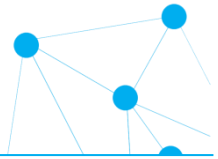
자료형, 리터럴과 변수



- 리터럴과 자료형

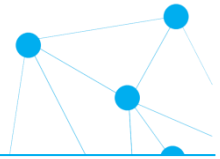
분류	내장 자료형	리터럴
수치	정수(int)	10 -30 0xfffe 073
	실수(float)	3.14 -0.45 123.032E-13
	복소수(complex)	complex(1,2) 1+2j 4+5j
	부울(bool)	True False
시퀀스	문자열(str)	'game' "over" "C"
	리스트(list)	[] [0, 1, 2, 3] [0, 'hello', 3.14]
	튜플(tuple)	(0, 1, 2, 3) ('hello', 'world', 'game')
매핑	딕셔너리(dict)	{ 3.14 : "phi", 4.5 : "score" }
집합	집합(set, frozenset)	{ 1, 2, 3 } {'one', 'two', 'three' }

딕셔너리(dictionary, 사전)



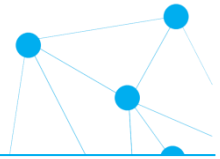
- 각 항목은 key 값과 value 값 쌍
- 예: 영한사전에서 key 값은 영어단어, value 값은 한글 설명
- { }중괄호로 감싸서 정의
- {key1: value1, key2:value2, …}
- 데이터의 저장순서가 중요하지 않다. 인덱스로 접근하지 않고 **key 값으로 접근한다**: 딕셔너리이름[키값]
- 수정, 삽입, 삭제 가능하다.

딕셔너리 예



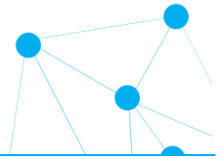
- d = {'cs101':'phyton programming', 'cs202':'data structures'}
- d['cs202’]
- d['cs101'] = 'phython programming' # 수정
- d['cs301'] = 'operating system' # 새로운 항목 삽입
  # key값이 'cs301'인 항목가 없을 경우 삽입

딕셔너리 주의점 및 유용한 함수들



- d = {'cs101':'phyton programming', 'cs202':'data structures'}
- key 값은 모두 달라야 한다
- a = {1: 'A', 2: 'B']
- d['cs301'] = 'operating system' # 새로운 항목(원소) 삽입
  # key값이 'cs301'인 항목(원소)가 없을 경우 삽입
- del d['cs202'] # key 값이 'cs202'인 원소 삭제

딕셔너리 유용한 함수들



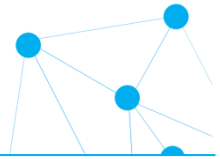
- d = {'cs101':'phyton programming', 'cs202':'data structures'}

- d.get(key, default): 딕셔너리 d에서 key인 항복의 value 값 반환. key가 없으면 default 값을 반환(default 값을 생략하면 d[key]와 동일)

  print(d.get('cs101'))

- d.keys() : d의 key 값들의 객체를 돌려준다.
  k = list(d.keys())
- D.values(): d의 value 값들의 객체를 알려준다.
  v = list(d.values())
- d.items() : d의 (key, value) 튜플의 객체를 돌려준다
  itemList = list(d.items())

집합(set)

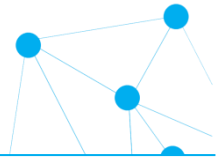


- 서로 다른 값들의 모임

- K리스트, 튜플과 차이: 리스트와 튜플은 순서대로 저장되고, 인덱스로 접근 가능. 집합은 순서없이 저장되어 접근 불가

- a = {1, 2, 3}
- a1 = set((1,2,3))
- a2 = set([1,2,3])
- a3= set('Hello')
- a1 = {}

집합 연산들



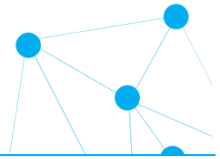
- 원소 삽입
  s = {1,2}
  s.add(3)

- 여러 개 원소를 동시에 추가 함수: update
  s.update([3,4,5])

- 원소 삭제
  s.remove(2)

- 집합 크기 함수: len
  len(s)

집합 연산들



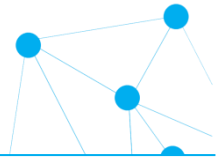
- 두 집합이 교집합, 합집합, 차집합

- 합집합 union
  s.union(t)

- 교집합 intersection
  s.intersection(t)

- 차집합 difference
  s.difference(t)

컴프리엔션(comprehension)



- 하나의 sequence로부터 다른 sequence를 정의하는 간단하고 유용한 방법

- list comprehension
  일반적인 형식 [out_exp for element in a_list if condition]

예: 1부터 30까지의 3의 배수들 리스트 구성

- 방법 1

```
a = []
for x in range(31):
        if x % 3 == 0:
            a.append(x)
```

- 방법 2

```
a = [x for x in range(31) if x%3 = 0]
```

리스트 컴프리엔션(comprehension) 예

- 1부터 9까지의 제곱한 값들의 리스트 구성

```
a = [x*x for x in range(10)]
```

- 한 줄로 주어진 성적들을 입력하여 정수 성적 리스트 만들기

```
scoreList = [int(x) for x in input().split()]
```