인공지능 기초 프로그래밍

Python 기초 (자료형2)

Daegu Campus of
KOREA POLYTECHNIC

파이썬 프로그래밍의 기초, 자료형 II (Recap)

- 리스트 자료형

  - **리스트(List)란?**

    - 자료형의 집합을 표현할 수 있는 자료형

```
>>> odd = [1, 3, 5, 7, 9]
```

    - 숫자와 문자열만으로 프로그래밍을 하기엔 부족한 점이 많음
      - 예) 1부터 10까지의 숫자 중 홀수 모음인 집합 {1, 3, 5, 7, 9}는 숫자나 문자열로 표현 불가능
      - 리스트로 해결 가능!

파이썬 프로그래밍의 기초, 자료형 II (Recap)

- 리스트 자료형

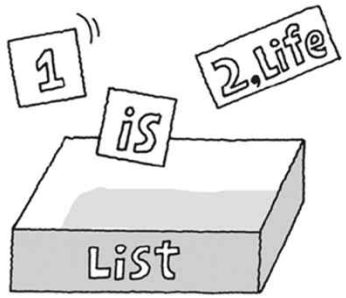
▪ 리스트 사용법

- 대괄호([ ])로 감싸고 각 요솟값은 쉼표(,)로 구분

```
리스트명 = [요소1, 요소2, 요소3, …]
```

- 리스트 안에 어떠한 자료형도 포함 가능

```
>>> a = []
>>> b = [1, 2, 3]
>>> c = ['Life', 'is', 'too', 'short']
>>> d = [1, 2, 'Life', 'is']
>>> e = [1, 2, ['Life', 'is']]
```



파이썬 프로그래밍의 기초, 자료형 II (Recap)

- 리스트 자료형
  - **리스트 인덱싱**
    - 문자열과 같이 인덱싱 적용 가능

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
```

    - 파이썬은 숫자를 0부터 세기 때문에 a[0]이 리스트 a의 첫 번째 요소

```
>>> a[0]
1
```

    - a[-1]은 리스트 a의 마지막 요솟값

```
>>> a[-1]
3
```

    - 요솟값 간의 덧셈

```
>>> a[0] + a[2]  ◀── 1 + 3
4
```

파이썬 프로그래밍의 기초, 자료형 II (Recap)

- 리스트 자료형
  - **리스트 인덱싱**
    - 리스트 내에 리스트가 있는 경우

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

    - a[-1]은 마지막 요솟값인 리스트 ['a', 'b', 'c'] 반환

```
>>> a[0]
1
>>> a[-1]
['a', 'b', 'c']
```

    - 리스트 a에 포함된 ['a', 'b', 'c'] 리스트에서 'a' 값을 인덱싱을 사용해 반환할 방법은?

```
>>> a[-1][0]
'a'
```

    - a[-1]로 리스트 ['a', 'b', 'c']에 접근하고, [0]으로 요소 'a"에 접근

파이썬 프로그래밍의 기초, 자료형 II (Recap)

- 리스트 자료형
  - **리스트 슬라이싱**
    - 문자열과 같이 슬라이싱 적용 가능

```
>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
[1, 2]
```

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a[:2]  ◀── 처음부터 a[1]까지
>>> c = a[2:]  ◀── a[2]부터 마지막까지
>>> b
[1, 2]
>>> c
[3, 4, 5]
```

파이썬 프로그래밍의 기초, 자료형 II (Recap)

- 리스트 자료형
  - **리스트 연산하기**
    - **더하기(+)**
      - + 기호는 2개의 리스트를 합치는 기능
      - 문자열에서 "abc" + "def" = "abcdef"가 되는 것과 같은 의미

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

    - **반복하기(*)**
      - * 기호는 리스트의 반복을 의미
      - 문자열에서 "abc" * 3 = "abcabcabc"가 되는 것과 같은 의미

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

    - **리스트 길이 구하기**
      - len( ) 함수 사용
      - 문자열, 리스트 외에 앞으로 배울 튜플과 딕셔너리에서도 사용 가능한 내장 함수

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수
  - **리스트의 수정과 삭제**
    - **리스트에서 값 수정하기**

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]
```

    - **리스트 요소 삭제하기**
      - del 키워드 사용

del 객체

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
```

```
>>> a = [1, 2, 3, 4, 5]
>>> del a[2:]
>>> a
[1, 2]
```

※ 슬라이싱 기법 활용 가능

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수
  - **리스트 관련 함수**
    - **append( )**
      - 리스트의 맨 마지막에 요소 추가

```
>>> a = [1, 2, 3]
>>> a.append(4) ◀── 리스트의 맨 마지막에 4를 추가
>>> a
[1, 2, 3, 4]
```

      - 어떤 자료형도 추가 가능

```
>>> a.append([5,6]) ◀── 리스트의 맨 마지막에 [5,6]을 추가
>>> a
[1, 2, 3, 4, [5, 6]]
```

    - **sort( )**
      - 리스트의 요소를 순서대로 정렬

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

      - 문자의 경우 알파벳 순서로 정렬 가능

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수
  - **리스트 관련 함수**
    - **reverse( )**
      - 리스트를 역순으로 뒤집어 줌
      - 요소를 역순으로 정렬하는 것이 아닌, 현재의 리스트 그대로 뒤집음

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a
['b', 'c', 'a']
```

    - **index( )**
      - 요소를 검색하여 위치 값 반환

```
>>> a = [1,2,3]
>>> a.index(3)  ◀── 3은 리스트 a의 세 번째(a[2]) 요소
2
```

      - 값이 존재하지 않으면, 값 오류 발생

```
>>> a.index(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>   ── 오류 메시지
ValueError: 0 is not in list
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수

▪ 리스트 관련 함수

▪ insert( )

- 리스트에 요소 삽입
- insert(a, b)
  - a번째 위치에 b를 삽입하는 함수

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4) ◀── a[0] 위치에 4 삽입
[4, 1, 2, 3]
```

▪ remove( )

- remove(x)
  - 리스트에서 첫 번째로 나오는 x를 삭제

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
[1, 2, 1, 2, 3]
```

- 값이 여러 개인 경우 첫 번째 것만 삭제

```
>>> a.remove(3)
[1, 2, 1, 2]
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수

▪ 리스트 관련 함수

- **pop( )**
  - 리스트의 맨 마지막 요소를 돌려주고 해당 요소 삭제
  - pop(x)
    - 리스트의 x번째 요소를 돌려주고 해당 요소 삭제

```
>>> a = [1, 2, 3]
>>> a.pop()
3
>>> a
[1, 2]
```

```
>>> a = [1, 2, 3]
>>> a.pop(1)
2
>>> a
[1, 3]
```

- **count( )**
  - 리스트에 포함된 요소의 개수 반환
  - count(x)
    - 리스트 안에 x가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수

```
>>> a = [1, 2, 3, 1]
>>> a.count(1)
2
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수

▪ 리스트 관련 함수

- **extend( )**
  - 리스트에 리스트를 더하는 함수
  - extend(x)
    - x에는 리스트만 올 수 있음

a.extend([4, 5])

=

a += [4, 5]

```
>>> a = [1, 2, 3]
>>> a.extend([4, 5])
>>> a
[1, 2, 3, 4, 5]
>>> b = [6, 7]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```


자료형 II - Practice #2 (list)

```python
# list add/delete
list_a = [1, "2", 4]
print(f'{list_a}')
list_a[1] = 4
print(f'{list_a}')
list_a = list_a + [5, 6]
print(list_a)
del list_a[2]
print(list_a)

str_1 = "123"
print(str_1[0], str_1[1], str_1[2])
# str_1[2] = "3"
# str_1 = str_1[:2]+"4"
# print(str_1)

# list append / sort
test_list = [1, 2, 3]
test_list.append([4, 8, 6, 7, 9])
test_list.append("string")
```

파이썬 프로그래밍의 기초, 자료형 II

- 튜플 자료형

▪ 튜플(Tuple)이란?

▪ 리스트와 유사한 자료형

리스트	튜플
[ ]로 둘러쌈	( )로 둘러쌈
생성 / 삭제 / 수정 가능	값 변경 불가능

```
>>> t1 = ()
>>> t2 = (1,)
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3
>>> t5 = ('a', 'b', ('ab', 'cd'))
```

- 튜플은 1개의 요소만을 가질 때는 요소 뒤에 콤마(,)를 만드시 붙여야 함 (예) t2 = (1,)
- 괄호( )를 생략해도 무방함 (예) t4 = 1, 2, 3
- 프로그램이 실행되는 동안 값을 유지해야 한다면 튜플을, 수시로 값을 변경해야 하면 리스트 사용

파이썬 프로그래밍의 기초, 자료형 II

- 튜플 자료형

  - **튜플의 요솟값을 지울 수 있을까?**

    - 튜플의 요솟값은 한 번 정하면 지우거나 변경할 수 없음!

```
>>> t1 = (1, 2, 'a', 'b')
>>> del t1[0] ◀── 튜플 t1의 첫 번째 요소를 지우려고 시도
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
```

형 오류(Type Error) 발생

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0] = 'c' ◀── 튜플 t1의 첫 번째 요솟값을 변경하려고 시도
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

형 오류 발생

파이썬 프로그래밍의 기초, 자료형 II

- 튜플 자료형

▪ 튜플 다루기

▪ 인덱싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0]
1
>>> t1[3]
'b'
```

▪ 슬라이싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[1:] ◀── t1[1]부터 끝까지
(2, 'a', 'b')
```

▪ 튜플 더하기와 곱하기

```
>>> t2 = (3, 4)
>>> t1 + t2
(1, 2, 'a', 'b', 3, 4)
```

```
>>> t2 * 3
(3, 4, 3, 4, 3, 4)
```

▪ 튜플 길이 구하기

- len( ) 함수

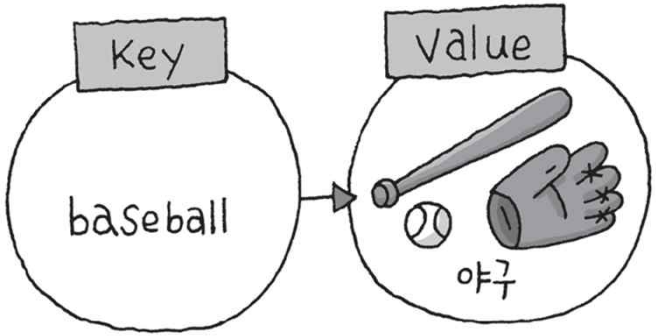
```
>>> t1 = (1, 2, 'a', 'b')
>>> len(t1)
4
```

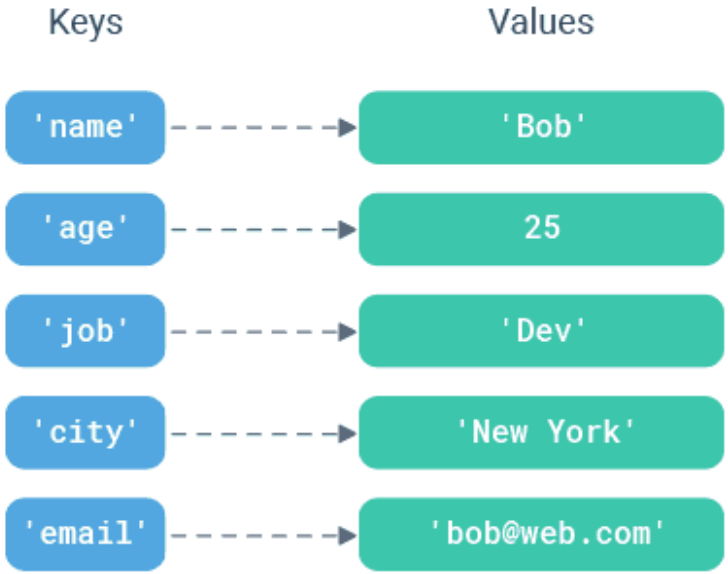
Homework

- 정수형, 실수형, 문자열 및 “리스트 내에 리스트” 등을 포함하여 자신을 소개하는 리스트를 만들고, 리스트 인덱싱과 슬라이싱 기법 등을 이용하여 자기 소개하는 것 표현해보기
- 문자열 변수에서 특정 인덱싱으로 접근 후 문자열을 변경할 수 있는가? 답변 및 가능한 방법 코드 작성 및 제출, mutable, immutable 자료형 정리해보기.
- List에서 Append, insert, extend 차이점, remove/pop 차이점, 각각 예제 코드 작성 및 설명
- 기존 파싱 코드에서 자료형 list와 문자열 관련 함수 split()을 이용하여 동일한 결과 도출하기
- 기존 파싱 코드에서 데이터를 어떻게 만들면 더욱 효율적으로 처리할 수 있는지
  - Data 형식 변경 및 추가, 코드 작성 결과 확인 (조건은 동일함, 크기가 100보다 크면 vehicle에서 truck으로)

파이썬 프로그래밍의 기초, 자료형 III

- 딕셔너리(Dictionary)란?
  - 대응 관계를 나타내는 자료형
  - 연관 배열(Associative aray) 또는 해시(Hash)
  - Key와 Value를 한 쌍으로 갖는 자료형
  - 순차적으로 해당 요솟값을 구하지 않고, Key를 통해 Value를 바로 얻는 특징





파이썬 프로그래밍의 기초, 자료형 III

- 딕셔너리(Dictionary)

  - **딕셔너리의 모습**

```
{Key1:Value1, Key2:Value2, Key3:Value3, ···}
```

  - Key와 Value의 쌍 여러 개 (Key : Value)

  - { }로 둘러싸임
  - 각 요소는 쉼표(,)로 구분됨

```
>>> dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
```

```
>>> a = {1: 'hi'}
```

```
>>> a = {'a': [1,2,3]}
```

파이썬 프로그래밍의 기초, 자료형 III

- 딕셔너리(Dictionary)

▪ 딕셔너리 쌍 추가, 삭제하기

▪ 딕셔너리 쌍 추가

```
>>> a = {1: 'a'}
>>> a[2] = 'b'  ◀── {2:'b'} 쌍 추가
>>> a
{1: 'a', 2: 'b'}
```

▪ 딕셔너리 요소 삭제

```
>>> del a[1]  ◀── key가 1인 key:value 쌍 삭제
>>> a
{'name': 'pey', 3: [1, 2, 3], 2: 'b'}
```

▪ 딕셔너리에서 Key 사용해 Value 얻기

```
>>> grade = {'pey': 10, 'julliet': 99}
>>> grade['pey']  ◀── Key가 'pey'인 딕셔너리의 Value를 반환
10
>>> grade['julliet']  ◀── Key가 'julliet'인 딕셔너리의 Value를 반환
99
```

- 리스트나 튜플, 문자열은 요솟값 접근 시 **인덱싱**이나 **슬라이싱** 기법을 사용

- 딕셔너리는 **Key**를 사용해 **Value** 접근

주의 사항:
- 동일한 key를 사용할 경우, 1개를 제외한 나머지 key:value 값은 모두 무시
- Key에는 변할 수 있는 자료형 사용하면 Error 발생

Practice #1 (Dictionary)

```python
# Practice 1
a = {1: 'a', 2: 'b'}
a[3] = 'c'
print(a)
a['name'] = ["Shin", "Park", "Kim"]
print(a)
del a[1] # Dictionary a에서 key가 1인 key:value 쌍을 삭제한다.
print(a)
a = {"Dept": ["AI-Engineering", "Smart Electronic"], "StudentNum": [22, 60]}
print(a['Dept'], a['StudentNum'])
a = {'1': 'a', '2': 'b', '3': 'c'}
# Key를 중복해서 사용하면 하나만 남기고 나머지는 무시
a['1'] = 'aa'
a['1'] = 'aaa'
print(a)

# Error Case
a = {
#     ['name', 'age']: (["Shin", "Kim", "Park"], [22, 23, 25])
    ('name', 'age'): (["Shin", "Kim", "Park"], [22, 23, 25])
}
print(a)
```

Thank you

Q&A

www.kopo.ac.kr
jsshin7@kopo.ac.kr

