Python 기초: 문자열과 배열

- 지능정보공학설계, Week 5 -

이 지 항 (Ph.D.)

Assistant Professor
Department of Human-Centered AI
Sangmyung University, Seoul, KR
(jeehang@smu.ac.kr)

문자열

배열의 종류

따옴표

- 문자열은 큰따옴표와 작은따옴표 모두 사용 가능

명령문	print("hello") # 'hello'를 출력! print('hello') # 'hello'를 출력!
결과	hello hello

- 문자열에 작은따옴표가 포함되어 있다면? 큰따옴표로 묶어주세요

명령문	print("My friend's house.")
결과	My friend's house.

- 문자열에 큰따옴표가 포함되어 있다면? 작은따옴표로 묶어주세요

명령문	print('그녀가 말했다. "안녕!".')
결과	그녀가 말했다. "안녕!".

인덱스

- 문자열을 변수에 저장하면 자동적으로 배열이 만들어짐

명령문	animal = 'frog' print(animal[3])
결과	g

- n개의 저장 장소가 있다면 인덱스는 0부터 n-1까지 존재 (3개의 저장소? 0부터 2)

- 마지막 항목에서부터 -1로 시작하여 1씩 감소하는 인덱싱 방법도 제공

명령문	animal = 'frog' print(animal[-1])
결과	g

- 한글에서도 같은 방식으로 동작

문자열 슬라이싱

- 인자 형식 [시작: 끝+1: 단계]

```
[시작:끝+1:단계]
```

- 인자가 1개인 경우

명령문	animal = 'frog' print(animal[1])
결과	r

- 인자가 2개인 경우

명령문	animal = 'frog' print(animal[1:3])
결과	ro

- 인자가 3개인 경우

명령문	animal = 'frog' print(animal[0:3:2])
결과	fo

인자가 생략된 경우

- 콜론(:)이 1개일 경우

명령문	`animal = 'frog'`
	`print(animal[:])`
	`print(animal[1:])`
	`print(animal[:2])`

결과	`frog` `# [:]는 전체의 문자열 출력`
	`rog` `# [1:]는 인덱스 1에서부터 문자열의 끝까지 출력`
	`fr` `# [:2]는 처음부터 2직전까지의 내용 출력`

- 콜론(:)이 2개일 경우

명령문	`animal = 'elephant'`
	`print(animal[::2])`
	`print(animal[::-2])`

결과	`eehn` `tapl`

컴퓨터에서 배열/리스트 인덱스는 0부터 시작한다는 것을 늘 기억하세요!

문자열의 병합

- 2개의 문자열 사이에 '+' 기호를 넣으면 앞뒤의 문자열이 병합됨

명령문	dog = '개' animal = '진돗' + dog print(animal)
결과	진돗개

문자열 함수 (메소드; method)

- **len() 함수 – 문자열의 길이를 알고 싶다!**
  - 괄호 안의 문자열(객체)의 길이를 반환

명령문	animal = 'elephant' print(len(animal))
결과	8

**** 함수 (메소드) 호출**

- 문자열형 객체에서 제공하는 다양한 메소드(함수)를 점(.)으로 연결하여 호출

객체.메소드()

명령문	animal = 'elephant' print('총 개수:', animal.count('e')) 이 문자열에는 'e' 문자가 몇 개 있을까?
결과	총 개수: 2

문자열 형 객체
→ 보기엔 그냥 문자열

문자열 형 객체 안에 있는 함수
→ 보기엔 함수지만, 객체와 연결되어 있음

- 정보 수집

명령문	animal = 'elephant' print('앞쪽 찾기:', animal.find('e')) print('ep 찾기:', animal.find('ep')) print('뒤쪽 찾기:', animal.rfind('e')) print('위치:', animal.index('e')) print('el 시작:', animal.startswith('el'))
결과	앞쪽 e 찾기: 0 # 문자나 문자열이 처음 나오는 인덱스 값을 반환 ep 찾기: 2 뒤쪽 찾기: 2 # 문자나 문자열이 가장 나중에 나오는 인덱스 값을 반환 위치: 0 # find()의 기능과 동일, 찾는 내용이 없으면 에러 발생 el 시작: True # 해당 문자열로 시작하면 True, 그렇지 않으면 False 출력

- in
  - 특정 문자 또는 문자열이 해당 문자열에 존재하는지 여부를 판단

```
(문자 또는 문자열) (not) in 문자열
```

- **정보 수정**
  - 문자열의 내용을 수정할 수 있는 메소드(함수)

명령문	ai = 'python program' print('선택수정:', ai.replace('p', 'P')) print('소문자:', ai.lower()) print('대문자:', ai.upper()) print('swap대소문자:', ai.swapcase()) print('첫문자만 대문자:', ai.capitalize())
결과	선택수정: Python Program # 단어 교체 소문자: python program # 모든 문자들을 소문자로 변환 대문자: PYTHON PROGRAM # 모든 문자들을 대문자로 변환 swap대소문자: PYTHON PROGRAM # 모든 대/소문자를 역으로 변환 첫문자만 대문자: Python program # 문자열의 첫 문자만 대문자로 수정

- **정보 분할**
  - 문자열의 공백을 삭제할 수 있는 메소드

명령문	`animal = ' elephant '`
	`print('왼쪽 벗겨내기:', animal.lstrip())`
	`print('오른쪽 벗겨내기:', animal.rstrip())`
	`print('좌우 벗겨내기:', animal.strip())`

결과	왼쪽 벗겨내기: elephant # 문자열의 왼쪽편 공란(들)을 삭제 오른쪽 벗겨내기: elephant # 문자열의 오른쪽편 공란(들)을 삭제 좌우 벗겨내기: elephant # 문자열의 좌우의 공란(들)을 삭제

- choice()
  - 무작위로 문자를 출력

명령문	import random chars = ['한', '글', '우', '수'] print(random.choice(chars))
결과의 예	수

- shuffle()
  - 실제로 배열의 순서가 바뀌어 저장됨

명령문	import random chars = ['한', '글', '우', '수'] random.shuffle(chars) print(chars)
결과의 예	['우', '수', '한', '글'] # 4개의 글자가 무작위의 순으로 재배열 됨

문자열

문자열 표시 방법, 슬라이싱, 병합

문자열 함수 (method) – 길이, 정보 수집, 수정, 분할

문자열 무작위 선정, 무작위 섞기 (shuffle)

과제

Quiz 5.6, Quiz 5.15, Quiz 5.16

각각에 대해, “퀴즈, 코드, 실행 결과”를 정리하여 보고서의 형식을 따라 제출부탁드립니다.

예를 들어 1장 Quiz 5.6, 2장 Quiz 5.15, 3장 Quiz 5.16 으로 정리하고, 각 장의 소제목은 Quiz에서 요구하는 중간 질문들로 넣어주시면 됩니다. 추가로 결과를 더 넣어 주시고요.

문자열

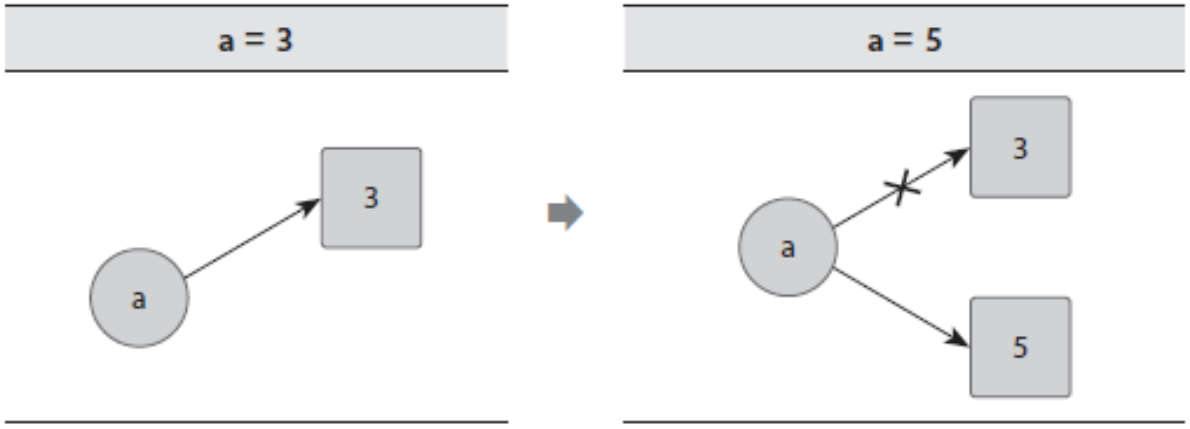
배열의 종류

- **배열형 자료구조**
  - 문자열형, 유니코드 문자열형, 리스트, 튜플, 바이트배열, xrange()
  - 한 개의 변수로 다수 개의 데이터를 저장해 두고 편리하게 접근
  - 어떻게? 위치 정보 (보통 인덱스라고 부르는...)

- mutable과 immutable

변경할 수 있는 데이터형(mutable)	변경할 수 없는 데이터형(immutable)
> 리스트형(list) > 사전형(dict) > 집합형(set)	> 숫자형: 정수형(int), 실수형(float) > 부울형(bool) > 문자열형(str) > 튜플형(tuple)

  - 정수형은 immutable



** 원래 있는 데이터를 보존해 두면서 변할 수 있는가?

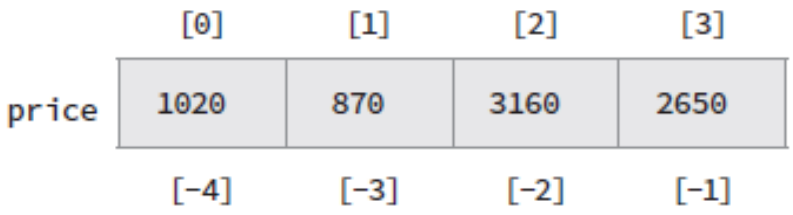
- **리스트**
  - 다수의 항목들을 입력 가능 (메모리가 허용하는 한…)
  - 대괄호([ ])를 사용하며 각각의 항목들은 콤마(,)로 구분
- **1차원 리스트**

	[0]	[1]	[2]	[3]
price	1020	870	3160	2650
	[-4]	[-3]	[-2]	[-1]

명령문	price = [1020, 870, 3160, 2650] fruits = ['사과', '오렌지', '포도', '복숭아'] print(price) print(fruits)
결과	[1020, 870, 3160, 2650] ['사과', '오렌지', '포도', '복숭아']

- 1차원 리스트의 개별 항목 접근

명령문	price = [1020, 870, 3160, 2650] fruits = ['사과', '오렌지', '포도', '복숭아'] print(price[1]) print(fruits[-1])
결과	870 복숭아



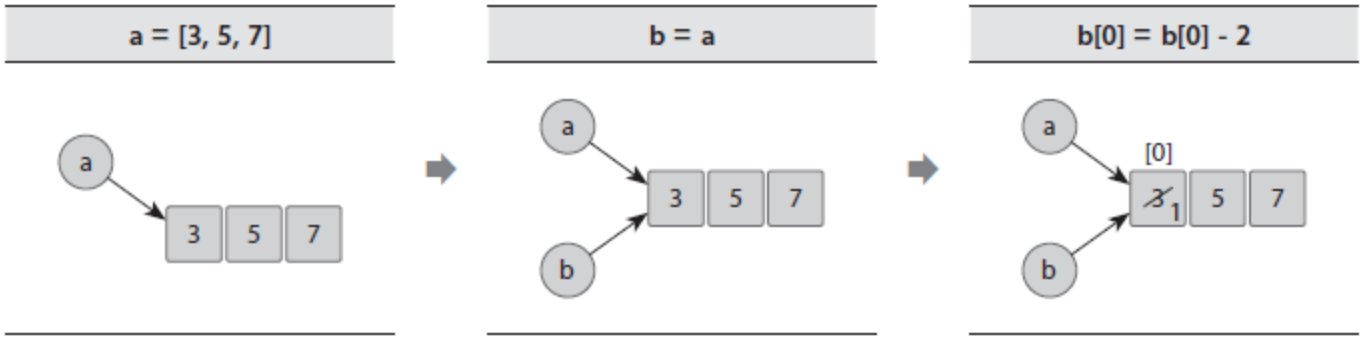
- **리스트의 복사**
  - a를 b에 복사하고 b의 index 0번에 저장된 값을 변경하면

명령문	a = [3, 5, 7] b = a b[0] = b[0] – 2 print('a=', a, 'b=', b)

  - a의 배열까지 수정됨 → 왜? 분명히 다른 변수인데? 왜?

```
a = [1, 5, 7] b = [1, 5, 7]
```

  - 같은 객체를 두 개의 다른 변수들이 가리킴



- **리스트의 슬라이싱**
  - 기본적으로 문자열의 슬라이싱과 동일 (문자열도 '문자'들이 들어간 '배열')
  - 리스트는 범용 구조 – 정수, 실수, 문자열 등 다양한 데이터를 저장
  - 구조:

[시작:끝+1:단계]	
명령문	price = [1020, 870, 3160, 365, 731] print(price[0:2]) # 인덱스 0에서 2 전까지 출력 print(price[0:4:2]) # 인덱스 0에서 4 전까지 2의 간격으로 출력 print(price[1::2]) # 인덱스 1에서 2의 간격으로 끝까지 출력
결과	[1020, 870] [1020, 3160] [870, 365]

리스트에서의 병합 및 삽입

- 복수 개 변수의 병합

명령문	x = 12.23
	y = 23.34
	packing = [x, y] # packing!
	type(packing)
	print('Packing:', packing)
	[c1, c2] = packing # unpacking!
	print('Unpacking:\nc1:', c1)
	print('c2:', c2)

결과	Packing: [12.23, 23.34] Unpacking: c1: 12.23 c2: 23.34

변수들을 하나의 리스트로 넣었습니다.

다시 꺼냈고요.

- 복수 개 리스트의 병합

명령문	fruits1 = ['사과', '오렌지', '포도'] fruits2 = ['복숭아', '키위'] allfruits = fruits1 + fruits2 print(allfruits)
결과	['사과', '오렌지', '포도', '복숭아', '키위']

+ 기호로 두 리스트를 합쳤습니다.

- append() 메소드
  - 1개의 인자를 입력으로 받아들임
  - 해당값을 리스트의 마지막에 추가로 붙여 넣을 수 있음

명령문	`prime = [2, 3, 5]`
	`prime.append(7)`
	`print(prime)`

결과	`[2, 3, 5, 7]`

  - 빈 리스트에 정수값을 1개씩 반복적으로 입력하는 예제

명령문	`a = list()  # list()는 빈 리스트 생성, a = []와 동일`
	`for i in range(5):`
	`  a.append(i)`
	`  print(a[i])`

결과	`0`
	`1`
	`2`
	`3`
	`4`

리스트에 원소 삽입

- insert() 메소드 → 원하는 위치에 개별 원소 삽입
  - 2개의 입력 인자 중 첫 번째는 인덱스 번호
  - 해당 인덱스 번호 직전의 위치에 두 번째 인자의 값을 입력

명령문	`fruits = ['사과', '오렌지', '포도']`
	`fruits.insert(1, '키위')`
	`print(fruits)`

결과	`['사과', '키위', '오렌지', '포도']`

** append()는 항상 리스트의 가장 마지막 위치에 원소를 삽입

- **리스트의 함축 (comprehension)**
  - 특정 리스트에 저장된 원소들이 있음
  - 이 리스트 내 모든 원소들에 대해 조건에 맞는 원소들만을 선택적으로 (새로운 리스트에) 추가

명령문	`mylist = [3, 5, 4, 9, 1, 8, 2, 1]`
	`newlist = [i for i in mylist if (i%2)==0]`
	`print(newlist)`

결과	`[4, 8, 2]`

- 예〉 `[i for i in (리스트명) if (조건식)]`
  ④ ② ① ③
- 리스트(①)의 원소들을 i(②)로 읽어 와서 조건식(③)에서 그 값을 테스트한 후 결과가 참이면 i(④)를 리스트에 입력

리스트의 항목 삭제

- **del 명령문으로 삭제**
  - 지정한 위치에 해당하는 항목을 삭제

명령문	`nums = [0, 1, 2, 3, 4, 5]`
	`del nums[1]`
	`print(nums)`

결과	`[0, 2, 3, 4, 5]`

  - 여러 개의 항목을 삭제하려면 슬라이스 기능을 사용

명령문	`nums = [0, 1, 2, 3, 4, 5]`
	`del nums[1:5]`
	`print(nums)`

결과	`[0, 5]`

pop() 함수로 삭제

- 리스트 객체에서 제공하는 삭제 기능

인자로 아무 것도 안주면, 제일 마지막 원소 삭제

명령문	nums = [1, 3, 5, 7, 9] nums.pop() # 9가 삭제되어 [1, 3, 5, 7]로 변경됨 nums.pop() # 7이 삭제되어 [1, 3, 5]로 변경됨
	print(nums) # 결과를 출력
결과	[1, 3, 5] # 남은 결과를 출력

- 입력값을 넣어주면 인덱스로 인식하여 해당 항목을 삭제

인자로 위치를 주면, 제일 마지막 원소 삭제

명령문	nums = [1, 3, 5, 7, 9] nums.pop(2) print(nums)
결과	[1, 3, 7, 9]

리스트의 항목 삭제

- 빈 리스트([])로 삭제

명령문	nums = [0, 1, 2, 3, 4, 5] nums[1:2] = [] print(nums)
결과	[0, 3, 4, 5]

** 해당 슬라이스에 빈 리스트를 넣어버리면, 그 부분이 사라짐

특정 원소의 존재 여부

- in
  - 문자열에 특정한 문자가 있는지 검사

명령문	word = 'orange' print('r' in word)
결과	True

- Not in
  - 예: 리스트 fruits에 특정한 과일이 없는지 검사

명령문	fruits = ['사과', '오렌지', '포도'] print('포도' not in fruits)
결과	False

원소의 반복

- 원소를 2번 반복하여 출력:

명령문	a = [1, 2, 3] b = a * 2 print(b)
결과	[1, 2, 3, 1, 2, 3]

- 각각의 원소에 2를 곱하기:

명령문	a = [1, 2, 3] for i in a: print(i * 2)
결과	2 4 6

리스트 내 원소의 갯수를 세보자

- len()
  - 원소의 개수를 조사

명령문	fruits = ['사과', '오렌지', '포도'] print(len(fruits))
결과	3

- count()
  - 특정 값의 개수를 측정

명령문	nums = [5, 7, 1, 3, 5, 7, 1, 3, 3, 5, 7, 9] print(nums.count(5))
결과	3

인덱스(리스트 내 원소 위치) 번호 찾기

- index()
  - array 리스트에서 x값이 처음 나오는 인덱스 번호를 반환

명령문 형식	array.index(x)

  - nums 리스트에서 3은 인덱스 2번과 5번에 위치하지만 index(3)은 첫 번째 인덱스를 반환

명령문	nums = [1, 2, 3, 4, 5, 3] print(nums.index(3))
결과	2

- sort()
  - 정렬 – 순서대로 줄을 세우자 (기본값은 작은 수 → 큰 수)

명령문	nums = [3, 5, 2, 1, 5, 3] nums.sort() print(nums)
결과	[1, 2, 3, 3, 5, 5]

- reverse()
  - 역순 – 이건 줄 세우는 것은 아니고, 들어 있는 순서를 뒤집는 것

명령문	fruits = ['사과', '오렌지', '포도', '복숭아'] fruits.reverse() print(fruits)
결과	['복숭아', '포도', '오렌지', '사과']

- 예: fruitdb 리스트

명령문	fruits = ['apple', 'orange', 'grape', 'orange'] fruitdb = [['apple', 1020], ['orange', 880], ['grape', 3160]]

- fruits: 1차원 리스트

fruits	'apple'	'orange'	'grape'	'orange'

- fruitdb: 2차원 리스트 (3×2 행렬)

fruitdb		
	'apple'	1020
	'orange'	880
	'grape'	3160

- 예: fruitdb 리스트
  - 특정한 위치에 저장된 내용을 읽어서 출력

명령문	`fruitdb = [['사과', 1020], ['오렌지', 880], ['포도', 3160]]`
	`print(fruitdb[1])`
	`print(fruitdb[1][0])`

결과	`['오렌지', 880]  # fruitdb[1]에 저장된 모든 내용을 출력` `오렌지  # fruitdb[1]에 저장된 내용 중 0번 인덱스의 내용만 출력`

fruits

'apple'	'orange'	'grape'	'orange'

fruitdb

'apple'	1020
'orange'	880
'grape'	3160

2차원 리스트

- 예: 2차원 리스트 record에 저장된 모든 리스트들의 1번 인덱스에 저장된 내용들만을 모아 새로운 index 리스트 생성

명령문	`record = [`
	`  [1, 2, 3],`
	`  [10, 20, 30],`
	`  [100, 200, 300]]`
	`index = [ary[1] for ary in record]`
	`print(index)`

결과	`[2, 20, 200]`

- 예: mylist에서 원소 개수가 2인 것들만 추출하여 newlist로 생성

명령문	`mylist = [[1, 2], [3, 4, 5], [6, 7]]`
	`newlist = [x for x in mylist if len(x)==2]`
	`print(newlist)`

결과	`[[1, 2], [6, 7]]`

Thank you so much!