String Dictionary Tuple

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Dec 16 – 20, 2019

Lab 4-3. Cities

- 지구 상에서 두 지점간 거리?
  - 지구는 둥그니까~
  - 직선 거리가 직선 거리가 아니다.
  - 구 위의 임의의 두 점의 거리는?

Radian 각도로 표현된 두 지점 사이 거리

$$(lat1, lon1), \qquad (lat2, lon2)$$

$$\cos\theta = \sin(lat1)\sin(lat2) + \cos(lat1)\cos(lat2)\cos(lon2 - lon1)$$

$$\theta = acos(\cos(\theta))$$

$$distance = r\theta$$

Lab 4-3. Cities

- 지구 상에서 두 지점간 거리?
  - 코드로 표현하는 방법?
  - math 라이브러리를 사용!
  - import math → math.pi, math.sin, math.cos, math.acos

$$(lat1, lon1), \qquad (lat2, lon2)$$

$$cos\,\theta = sin(lat1)sin(lat2) + cos(lat1)cos(lat2)cos(lon2 - lon1)$$

$$\theta = acos(cos(\theta))$$

$$distance = r\theta, \qquad 2\pi\, radian = 360°$$

Lab 4-3. Cities

- 사용자에게 두 위치를 받아서 거리를 출력하기

```
def radian(d)

def dist_between_loc(loc1, loc2)

def loc_dist()
```

Lab 4-3. Cities

- 회사에서 집까지 거리 구하기

```
$ python cities.py↲
첫 번째 위도 : 37.225432↲
첫 번째 경도 : 127.070405↲

두 번째 위도 : 37.450019↲
두 번째 경도 : 126.952517↲

거리 : 27060.502276595416
```

https://tinyurl.com/rn2q7xy

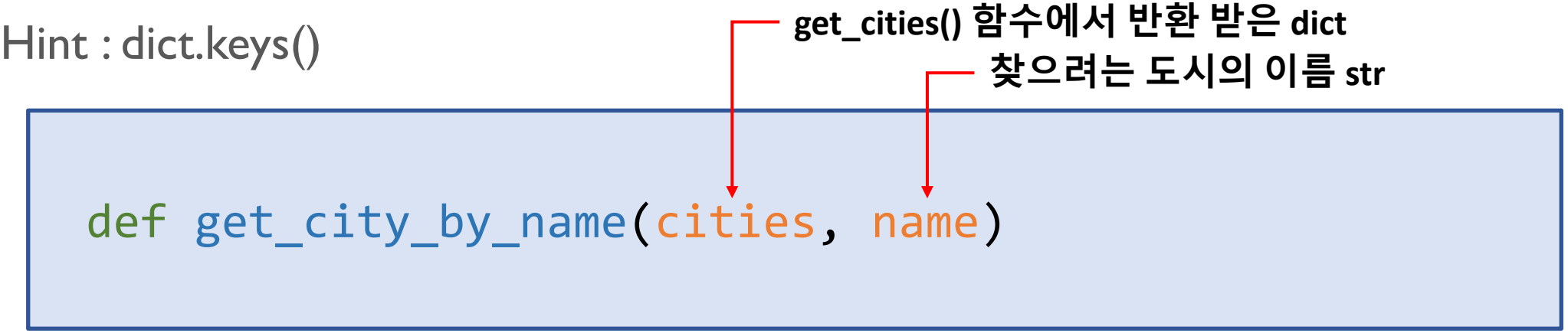
Lab 4-3. Cities

- 파일에서 도시 정보 읽기
  - 도시 정보 모아둔 파일 cities.txt
  - 도시의 이름, 도시의 나라\t도시의 위도\t도시의 경도
  - 파일을 읽어서 {key = 도시의 이름 : value = (도시의 위도, 도시의경도)} 의 dictionary 자료 만들기 → DB 구축하기

```
def get_cities()
```

Lab 4-3. Cities

- 구축한 DB 에서 필요한 정보 찾기
  - get_cities() 함수를 통해 .txt 에 있는 정보를 프로그램으로 가져옴
  - 내가 원하는 정보를 찾아내는 함수 만들기
  - 도시의 이름을 넣어서
    해당 도시가 있는 경우 위치 좌표 반환 (위도, 경도)
    없는 경우 None 반환
  - Hint : dict.keys()



Lab 4-3. Cities

- 사용자의 요구에 따라 거리 알려주기
  - 도시 정보 모아둔 파일 cities.txt 에 있는 도시 이름 두 개를 넣었을 경우 (순서에 상관 없이) 거리를 출력
  - 이전에 구현한 함수들을 활용
  - 단, get_cities() 함수는 전체 코드에서 단 한 번만 호출

```
def city_dist()
```

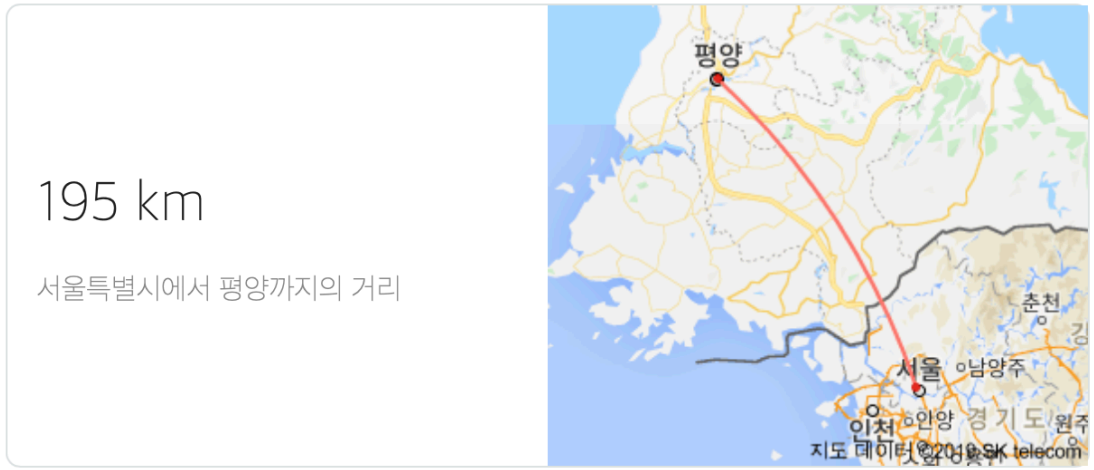
Lab 4-3. Cities

- 서울에서 평양 거리

```
$ python cities.py↲
1 번째 도시 이름 : Seoul↲
2 번째 도시 이름 : Pyeongyang↲
239024.51825296393

1 번째 도시 이름 : a↲
DB에 없는 도시 입니다.
2 번째 도시 이름 : done↲
프로그램을 종료합니다.
```

왜 오차가 발생할까요?



Advanced Lab

Lab 4-4. Differentiation

- 최종적으로 만들고자 하는 그림
  - We can do it!!!

```
assert d_dx("5x^2 + 2x + -7") == "10x + 2"
assert d_dx("x^4 + -7x") == "4x^3 + -7"
assert d_dx("1 + x + 2x + 3x^2") == "1 + 2 + 6x"
```

Lab 4-4. Differentiation

- 구체적인 제약조건
  - **다항함수 문자열**을 입력받았을 때, 그 미분함수를 **다항함수 문자열**로 출력하기

  - **다항함수 문자열**
    - 다항함수 문자열은 최소 1개 이상의 항으로 구성되어 있어야 한다.
    - 각각의 항은 덧셈(`'+'`) 기호로 구분되어 있으며, 덧셈 기호 사이에는 한 칸의 공백이 있다.
      - `'[first_term] + [second_term] + … + [last_term]'`

    - 항은 (정수) **계수**와 (음이 아닌 정수) **차수**로 표현할 수 있다.
      - 차수가 0인 경우, 이 항을 **상수항**이라 하며, 정수 계수를 있는 그대로 출력한다. (예: `'-1'`, `'0'`, `'1'`)
      - 차수가 1 이상이면, 정수 계수 뒤에 문자 x를 붙인다.
        추가로, 차수가 2 이상이면, 문자 x 뒤에 기호 ^와 차수를 붙인다. (예: `'-5x^2'`, `'0x'`, `'3x^4'`)
      - 차수가 1이상이고 계수의 절댓값이 1인 경우, 숫자 1을 생략한다. (예: `'-x'`, `'x^2'`, `'-x^3'`)

Lab 4-4. Differentiation

- 구체적인 제약조건
  - 미분함수
    - 어느 임의의 다항함수의 미분함수는 여러 **다항함수 문자열**로 표현 가능하다.

  - 예 : equation = `'x^2 + 2x + 1'`
    - `'2x + 2 + 0'`
    - `'2x + 2'`
    - `'2 + 2x'`
    - `'1 + x + 1 + x + 0x^2 + 0x^3'`

  - 채점 코드는 항의 순서, 계수가 0인 항, 동류항 묶기 등등을 신경쓰지 않는다!
  - 신경 쓸만한 부분은, 다항함수 문자열은 **최소 1개의 항**이 필요하다는 점
    - 빈 문자열 `''`을 출력하면 안 된다. 최소한 `'0'`은 출력해야 함

Lab 4-4. Differentiation

- `def print_term(factor, degree):`
  - *factor* : 항의 계수 (정수)
    - 음수인 경우?
    - 절댓값이 1인 경우?
  - *degree* : 항의 차수 (0 이상의 정수)
    - 0인 경우?
    - 1인 경우?

```
assert print_term(0, 2) == "0x^2"
assert print_term(-1, 1) == "-x"
assert print_term(5, 0) == "5"
```

Lab 4-4. Differentiation

- `def print_equation(terms):`
  - *terms* : list
    - 각 원소는 [factor : int, degree : int]의 형태
  - Expected output
    - `' + '` 문자를 기준으로 합침 : `string.join()`?

```
assert print_equation(
  [[0, 2], [-1, 1], [5, 0]]
) == "0x^2 + -x + 5"
```

Lab 4-4. Differentiation

- `def parse_term(term_str):`
  - `print_term()`의 역함수 꼴
  - *term_str* : 항을 문자열로 표현

  - Expected outputs
    - list : [factor : int, degree : int]

```
assert parse_term("0x^2") == [0, 2]
assert parse_term("-x") == [-1, 1]
assert parse_term("5") == [5, 0]
```

Lab 4-4. Differentiation

- `def parse_equation(equation):`
  - *equation* : str
    - `' + '` 문자를 기준으로 쪼개기 : `string.split()`?
  - Expected output

```
assert parse_equation("0x^2 + -x + 5") == \
    [[0, 2], [-1, 1], [5, 0]]
```

Lab 4-4. Differentiation

- `def d_dx_as_terms(terms):`
  - *terms* : list of list
    - [[1항 계수, 1항 차수], [2항 계수, 2항 차수], …, [3항 계수, 3항 차수]]
  - Expected output
    - 형식은 *terms*와 동일
    - 가능한 답은 여러가지가 될 수 있음!
    - 신경 쓸 부분 (**다항함수 조건**)
      - 출력된 계수는 정수인가
      - 출력된 차수는 0 이상의 정수인가
      - 항의 개수는 1 이상인가
      - 미분 법칙에 잘 맞는가 : $(\frac{d}{dx}ax^n = (an)x^{n-1}, \frac{d}{dx}a = 0$ and $\frac{d}{dx}(f+g) = \frac{d}{dx}f + \frac{d}{dx}g)$

Lab 4-4. Differentiation

- `def d_dx(equation):`
  - *equation* : str
  - Expected output : str

  - 지금껏 구현했던 함수들을 총동원하기
  - 3줄로 간결하게 표현할 수 있음