AI Programming

Lecture 10

Assignment 8 Solution

```
import random

numRows = int(input("행 개수를 입력하시오 : "))
numCols = int(input("열 개수를 입력하시오 : "))

myList = []
newList = []
tempList = []

for i in range(numRows):
    for k in range(numCols):
        tempList.append(random.randint(1, 10))
    myList.append(tempList)
    tempList = []
```

```
print("₩n원본 행렬")
for i in range(numRows):
    for k in range(numCols):
        print("%3d" % myList[i][k], end="")
    print("")


for i in range(numCols):
    for k in range(numRows):
        tempList.append(myList[k][i])
    newList.append(tempList)
    tempList = []

print("₩n전치 행렬")
for i in range(numCols):
    for k in range(numRows):
        print("%3d" % newList[i][k], end="")
    print("")
```

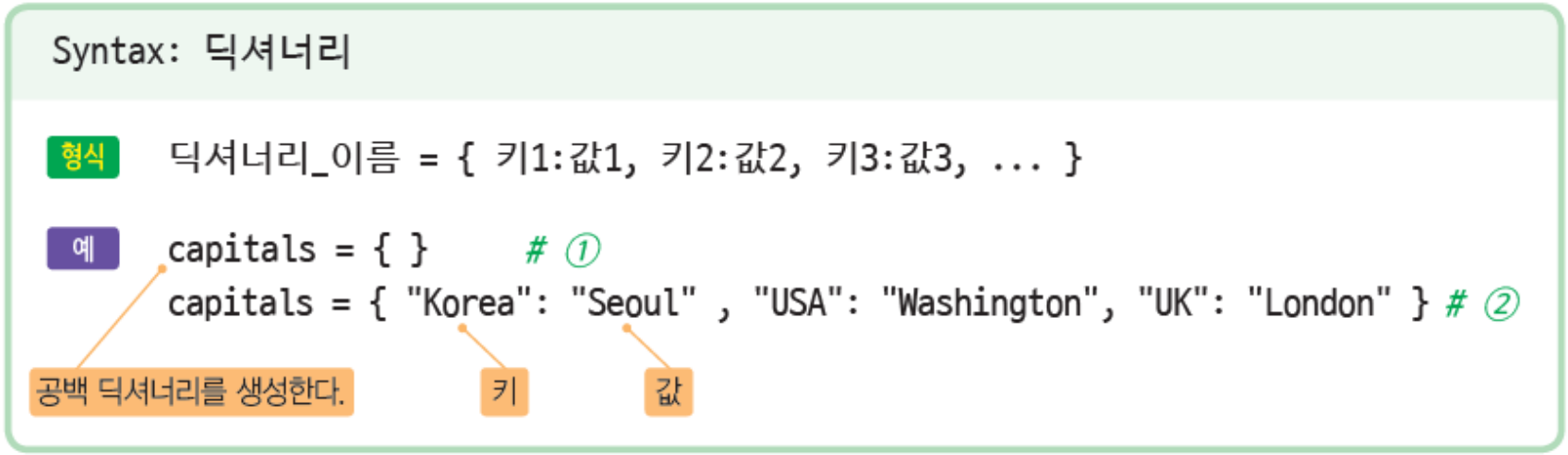
Preview

- **Ch. 7 리스트, 튜플, 딕셔너리**
  - 7.5 딕셔너리
  - 7.6 리스트, 튜플, 딕셔너리의 심화내용

7.5 딕셔너리

Dictionary

- **Dictionary**
  - Pairs of key and value (no priority)
  - `myDictionary = {key1:value1, key2:value2, ...}`



Dictionary

- **Applications**
  - Database

키	값
학번	1000
이름	홍길동
학과	컴퓨터학과

```
student1 = {'학번' : 1000, '이름': '홍길동', '학과': '컴퓨터학과'}
student1
```

출력 결과

```
{'학번': 1000, '이름': '홍길동', '학과': '컴퓨터학과'}
```

Dictionary

연산	설명
`d = dict()`	공백 딕셔너리를 생성한다.
$d=\{k_1:v_1,k_2:v_2, \ldots,k_n:v_n\}$	초기값으로 딕셔너리를 생성한다.
`len(d)`	딕셔너리에 저장된 항목의 개수를 반환한다.
`k in d`	k가 딕셔너리 d 안에 있는지 여부를 반환한다.
`k not in d`	k가 딕셔너리 d 안에 없으면 True를 반환한다.
`d[key] = value`	딕셔너리에 키와 값을 저장한다.
`v = d[key]`	딕셔너리에서 key에 해당되는 값을 반환한다.
`d.get(key, default)`	주어진 키를 가지고 값을 찾는다. 만약 없으면 default 값이 반환된다.
`d.pop(key)`	항목을 삭제한다.
`d.values()`	딕셔너리 안의 모든 값의 시퀀스를 반환한다.
`d.keys()`	딕셔너리 안의 모든 키의 시퀀스를 반환한다.
`d.items()`	딕셔너리 안의 모든 (키, 값)을 반환한다.

Dictionary

- **Indexing (keys)**
  - `myDictionary[key], myDictionary.get(key)`

```
student1['학번']
student1['이름']
student1['학과']
```

출력 결과

```
2000
'홍길동'
'파이썬학과'
```

```
student1.get('이름')
```

출력 결과

```
'홍길동'
```

Dictionary

- myDictionary[key] **vs.** myDictionary.get(key)

```
>>> student1={'학번':2000, '이름':'홍길동', '학과':'파이썬학과'}
>>> student1['주소']
Traceback (most recent call last):
  File "<pyshell#55>", line 1, in <module>
    student1['주소']
KeyError: '주소'
>>> student1.get('주소')
>>>
```

Dictionary

- **Insert**
  - `myDictionary[key] = value`

```
student1['연락처'] = '010-1111-2222'
student1
```

출력 결과

```
{'학번': 1000, '이름': '홍길동', '학과': '컴퓨터학과', '연락처': '010-1111-2222'}
```

Dictionary

- **Replace**
  - `myDictionary[key] = value`

```
student1['학과'] = '파이썬학과'
student1
```

출력 결과

```
{'학번': 1000, '이름': '홍길동', '학과': '파이썬학과', '연락처': '010-1111-2222'}
```

Dictionary

- **Delete**
  - `del(myDictionary[key])`
  - `myDictionary.pop(key)`

```
del(student1['학과'])
student1
```

출력 결과

```
{'학번': 1000, '이름': '홍길동', '연락처': '010-1111-2222'}
```

Dictionary

- **Identical keys**
  - 마지막에 입력한 value 사용

```
student1 = {'학번': 1000, '이름': '홍길동', '학과': '파이썬학과', '학번': 2000}
student1
```

출력 결과

```
{'학번': 2000, '이름': '홍길동', '학과': '파이썬학과'}
```

Dictionary

- `myDictionary.keys()`
  - Return all keys in `myDictionary`.

```
student1.keys()
```

출력 결과

```
dict_keys(['학번', '이름', '학과'])
```

```
list(student1.keys())
```

출력 결과

```
['학번', '이름', '학과']
```

Dictionary

- `myDictionary.values()`
  - Return all values in `myDictionary`.

```
student1.values()
```

출력 결과

```
dict_values([2000, '홍길동', '파이썬학과'])
```

Dictionary

- `myDictionary.items()`
  - Return a tuple of `myDictionary`.

```
student1.items()
```

출력 결과

```
dict_items([('학번', 2000), ('이름', '홍길동'), ('학과', '파이썬학과')])
```

Dictionary

- **Finding keys**
  - `key in myDictionary`

```
'이름' in student1
'주소' in student1
```

출력 결과

```
True
False
```

Dictionary

- **Exercise**

```
1  singer = {}
2
3  singer['이름'] = '트와이스'
4  singer['구성원 수'] = 9
5  singer['데뷔'] = '서바이벌 식스틴'
6  singer['대표곡'] = 'SIGNAL'
7
8  for k in singer.keys() :
9      print('%s --> %s' % (k, singer[k]))
```

출력 결과

```
이름 --> 트와이스
구성원 수 --> 9
데뷔 --> 서바이벌 식스틴
대표곡 --> SIGNAL
```

7.6 리스트, 튜플, 딕셔너리의 심화 내용

7.6.1 세트

Set

- **Set**
  - Dictionary only containing keys
  - No priority, no duplication

```
mySet1 = {1, 2, 3, 3, 3, 4}
mySet1
```

출력 결과

```
{1, 2, 3, 4}
```

Set

- **Set**
  - `set()`: convert list, tuple, and dictionary to a set

```
salesList = ['삼각김밥', '바나나', '도시락', '삼각김밥', '삼각김밥', '도시락', '삼각김밥']
set(salesList)
```

출력 결과

```
{'도시락', '바나나', '삼각김밥'}
```

Set

- **Operations**
  - Intersection, union, difference, symmetric difference
  - Operators: `&, |, -, ^`

```
mySet1 = {1, 2, 3, 4, 5}
mySet2 = {4, 5, 6, 7}
mySet1 & mySet2      # 교집합
mySet1 | mySet2      # 합집합
mySet1 - mySet2      # 차집합
mySet1 ^ mySet2      # 대칭 차집합
```

출력 결과

```
{4, 5}
{1, 2, 3, 4, 5, 6, 7}
{1, 2, 3}
{1, 2, 3, 6, 7}
```

Set

- **Operations**
  - Intersection, union, difference,…

```
mySet1.intersection(mySet2)             # 교집합
mySet1.union(mySet2)                    # 합집합
mySet1.difference(mySet2)               # 차집합
mySet1.symmetric_difference(mySet2)     # 대칭 차집합
```

7.6 리스트, 튜플, 딕셔너리의 심화 내용

7.6.2 컴프리헨션

Comprehension

- **Comprehension (함축)**

Syntax: 리스트 함축

형식 [수식 for (변수 in 리스트) if (조건)]

예 squares = [x*x for x in range(10)]

- 새로운 리스트
- 출력식으로 새로운 리스트의 요소가 된다.
- 입력 리스트에 있는 요소 x에 대하여
- 입력 리스트

```
squares = []

for x in range(10):
        squares.append(x*x)
```

Comprehension

- **Practice**

```
numList = []
for num in range(1, 6) :
    numList.append(num)
numList
```

출력 결과

```
[1, 2, 3, 4, 5]
```

```
numList = [num for num in range(1, 6)]
numList
```

출력 결과

```
[1, 2, 3, 4, 5]
```

Comprehension

- **Practice**

```
numList = [num * num for num in range(1, 6)]
numList
```

출력 결과

```
[1, 4, 9, 16, 25]
```

Comprehension

- **Comprehension & If**

```
squares = [ x*x for x in range(10) if x % 2 == 0 ]
```

출력식 / 입력 리스트 / 조건

일반 for 루프

```
squares = [ ]
for x in range(10):
        if x%2 == 0 :
                squares.append(x*x)
```

리스트 함축

```
squares = [ x*x for x in range(10) if x%2 == 0 ]
```

Comprehension

- **Practice**

```
numList = [num for num in range(1, 21) if num % 3 == 0]
numList
```

출력 결과

```
[3, 6, 9, 12, 15, 18]
```

Comprehension

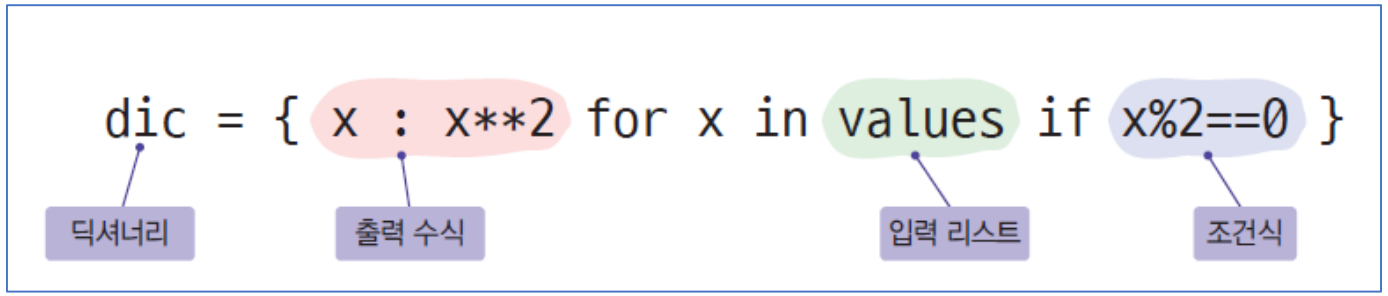
- **Comprehension of 2D list**

```
a = [[i for i in range(3)] for _ in range(4)]
b = [[row[i] for row in a] for i in range(3)]
print("원본 행렬: ", a)
print("전치 행렬: ", b)
```

```
원본 행렬:  [[0, 1, 2], [0, 1, 2], [0, 1, 2], [0, 1, 2]]
전치 행렬:  [[0, 0, 0, 0], [1, 1, 1, 1], [2, 2, 2, 2]]
```

Comprehension

- **Comprehension of dictionary**




```
dict = {i : i**2 for i in range(1, 7) if i%2 == 0}
print(dict)
```

```
{2: 4, 4: 16, 6: 36}
```

7.6 리스트, 튜플, 딕셔너리의 심화 내용

7.6.3 동시에 여러 리스트에 접근

Advanced List

- zip()

```
foods = ['떡볶이', '짜장면', '라면', '피자', '맥주', '치킨', '삼겹살']
sides = ['오뎅', '단무지', '김치']
tupList = list(zip(foods, sides))
dic = dict(zip(foods, sides))
tupList
dic
```

출력 결과

```
[('떡볶이', '오뎅'), ('짜장면', '단무지'), ('라면', '김치')]
{'떡볶이': '오뎅', '짜장면': '단무지', '라면': '김치'}
```

Advanced List

- zip()

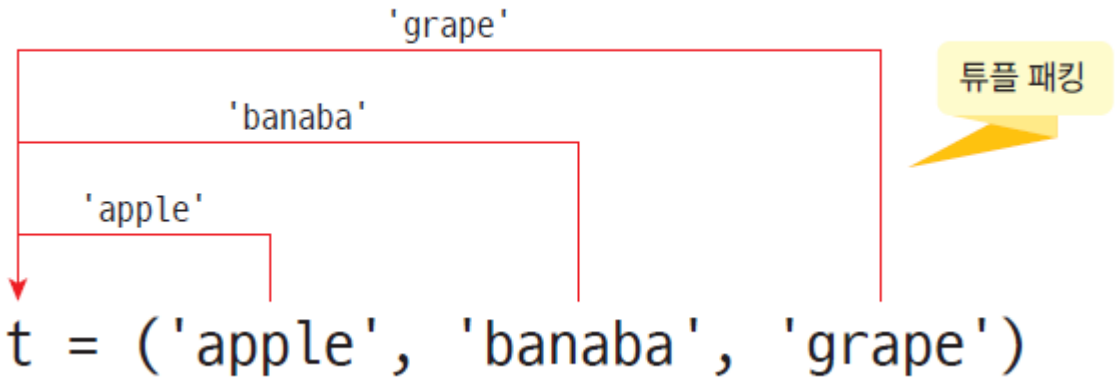
```
foods = ['떡볶이', '짜장면', '라면', '피자', '맥주', '치킨', '삼겹살']
sides = ['오뎅', '단무지', '김치']
for food, side in zip(foods, sides) :
    print(food, ' --> ', side)
```

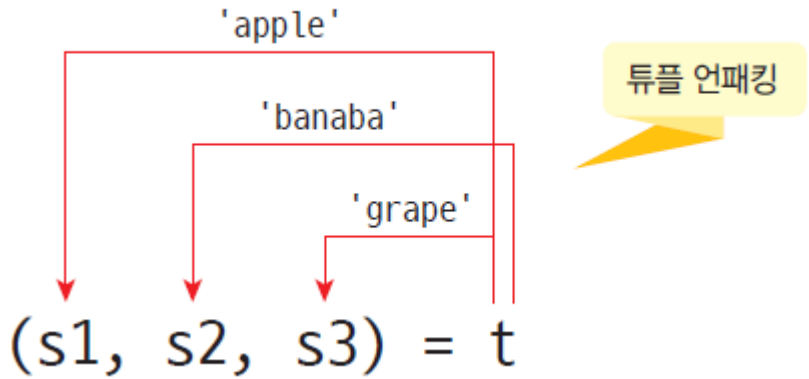
출력 결과

```
떡볶이 --> 오뎅
짜장면 --> 단무지
라면 --> 김치
```

Advanced List

- **Revisit: Unpacking**
  - Tuple에 저장된 데이터를 풀어서 개별 변수에 저장





Summary

- **Tuple**
- **Dictionary**
  - `myDictionary = {key1:value1, key2:value2, ...}`
- **Set**
- **Comprehension**
  - `myList = [equation for i in range(start, end+1, step) if condition]`

Assignment 9

- **연락처 프로그램**
  - Dictionary 활용
    - key: 이름, value: 전화번호
  - 기능
    - 1. 연락처 추가
    - 2. 연락처 삭제
    - 3. 연락처 검색
    - 4. 연락처 출력
    - 5. 종료 (break)

```
1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 1
이름: 전요한
전화번호: 010-1111-4444

1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 1
이름: 강인구
전화번호: 010-2222-3333
```

Assignment 9

- **4. 연락처 출력**

```
1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 4
전요한의 전화번호: 010-1111-4444
강인구의 전화번호: 010-2222-3333
```

Assignment 9

- **3. 연락처 검색**
  - 이름을 기반으로 검색
    - 연락처에 존재하는 이름이면 번호 출력
    - 없는 이름이면 에러메시지 출력

```
1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 3
이름: 강인구
강인구의 전화번호: 010-2222-3333

1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 3
이름: 변기태
Error: 주소록에 없는 이름!
```

Assignment 9

- **2. 연락처 삭제**
  - 이름을 기반으로 삭제
    - 존재하지 않는 이름이면 에러메시지 출력

1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 2
이름: 변기태
Error: 주소록에 없는 이름!

1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 2
이름: 전요한

1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 4
강인구의 전화번호: 010-2222-3333