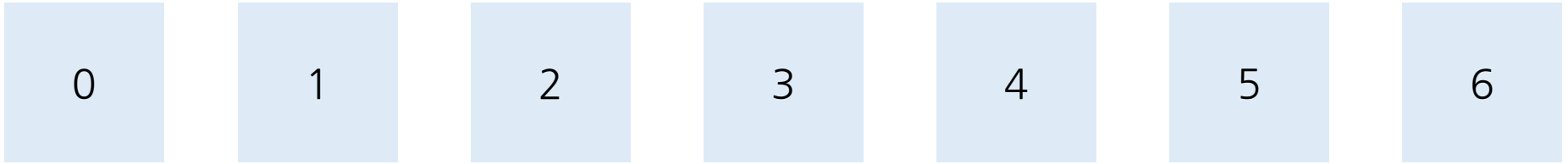
3주차: 리스트, 튜플, 세트, 딕셔너리

커리큘럼

1. 변수, 입출력
2. 조건문, 반복문
3. **리스트, 튜플, 세트, 딕셔너리**
4. 함수
5. class
6. 알고리즘 입문, 그리디 알고리즘
7. 재귀함수
8. 탐색
9. DP(Dynamic Programming)

시퀀스 자료형



✓ 시퀀스(sequence) : 연속, 배열하다

✓ 인덱스(index): 위치; 0,1,2,...

✓ 종류

- 문자열 str( )
- range(start, stop, step)
- 리스트 list( )
- 튜플 tuple( )
- 세트 set( )
- 딕셔너리 dict( )

시퀀스 공통 함수, 연산자

함수, 연산자	의미	사용 예시	결과
len	길이 계산	len([1,2,3])	3
+	2개의 시퀀스 연결	[1,2]+[3,4,5]	[1,2,3,4,5]
*	반복	['Welcome']*2	['Welcome','Welcome']
in	소속	3 in [1,2,3]	True
not in	소속하지 않음	5 not in [1,2,3]	True
[]	인덱스	myList[1]	myList 1번째 요소
min	가장 작은 요소	min([1,2,3])	1
max	가장 큰 요소	max([1,2,3])	3
for 루프	반복	for x in [1,2,3]; print(x)	1 2 3

인덱싱, 슬라이싱

```
>>> mylist = [1,2,3,4,5]
>>> mylist[-1]
5
>>> mylist[-2]
4
>>> mylist[1:3]
[2,3]
>>> mylist[1:]
[2,3,4,5]
>>> mylist[:3]
[1,2,3]
>>> mylist[:]
[1,2,3,4,5]
```

리스트

- ✓ 여러 개의 자료를 담을 수 있는 시퀀스 자료형의 기본
- ✓ 요소를 변경할 수 있다.(Mutable)
- ✓ 선언 : [ ] 사용, list( )
- ✓ 리스트 안에는 다양한 자료형이 들어갈 수 있음

ex)

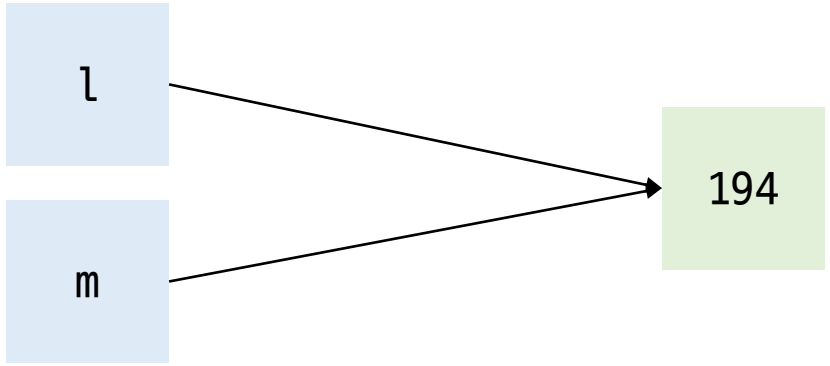
```
list_1=[]
list_2=list()
list_3=[1]
list_4=[1,2,3]
list_5=[1,1.1,'hello']
list_6=[1,2,[3,4]]
```

리스트

함수	의미	사용 예시(l=[1,3,2])	결과
l.append(x)	추가(항)	l.append(4)	[1,3,2,4]
l.insert(i,x)	i 위치에 삽입	l.append(1,4)	[1,4,3,2]
l.extend(<list>)	추가(리스트)	l.extend([4,5])	[1,3,2,4,5]
l.remove(x)	x가 처음 나타나는 곳 제거	l.remove(3)	[1,2]
l.pop(i)	i번째 항 제거 및 출력	print(l.pop(1))	3 [1,2]
del l[i]	i번째 항 제거	dek l[1]	[1,2]
l.clear()	모든 항목 삭제	l.clear	[]
l.reverse()	항목 순서 역으로 나열	l.reverse()	[2,3,1]
l.sort()	오름차순 정렬	l.sort()	[1,2,3]

리스트

함수	의미	사용 예시(l=[1,3,2])	결과
sorted(<list>)	정렬, 기존 것은 그대로	m=sorted(l)	[1,2,3]
l.count(x)	x의 갯수	l.count(1)	1
l.index(x)	x가 첫번째 등장하는 위치	l.index(1)	0
l.copy()	복사	m=l.copy()	[1,3,2]
copy.deepcopy(<list>)	복사(+import copy)	m=copy.deepcopy(l)	[1,3,2]



얕은 복사(Shallow copy)

깊은 복사(deep copy)

리스트

리스트 함축

```
expression for i in old_list if filter(i)
```

```
>>> l = [3,4,5]
>>> m = [x*2 for x in l if x!=4]
>>> m
[6,10]
```

```
>>> l = [i for i in range(10) if i%2!=0]
>>> l
[1,3,5,7,9]
```

이중 리스트

```
l=[[1,2,3],[4,5,6]]

row=len(l)
col=len(l[0])

for r in range(row):
    for c in range(col):
        print(s[r][c], end=‘ ’)
    print()
```

버블 정렬

✓ 앞에서 연속된 두 값을 비교해 나가다 보면, 첫 번째 순환 때 가장 큰 값이 가장 뒤로 이동하게 되면서 자동으로 정렬이 되는 알고리즘

✓ 시간이 가장 오래 걸림

[3,2,4,1]
[2,3,4,1]
[2,3,4,1]
[2,3,1,4]
[2,3,1,4]
[2,1,3,4]
[1,2,3,4]
[1,2,3,4]

선택 정렬

- ✓ 가장 기초적인 정렬로, 해당 리스트 내 가장 작은 값을 찾아서 그걸 앞으로 하나씩 옮기는 과정을 반복하는 알고리즘

[3,1,4,1,5,9,2,6]
[1,3,4,1,5,9,2,6]
[1,1,4,3,5,9,2,6]
[1,1,2,3,5,9,4,6]
[1,1,2,3,5,9,4,6]
[1,1,2,3,4,9,5,6]
[1,1,2,3,4,5,9,6]
[1,1,2,3,4,5,6,9]
[1,1,2,3,4,5,6,9]

삽입 정렬

- ✓ 첫번째 항목부터 시작해서 그 앞의 자료들과 비교해서 삽입할 위치를 정하고, 그 위치 뒤쪽의 자료들을 모두 한 칸 뒤로 이동해서 해당 항목을 삽입하는 알고리즘

[3,1,4,1,5,9,2,6]
[1,3,4,1,5,9,2,6]
[1,3,4,1,5,9,2,6]
[1,1,3,4,5,9,2,6]
[1,1,3,4,5,9,2,6]
[1,1,3,4,5,9,2,6]
[1,1,2,3,4,5,9,6]
[1,1,2,3,4,5,6,9]

튜플

- ✓ 여러 개의 자료를 담을 수 있는 시퀀스 자료형의 기본
- ✓ 요소를 변경할 수 없다.(immutable)
- ✓ 선언 : ( ), tuple( )
- ✓ 값을 변경 시 에러 발생

문자열(추가)

- ✓ .split(): 어떤 문자열을 괄호 안의 단위로 나누어서 리스트로 저장하는 함수

```
>>> ‘hello world’.split()
[‘hello’, ‘world’]
>>> ‘1,2,3,4,5’.split()
[‘1’, ‘2’, ‘3’, ‘4’, ‘5’]
```

- ✓ map(func, *iterables): 함수를(func) 순회 가능한 데이터(*iterables)에 적용하는 함수

- ✓ 순회 가능한 데이터? set, list, dict, …

```
>>> a,b = map(int,input(),split())
3 5
>>> [3,5]
[3, 5]
>>> l = list(map(int,[‘1’, ‘2’]))
>>> l
[1, 2]
```

세트

- ✓ 순서가 존재하지 않아 변경에 의미가 없다.
- ✓ 선언 : { }, set( )
- ✓ 교집합, 합집합, 차집합 - 집합의 연산
- ✓ **중복을 제거, 색인 불가능**

```
>>> set([1,1,3,4,6,2,2])
{1,2,3,4,6}
```

세트

함수, 연산자	의미	사용 예시 s1={1,2,3,4} s2={3,4,5,6}	결과
s1.add(x)	x 추가	s1.add(5)	{1,2,3,4,5}
s1.remove(x)	x 제거	s1.remove(1)	{2,3,4}
s1.update(<list>)	list 항목 추가	s1.update([5,6])	{1,2,3,4,5,6}
s1.copy()	복사	s3=s1.copy()	{1,2,3,4}
&,intersection()	교집합	s1&s2 s1.intersection(s2)	{3,4}
¦,union()	합집합	s1¦s2 s1.union(s2)	{1,2,3,4,5,6}
-,difference()	차집합	s1-s2 s1.difference(s2)	{1,2}

딕셔너리

- ✓ 요소를 변경할 수 있다.(Mutable)
- ✓ 선언 : { <key> : <value>}, dict( )
- ✓ key-value pair: 정수 이외 자료를 입력해서 색인을 사용할 수 있음

```
>>> dic1={‘Cho’ : 85, ‘Kim’ : 95, ‘Park’ : 90}
>>> dic1[‘Kim’]
95
```

딕셔너리

```
dic1={'Cho' : 85, 'Kim' : 95, 'Park' : 90}
```

연산자	의미
<key> in dic1	key 유무 True/False
dic1.keys()	key를 리스트로 반환
dic1.values()	value를 리스트로 반환
dic1.items()	(key, value) 반환
dic1[<key>] = value	새로운 key, value 추가
del dic1[<key>]	key에 해당하는 pair 제거
dic1.get(<key>,<default>)	key가 존재하면 value, 존재하지 않으면 default
dic1.clear()	모든 pair 제거
for key in dic1	key를 변수로 반복문

딕셔너리

✓ enumerate(iterable, start=0): index와 value를 분리, for문과 list 조합에 사용

```
>>> for index,value in enumerate(['a', 'b', 'c']):
        print(index, value)
0 a
1 b
2 c
```

✓ items(): key와 value를 분리, for문과 dict 조합에 사용

```
>>> for key,value in dic1.items():
        print(key, value)
Cho 85
Kim 95
Park 90
```


과제

3-1

버블 정렬을 구현하는 문제입니다.

단, sort, sorted 등 정렬 함수 사용금지.

입력 예시	출력 예시
3 1 4 1 5 9 2 6	1 1 2 3 4 5 6 9

3-2

선택 정렬을 구현하는 문제입니다.

단, sort, sorted 등 정렬 함수 사용금지.

```
3 1 4 1 5 9 2 6
```

입력 예시

```
1 1 2 3 4 5 6 9
```

출력 예시

3-3

삽입 정렬을 구현하는 문제입니다.

단, sort, sorted 등 정렬 함수 사용금지.

입력 예시	출력 예시
3 1 4 1 5 9 2 6	1 1 2 3 4 5 6 9