S05. 데이터구조, Data Structure

■ 리스트형, List Type

```
---------------------------------------------------------------------------
# 리스트형 선언 및 색인
pockets = [ 4, 6, 1, 9 ]        # 4개의 숫자형을 가진 리스트형 선언
pockets



---------------------------------------------------------------------------
# 변수의 타입 확인
type(pockets)



---------------------------------------------------------------------------
pockets[0]      # 첫번째 값 확인



---------------------------------------------------------------------------
pockets[-1]     # 마지막 값 확인



---------------------------------------------------------------------------
# [0] [1] [2] [3]
print( pockets[0] )
print( pockets[1] )
print( pockets[2] )
print( pockets[3] )
```

```
--------------------------------------------------------------------------------
# IndexError : list index out of range
print( pockets[4] )



--------------------------------------------------------------------------------
# 리스트형의 길이 확인
len(pockets)



--------------------------------------------------------------------------------
# 리스트형 데이터 변경하기
pockets[0] = 5
pockets



--------------------------------------------------------------------------------
# 리스트항목 추가하기 : append() 함수
pockets.append(7)
pockets



--------------------------------------------------------------------------------
# 리스트항목 제거하기 : remove() 함수
pockets.remove(1)  # 특정값 제거
pockets
```

```
# 리스트항목 삽입하기 :  insert() 함수
pockets.insert(1,3) # 두번째 색인(1)에 3 삽입
pockets
```

```
# 리스트항목 추출하기 : pop() 함수
pockets.pop(3)       # 4번째 색인 값 반환 후 제거
pockets
```

```
# 리스트형 데이터 자르기
pockets[1:3]       # pockets[1]부터 pockets[2]까지 자르기
```

```
pockets[:3]
```

```
pockets[-2:]
```

```
# 리스트형 복사하기
pockets_copy = pockets    # 신규변수생성 및 기존리스트 대입
pockets_copy.append(1)    # 신규변수에 1추가
pockets_copy
```

call by reference방식으로 리스트의 주소값을 복사해준다.

```
print(id(pockets))
print(id(pockets_copy))
```

```
pockets_real_copy = pockets[:]    # 신규변수생성 및 pockets 복사
pockets_real_copy.append(9)       # 신규변수에 9추가
print(id(pockets_real_copy))
print(id(pockets))
```

처음부터 끝까지 모든 데이터 값을 복사한다.(call by value)

```
# 리스트형 데이터 합치기 & 확장하기
a = [1, 2, 3]
b = [4, 5, 6]
c = a + b

# 리스트 값 확인
print('a = ', a)
print('b = ', b)
print('c = ', c)

# 객체식별자 확인
print('id(a) = ', id(a))
print('id(b) = ', id(b))
print('id(c) = ', id(c))
```

#데이터 확장

```
a.extend(b)
print('a = ', a)
print('id(a) = ', id(a))
```

```
# 리스트삭제를 위한 del() 함수
print('a = ', a)
del a[0]
print('a = ', a)
del a[1:3]
print('a = ', a)
del a[:]
print('a = ', a)
```

```
del a
print('a = ', a)
```

```
# 리스트형의 다양한 쓰임새
programming = [ 'Python', 'Java', 'C++' ]
print( programming )
```

```
--------------------------------------------------------------------------------
programming.append(100)       # 리스트에 숫자형 100 넣기
programming.append(3.14)      # 리스트에 실수형 3.14 추가
print( programming )



--------------------------------------------------------------------------------
# 3x3 2차원 행렬을 중첩리스트로 선언
nest = [ [1, 2, 3,], [4, 5, 6], [7, 8, 9] ]

print( nest )
print( nest[0] )
print( nest[0][0] )



--------------------------------------------------------------------------------
word = '파이썬 문자열 색인'
print( word )
print( word[0] )
print( word[-1] )



--------------------------------------------------------------------------------
# IndexError : string index out of range
print( word[10] )
```

다중리스트 선

```
print( word[:3] )

print( word[4:] )

print( word[-2:] )
```

■ 튜플형, Tuple Type

```
# 튜플 생성
movie = '슈퍼맨II', 1980, '배트맨', 1989

print ( movie )
print ( movie[1] )
print ( movie[-2:] )
```

```
# 튜플값 변경 시도 (형오류발생)
movie[1] =  1982
```

튜플은 immutable하기때문에 값을 변경할수 없다.

```
movie_list = list(movie)      # 튜플 -> 리스트형 변환

print(type(movie_list))       # 데이터형 확인
print(movie_list )            # 데이터값 확인 (대괄호 기호 확인)
print(tuple(movie_list))      # 리스트 -> 튜플형 변환(소괄호 기호 확인)
```

객체를 convert시킨다.

■ 세트형, Set Type

```
# 세트형 생성
lang = { 'Java', 'Java', 'Python', 'C++', 'Python' }

print( lang )                    # 세트형값 확인 (중복 제거 확인)
print( 'Java' in lang )          # 항목 존재 유무 확인
print( 'javascript' in lang )
```

순서가 의미가 없어 결과값도 마음대로 나온다.

```
# 세트형 집합 연산자
a = set('abracadabra')
b = set('alacazam' )

print( 'a = ', a )
print( 'b = ', b )
print( '차집합, a - b = ', a-b )
print( '합집합, a | b = ', a|b )
print( '교집합, a & b = ', a&b )
print( '여집합, a ^ b = ', a^b )
```

■ 사전형, Dictionary Type

```
balls = { 'red' : 4, 'blue' : 3, 'green' : 5 }

print( balls )          # 사전형값 확인
print( type(balls) )    # 데이터형 확인
print( len(balls) )     # 사전형 길이 확인
```

```
--------------------------------------------------------------------------------
# 검은공 항목 추가
balls['black'] = 1
print( balls )



--------------------------------------------------------------------------------
# 녹색공 항목 제거
del balls['green']
print( balls )



--------------------------------------------------------------------------------
# 값 변경
balls['black'] = 3;
print( balls )



--------------------------------------------------------------------------------
# 사전키 치출후 리스트형
print( list(balls.keys()) )              # 사전키 추출후 리스트형으로 전환
print( sorted(balls.keys()) )            # 사전키 추출후 오름차순 정렬
print( list(balls.values()) )            # 사전값 추출후 리스트형으로 전환



--------------------------------------------------------------------------------
print( 'blue' in balls )                 # 키존재 유무 확인
print( 'white' not in balls )            # 키누락 유무 확인
```

```
--------------------------------------------------------------------------------
# 리스트안의 항목이 키와 값의 쌍으로 이루어진 튜플인 경우
balls2 = dict( [('brown', 3), ('gray', 7)] )
print( balls2 )

# 키가 단순한 문자열인 경우, dict 함수의 인자값 형태로 진행
balls3 = dict( brown=4, gray=8 )
print( balls3 )
```