[리스트(List)]

- 리스트 특징
- 리스트 관련 함수, 메소드

리스트

- 리스트는 하나의 변수에 여러 개의 값을 할당하는 자료형
- 리스트의 시작과 끝은 대괄호로 구분. 괄호 속에 콤마로 구분된 값(요소, item)들이 있음

```
>>> odd = [1, 3, 5, 7, 9]          # [ ]를 이용하여 리스트 생성
>>> squares = [1, 4, 9, 16, 25]
>>> empty = []                     # 공백 리스트
>>> empty2 = list()                # 공백 리스트 만드는 다른 방법
>>> empty
[]
>>> empty2
[]
>>> a = [1, 2.3, 'hello', [1, 2], (1, 2)] # 리스트 요소는 어떤 것이라도 가능
```

리스트의 인덱싱(요소 접근하기)과 슬라이싱(잘라내기)

- 문자열과 같이 인덱싱, 슬라이싱이 가능
- 인덱싱

```
>>> L  = [1, 2, 3, 4, 5]
>>> L[0] # 리스트 요소 추출하기(인덱싱은 0부터)
1
>>> L[-1] # 마지막 요소
5
```

슬라이싱(잘라내기)

- 슬라이싱

```
>>> L = [1, 2, 3, 4, 5]
>>> L[0:3]          # 첨자 0부터 2까지 잘라내기
[1, 2, 3]
>>> L[-3:]          # 뒤에서 3번째부터 마지막까지 잘라내기
[3, 4, 5]
>>> L2 = L[:]       # 슬라이싱 이용해 새 리스트 생성 가능
>>> L2
[1, 2, 3, 4, 5]
>>> L3 = L[::2]     # step 2만큼씩
>>> L3
[1, 3, 5]
>>> L4 = L[::-2] # step -2만큼씩
>>> L4
[5, 3, 1]
```

리스트 합치기(병합) : + 연산 사용

```
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b # 리스트 합치기
[1, 2, 3, 4, 5, 6]
```

리스트의 요소 수정, 추가하기

```
# 수정하기
>>> cubes = [1, 8, 27, 65, 125] #세제곱수
>>> 4 ** 3 # 4의 세제곱은 64!
64
>>> cubes[3] = 64 # 리스트 요소 수정하기
>>> cubes #수정된 리스트
[1, 8, 27, 64, 125]

# 마지막에 요소 추가하기
>>> cubes.append(216) # 마지막에 리스트 요소 추가하기
>>> cubes.append(7 ** 3) # 마지막에 리스트 요소 추가하기
>>> cubes
[1, 8, 27, 64, 125, 216, 343]
```

요소 수정 및 제거 하기

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters[2:5] = ['C', 'D', 'E'] # 리스트 요소 수정
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> letters[2:5] = [] # 리스트 요소 제거
>>> letters
['a', 'b', 'f', 'g']
>>> letters[:] = [] # 리스트 요소를 모두 제거
>>> letters
[]
```

내장 함수를 이용한 리스트 다루기

```
# 리스트 길이 알아내기
>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
4
# 리스트 요소 합, 최소값, 최대값 구하기
>>> num = [1, 2, 3, 4, 5]
>>> sum(num)
15
>>> min(num)
1
>>> max(num)
5
```

리스트 처리 함수 요약

함수명	사용 방법	설명
del()	del(List[pos])	List에서 pos 위치의 요소 삭제
len()	len(List)	List 요소의 전체 개수를 센다
sum()	sum(List)	List 요소의 합을 계산
min()	min(List)	List 요소 중 최솟값 찾기
max()	max(List)	List 요소 중 최댓값 찾기
list()	L1=list(range(3)) L2=list((1,2))	특정값 또는 튜플 등 자료구조를 리스트로 변환
sorted()	newList = sorted(List)	List 요소를 정렬해 newList 만듬

리스트 처리 메소드 요약

메소드명	사용 방법	설명
append()	List.append(val)	List 맨 뒤에 val 요소 추가
clear()	List.clear()	List의 내용을 지움
copy()	newList = List.copy()	List의 내용을 newList에 복사
count()	List.count(val)	List에서 val 값의 갯수를 센다
extend()	List.extend(newList)	List 뒤에 newList를 추가
index()	List.index(val)	val을 찾아 해당 첨자 반환(여럿이면 첫 첨자)
insert()	List.insert(pos, val)	List의 pos 위치에 val 삽입
pop()	List.pop()	List 맨 뒤에서 요소 삭제
remove()	List.remove(val)	List에서 val 값 요소 삭제(여럿이면 첫 요소만)
reverse()	List.reverse()	List 요소를 뒤집어 역순으로 만듬
sort()	List.sort()	List 요소를 정렬

리스트의 복사

리스트를 복사할 때 주의해야 할 점이 있다. 리스트의 복사에는 다음 두 종류가 있다.

- 얕은 복사(shallow copy) :
  복사 전의 리스트와 복사 후의 리스트가 같은 메모리 공간을 공유하는 복사
- 깊은 복사(deep copy) :
  복사 전 리스트와 복사 후 리스트가 별개의 다른 메모리 공간을 사용하는 복사

얕은 복사

- 숫자 4개로 이루어진 리스트 oL을 얕은 복사해 새 리스트 nL을 만든 후 oL의 값을 변경한 후 oL과 nL의 결과를 출력하는 프로그램
- oL의 값을 바꾸었지만 새로운 리스트 nL의 값도 같이 변경되었기 때문에 두 리스트가 같은 메모리를 사용한다는 것을 알 수 있음

```
>>> oL = [1,2,3,4]
>>> nL = oL
>>> nL
[1, 2, 3, 4]
>>> oL.insert(0,0) # index 0위치에 0 삽입
>>> oL.append(5) # 끝에 5 추가
>>> oL
[0, 1, 2, 3, 4, 5]
>>> nL
[0, 1, 2, 3, 4, 5]
```

깊은 복사

- 깊은 복사(deep copy)는 다음과 같이 세 가지 방법으로 가능
- 리스트 복사 후에 oL의 값을 변경해도 새로운 리스트 nL1, nL2, nL3의 값은 불변임을 확인 가능

```
>>> oL = [1, 2, 3, 4]
>>> nL1 = oL.copy() # 깊은 복사 방법 1
>>> nL2 = oL[:]     # 깊은 복사 방법 2
>>> nL3 = list(oL)  # 깊은 복사 방법 3
>>> oL.insert(0, 0) # index 0 위치에 0 삽입
>>> oL.append(5)    # 끝에 5 추가
>>> oL
[0, 1, 2, 3, 4, 5]
>>> nL1
[1, 2, 3, 4]
>>> nL2
[1, 2, 3, 4]
>>> nL3
[1, 2, 3, 4]
```

List comprehension(리스트 컴프리헨션, 리스트 함축)

- 간단히 리스트를 만들 수 있는 방법

```
리스트 = [ 식 for 요소 in 반복객체(range(),리스트,튜플,문자열) ]
리스트 = [ 식 for 요소 in 반복객체(range(),리스트,튜플,문자열) if 조건식 ]
```

- 1부터 100까지의 수가 들어 있는 리스트 생성

```
L = [ i for i in range(1, 101) ]
```

- 1부터 100까지의 수 중 짝수만으로 구성된 리스트 생성

```
L2 = [ i for i in range(1,101) if i % 2 == 0 ]
```

2차원 리스트 만들기 v1

```
r = int(input('Type row: '))
c = int(input('Type column: '))

L = []
a = 1
for i in range(r):
    L += [[0]*c]
    for j in range(c):
        L[i][j] = a
        a += 1

print(L)
for i in range(r):
    print(L[i])
```

2차원 리스트 만들기 v2

```
L1=[]
L2=[]
v = 1
r = int(input('Type row: '))
c = int(input('Type column: '))

for i in range(r):
    for j in range(c):
        L1.append(v)
        v += 1
    L2.append(L1)
    L1=[]

print(L2)

for i in range(r):
    print(L2[i])
```