리스트 연산

리스트 연산

❖ 리스트에서 특정 요소의 위치 구하기(index)

- index(키워드 [, offset])

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성',
            '토성', '천왕성', '해왕성', '지구']
planet = '지구'
pos = solarsys.index(planet)
print('%s은(는) 태양계에서 %d번째에 위치하고 있습니다.' %(planet, pos))
pos = solarsys.index(planet, 5)
print('%s은(는) 태양계에서 %d번째에 위치하고 있습니다.' %(planet, pos))
```

❖ 리스트에서 특정 위치의 요소를 변경하기

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성',
            '토성', '천왕성', '해왕성']
planet = '화성'
pos = solarsys.index(planet)
solarsys [pos] = 'Mars'
print(solarsys)
```

```
['태양', '수성', '금성', '지구', 'Mars', '목성', '토성', '천왕성', '해왕성']
```

리스트 연산

❖ 리스트에서 특정 구간에 있는 요소 추출하기

- 슬라이싱 이용

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성',
            '토성', '천왕성', '해왕성']
rock_planets = solarsys[1:4]
gas_planets = solarsys[4:]
print('태양계의 암석형 행성: ', end='')
print(rock_planets)
print('태양계의 가스형 행성: ', end='')
print(gas_planets)
```

```
태양계의 암석형 행성: ['수성', '금성', '지구']
태양계의 가스형 행성: ['화성', '목성', '토성', '천왕성', '해왕성']
```

리스트 연산

❖ 리스트에서 짝수 번째 요소만 추출하기

```
listdata = list(range(1, 21))
evenlist = listdata[1::2]
print(evenlist)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
listdata = list(range(1, 21))
oddlist = listdata[::2]
print(oddlist)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

리스트 연산

❖ 리스트 요소 순서를 역순으로 만들기

- reverse()

```
listdata = list(range(5))
listdata.reverse()
print(listdata)
```

```
[4, 3, 2, 1, 0]
```

리스트 연산

❖ 리스트 요소 순서를 역순으로 만들기(reversed)

- 원본은 변화없고 조정된 새로운 시퀀스를 리턴
- list()로 리스트 자료형 변환 필요

```
listdata = list(range(5))
ret1 = reversed(listdata)
print(type(ret1))
print('원본 리스트 ', end='');print(listdata);
print('역순 리스트 ', end='');print(list(ret1))

ret2 = listdata[::-1]
print('슬라이싱 이용 ', end='');print(ret2)
```

```
<class 'list_reverseiterator'>
원본 리스트 [0, 1, 2, 3, 4]
역순 리스트 [4, 3, 2, 1, 0]
슬라이싱 이용 [4, 3, 2, 1, 0]
```

리스트 연산

❖ 리스트 합치기(+)

- 두 개의 리스트를 합쳐 새로운 리스트를 생성

```
listdata1 = ['a', 'b', 'c', 'd', 'e']
listdata2 = ['f', 'g', 'h', 'i', 'j']

listdata3 = listdata1 + listdata2
listdata4 = listdata2 + listdata1

print(listdata3)
print(listdata4)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
['f', 'g', 'h', 'i', 'j', 'a', 'b', 'c', 'd', 'e']
```

리스트 연산

❖ 리스트 반복하기(*)

- list * n 형태로 리스트를 n번 반복하여 새로운 리스트를 생성

```
listdata = list(range(3))
ret = listdata*3
print(ret)
```

```
[0, 1, 2, 0, 1, 2, 0, 1, 2]
```

리스트 연산

❖ 리스트에 요소 추가하기(append)

- 인자로 입력된 값을 리스트의 맨 마지막 요소로 추가

```
listdata = []
for i in range(3):
   txt = input('리스트에 추가할 값을 입력하세요[%d/3]: ' %(i+1))
   listdata.append(txt)
   print(listdata)
```

리스트 연산

❖ 리스트의 특정 위치에 요소 삽입하기(insert)

- 지정한 위치에 새로운 값을 리스트에 삽입
- 기존 값들은 한 칸씩 뒤로 밀림

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성', '토성', '천왕성',
'해왕성']
pos = solarsys.index('목성')
solarsys.insert(pos, '소행성')
print(solarsys)
```

```
['태양', '수성', '금성', '지구', '화성', '소행성', '목성', '토성', '천왕성', '
해왕성']
```

❖ 리스트의 특정 위치의 제거하기(del)

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성', '토성', '천왕성',
'해왕성']
del solarsys[0]
print(solarsys)
del solarsys[-2]
print(solarsys)
```

```
['수성', '금성', '지구', '화성', '목성', '토성', '천왕성', '해왕성']
['수성', '금성', '지구', '화성', '목성', '토성', '해왕성']
```

리스트 연산

❖ 리스트에서 특정 요소 제거하기

- 요소의 값으로 제거

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성', '토성', '천왕성',
'해왕성']
solarsys.remove('태양')
print(solarsys)
```

```
['수성', '금성', '지구', '화성', '목성', '토성', '천왕성', '해왕성']
```

리스트 연산

❖ 리스트에서 특정 구간에 있는 모든 요소 제거하기

- del 슬라이싱

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성', '토성', '천왕성',
'해왕성']
del solarsys[1:3]
print(solarsys)
```

```
['태양', '지구', '화성', '목성', '토성', '천왕성', '해왕성']
```

리스트 연산

❖ 리스트에 있는 요소 개수 구하기(len)

```
listdata = [2, 2, 1, 3, 8, 5, 7, 6, 3, 6, 2, 3, 9, 4, 4]
listsize = len(listdata)
print(listsize)
```

```
15
```

❖ 리스트에서 특정 요소 개수 구하기(count)

```
listdata = [2, 2, 1, 3, 8, 5, 7, 6, 3, 6, 2, 3, 9, 4, 4]
c1 = listdata.count(2)
c2 = listdata.count(7)
print(c1)    # 3이 출력됨
print(c2)    # 1이 출력됨
```

```
3
1
```

리스트 연산

❖ 리스트 제거하기

```
listdata = [2, 2, 1, 3, 8, 5, 7, 6, 3, 6, 2, 3, 9, 4, 4]
del listdata
print(listdata)
```

예외 발생

리스트 연산

❖ 리스트 요소 정렬하기(sort)

- 리스트 내부의 요소를 정렬함
- 역순정렬시 reverse=True 인자 전달

```
namelist = ['Mary', 'Sams', 'Aimy', 'Tom', 'Michale', 'Bob', 'Kelly']
namelist.sort()
print(namelist)

# 역순 정렬
namelist.sort(reverse=True)
print(namelist)
```

```
['Aimy', 'Bob', 'Kelly', 'Mary', 'Michale', 'Sams', 'Tom']
['Tom', 'Sams', 'Michale', 'Mary', 'Kelly', 'Bob', 'Aimy']
```

리스트 연산

❖ 리스트 요소 정렬하기(sorted)

- 원본에 변화없고, 정렬된 새로운 리스트를 리턴
- 역순 정렬시 reverse=True 인자 추가

```
namelist = ['Mary', 'Sams', 'Aimy', 'Tom', 'Michale', 'Bob', 'Kelly']
ret1 = sorted(namelist)
ret2 = sorted(namelist, reverse=True)
print(namelist)
print(ret1)
print(ret2)
```

```
['Mary', 'Sams', 'Aimy', 'Tom', 'Michale', 'Bob', 'Kelly']
['Aimy', 'Bob', 'Kelly', 'Mary', 'Michale', 'Sams', 'Tom']
['Tom', 'Sams', 'Michale', 'Mary', 'Kelly', 'Bob', 'Aimy']
```

❖ 리스트 요소 무작위로 섞기(shuffle)

- random 모듈의 함수로 정의
  - random 임포트필요

```
from random import shuffle

listdata = list(range(1, 11))
for i in range(3):
   shuffle(listdata)
   print(listdata)    # 출력 결과는 실행할 때마다 달라짐
```

```
[10, 1, 9, 6, 3, 7, 5, 8, 4, 2]
[10, 8, 9, 3, 7, 1, 6, 2, 4, 5]
[4, 6, 3, 5, 7, 8, 10, 2, 9, 1]
```

리스트 연산

❖ 리스트의 모든 요소를 인덱스와 쌍으로 추출하기(enumerate)

- ㅇ 리스트를 입력받아, 익덱스왁 값의 쌍을 가지는 enumerate 객체를 리턴
  - list()로 리스트 형변환 가능
  - for 문에서 인덱스와 값으로 unpack해서 주로 사용

```
solarsys = ['태양', '수성', '금성', '지구', '화성', '목성', '토성', '천왕성',
'해왕성']
ret = list(enumerate(solarsys))
print(ret)

for i, body in enumerate(solarsys):
   print('태양계의 %d번째 천체: %s' %(i, body))
```

```
[(0, '태양'), (1, '수성'), (2, '금성'), (3, '지구'), (4, '화성'), (5, '목성'),
(6, '토성'), (7, '천왕성'), (8, '해왕성')]
태양계의 0번째 천체: 태양
태양계의 1번째 천체: 수성
:
태양계의 7번째 천체: 천왕성
태양계의 8번째 천체: 해왕성
```

리스트 연산

❖ 리스트의 모든 요소의 합 구하기(sum)

```
listdata = [2, 2, 1, 3, 8, 5, 7, 6, 3, 6, 2, 3, 9, 4, 4]
ret = sum(listdata)
print(ret)    # 65가 출력됨
```

```
65
```

❖ 리스트 요소가 모두 참인지 확인하기(all, any)

```
listdata1 = [0, 1, 2, 3, 4]
listdata2 = [True, True, True]
listdata3 = ['', [], (), {}, None, False]
print(all(listdata1))  # False가 출력됨
print(any(listdata1))  # True가 출력됨
print(all(listdata2))  # True가 출력됨
print(any(listdata2))  # True가 출력됨
print(all(listdata3))  # False가 출력됨
print(any(listdata3))  # False가 출력됨
```

- False로 해석되는 것들
  - 숫자 0
  - 빈 문자열 '', ""
  - 빈 리스트 []
  - 빈 튜플 ()
  - 빈 사전 {}
  - None