List

- 여러 개의 값을 하나의 이름으로 관리하는 데이터 구조
- 여러 자료를 순서적으로 저장하는 구조
- 다양한 자료형으로 리스트 구성 가능
- 대괄호 사이에 쉼표로 구분 된 값 (항목) 목록으로 작성

```
colors = ['red', 'blue', 'green']
```

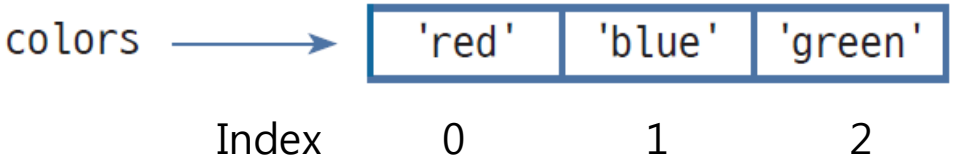

```
#list 정의 예
l = [1, 2, 3, 4]                    #4개의 수치를 리스트로 정의
print(l)
colors = ['red','green', 'blue']    #3개의 칼라 리스트 정의
print(colors)
scores = [95.4, 60, 80, 77, 99]     #5개의 점수 리스트 정의
print(scores)
info = ["홍길동", 22, 185.3, 77.2]   #성명 "홍길동", 나이 22, 키 185.3, 몸무게 77.2 을 info 리스트로 정의
print(info)
```

```
[1, 2, 3, 4]
['red', 'green', 'blue']
[95.4, 60, 80, 77, 99]
['홍길동', 22, 185.3, 77.2]
```

List

- 인덱싱(indexing)
  - list 를 구성하는 항목에 접근하기 위해 위치값(index) 사용 (0부터 시작)

```
colors = ['red', 'blue', 'green']
```




```
#list indexing
print('colors=', colors)
print('colors[0]=', colors[0])
print('colors[1]=', colors[1])
print('colors[2]=', colors[2])
print('colors[3]=', colors[3])
```

```
colors= ['red', 'green', 'blue']
colors[0]= red
colors[1]= green
colors[2]= blue

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-36-c67a9dc1e822> in <module>
      3 print('colors[1]=', colors[1])
      4 print('colors[2]=', colors[2])
----> 5 print('colors[3]=', colors[3])

IndexError: list index out of range
```

List

- 슬라이싱(slicing)
  - list index로 범위를 지정하여 list 일부인 sub list 반환

형식 : 리스트명[시작인덱스:마지막인덱스]

-시작인덱스부터 ' 마지막인덱스 – 1 ' 까지 sub list 반환

- 시작 인덱스 값 생략 시 처음부터 접근
- 마지막 인덱스 값 생략 시 끝까지 접근

- 리버스 인덱스(reverse index) : 인덱스에 –를 붙여 끝에서부터 접근

```
#slicing
print('colors[0:2]=', colors[0:2])  #0:(2-1)범위 값들
print('colors[:2]=', colors[:2])    #시작부터 (2-1) 범위 값들
print('colors[1:]=', colors[1:])    #1부터 끝까지 범위 값들
print('colors[:]=', colors[:])    #처음부터 끝까지 범위 값들
print('colors[-2:]=', colors[-2:])    #끝에서 두번째부터 끝까지 범위 값들
```

```
colors[0:2]= ['red', 'green']
colors[:2]= ['red', 'green']
colors[1:]= ['green', 'blue']
colors[:]= ['red', 'green', 'blue']
colors[-2:]= ['green', 'blue']
```

List

- 인덱싱(indexing) & 슬라이싱(slicing) 예

```
#list indexing, slicing
print(l)       #l list 출력
print(l[0])    #인덱스 0의 값
print(l[2])    #인덱스 2의 값
print(l[-1])   # 리버스인덱스 -1의 값
print(l[-2])   # 리버스인덱스 -2의 값
print(l[1:3]) #슬라이싱,  인덱스 1:(3-1) 범위의 값들
print(l[:3])   #슬라이싱, 시작부터 (3-1) 범위의 값들
print(l[3:])   #슬라이싱, 인덱스 3부터 끝까지 범위의 값들
print(l[-3:])   #슬라이싱, 리버스인덱스 -3부터 끝까지 범위의 값들
print(l[0:6])   #슬라이싱, 범위를 벗어나는 경우 시작부터 끝까지 범위의 값들로 처리
```

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

List

- 슬라이싱(slicing) : 증가값
  - list index로 범위를 증가값 만큼씩 이동하면서 list 일부 반환

형식 : 리스트명[시작인덱스:마지막인덱스:증가값]

-시작인덱스부터 ' 마지막인덱스 – 1 ' 까지 증가값 만큼씩 이동하면서 sub list 반환

-증가 값이 음수이면 역순(끝에서 시작순)으로 접근

```
# slicing step
print('colors=', colors)
print('colors[::2]=', colors[::2])      #시작부터 끝까지 2씩 증가하여 값들 추출
print('colors[::-1]=', colors[::-1])    #끝부터 시작까지 역으로 1씩 감소하여 값들 추출
```

```
colors= ['red', 'green', 'blue']
colors[::2]= ['red', 'blue']
colors[::-1]= ['blue', 'green', 'red']
```

List

- list 연산
  + : 두 개 리스트 결합
  * : 리스트의 항목을 n배수 만큼 추가

```
# list 연산
l1 = [1,2,3,4,5]
l2 = [6,7,8,9]
l3 = l1 + l2   #list 결합
print(l3)
l4 = l1 * 2   #list 2번 반복
print(l4)
```

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

List

- list 길이, 타입 확인
  - 길이 확인 함수: len()
  - 타입 확인 함수: type()

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(letters)
print(len(letters))
print(type(letters))
print(type(letters[0]))

s=['male', 1, 'female', 2]
print(s)
print(len(s))
print(type(s[0]))
print(type(s[1]))
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
7
<class 'list'>
<class 'str'>
['male', 1, 'female', 2]
4
<class 'str'>
<class 'int'>
```

List

- list 추가, 결합, 삽입, 삭제

함수	기능	용례
append( )	새로운 값을 기존 리스트의 맨 끝에 추가	color.append('white')
extend( )	새로운 리스트를 기존 리스트에 추가(덧셈 연산과 같은 효과)	color.extend(['black','purple']
insert( )	기존 리스트의 i번째 인덱스에 새로운 값을 추가. i번째 인덱스를 기준으로 뒤쪽의 인덱스는 하나씩 밀림	color.insert(0, 'orange')
remove( )	리스트 내의 특정 값을 삭제	color.remove('white')
del	특정 인덱스값을 삭제	del color[0]

List

- list 값 변환(replace), 삭제(remove) , 삽입(insert), 추가(append) 예

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(letters)
# replace
letters[3:5] = ['D', 'E']    #letters의 3~4범위의 값을  ['D', 'E']로 변경
letters[6] = 'F'             #letters의 인덱스 6의 값을 'F'로 변경
print(letters)
# remove
letters[2:4] = []            #letters의 2~3범위의 값을  빈값(empty list) []으로 변경
print(letters)
# insert
letters.insert(0, 'x')      #letters의 인덱스 0위치에 'x' 삽입
print(letters)
# append
letters.append('h')         #letters의 끝에 'h' 추가
print(letters)
# remove
letters.remove('x')        #letters안의 'x' 삭제
print(letters)
# del
del letters[0]              #letters의 0번 인덱스 값 삭제
print(letters)
# clear the list
letters[:] = []            #letters의 모든범위의 값을  빈값(empty list) []으로 변경
print(letters)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
['a', 'b', 'c', 'D', 'E', 'f', 'F']
['a', 'b', 'E', 'f', 'F']
['x', 'a', 'b', 'E', 'f', 'F']
['x', 'a', 'b', 'E', 'f', 'F', 'h']
['a', 'b', 'E', 'f', 'F', 'h']
['b', 'E', 'f', 'F', 'h']
[]
```

List

- 패킹(packing): 한 변수에 여러 개의 데이터를 할당하는 것.
- 언패킹(unpacking): 한 변수의 데이터를 각각의 변수로 반환하는 것.

```
#unpacking 예
r,g,b = colors  # colors 리스트의 값을 언패킹하여 r,g,b 각 변수에 반환
print(r,g,b)
r,g,b, o = colors  # not enough values to unpack (expected 4, got 3)
```

```
red green blue

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-57-8f1a5901f94e> in <module>
      1 r,g,b = colors  # colors 리스트의 값을 언패킹하여 r,g,b 각 변수에 반환
      2 print(r,g,b)
----> 3 r,g,b, o = colors  # not enough values to unpack (expected 4, got 3)

ValueError: not enough values to unpack (expected 4, got 3)
```

List 연습

- 실습 문제 2. 리스트 연습
  (1) 3명의 성명으로 names 리스트 생성 및 출력
  (2)처음부터 두번째까지 접근하여 출력
  (3)세번째에 새로운 이름 삽입
  (4)끝에서 두번째부터 끝까지 접근하여 출력
  (5)새로운 이름을 추가
  (6)첫번째 성명을 다른이름으로 변경
  (7)끝에서 2개 제거

```
names = ['kim', 'lee', 'min']
print(names)
print(names[:2])
names.insert(2,'park')
print(names)
print(names[-2:])
names.append('song')
names[0] = 'shin'
print(names)
del names[-2:]
print(names)
```

```
['kim', 'lee', 'min']
['kim', 'lee']
['kim', 'lee', 'park', 'min']
['park', 'min']
['shin', 'lee', 'park', 'min', 'song']
['shin', 'lee', 'park']
```