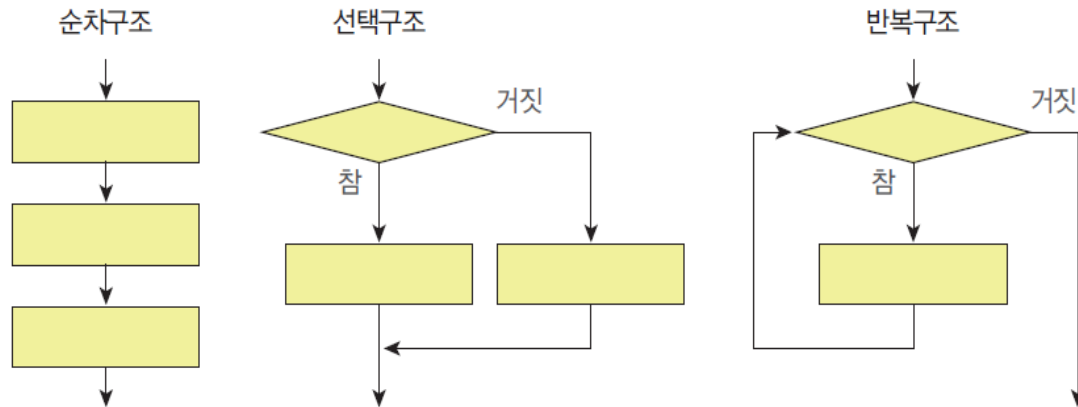


기초 인공지능 프로그래밍

4장 조건문 & 리스트 자료형

if 조건문

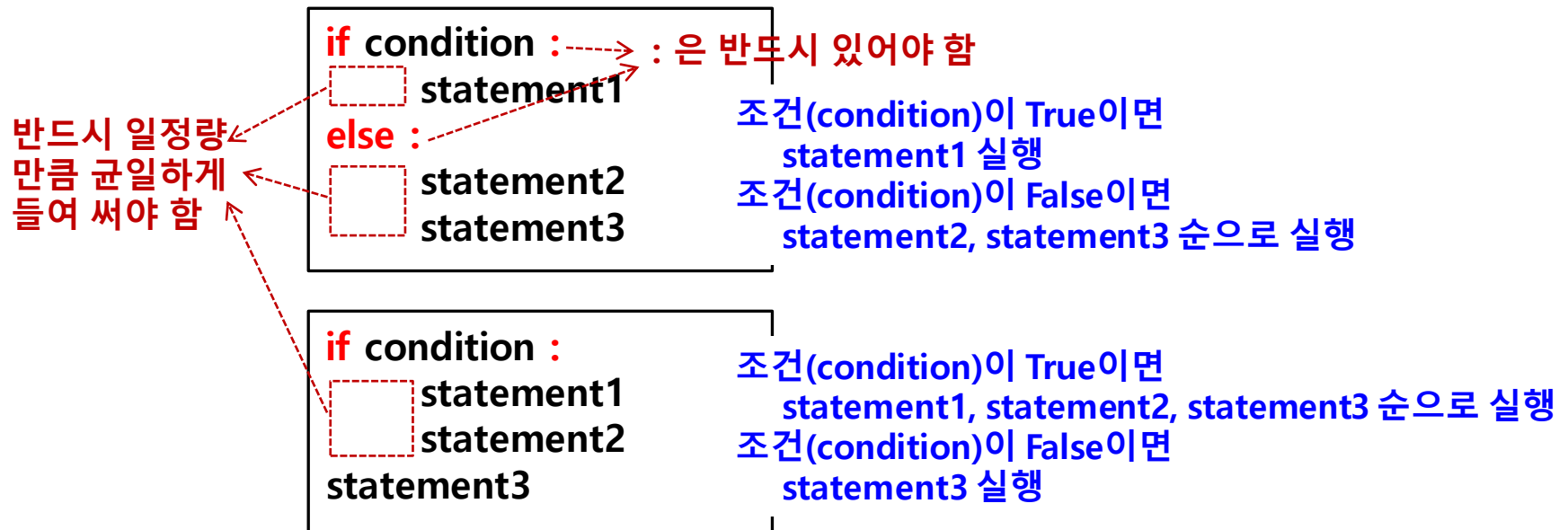
- 3가지 기본 제어 구조
 - 순차 구조(sequence) : 명령들을 순차적으로 실행하는 구조
 - 선택 구조(selection) : 명령을 조건에 따라 선택하여 실행하는 구조
 - 반복 구조(iteration) : 동일한 명령들을 반복 실행하는 구조



- if 조건문(conditional statements)은 어떤 상황에 따라 실행해야 할 코드가 다를 때 사용(선택 구조에 해당)

if 조건문

- if 조건문의 형식
 - 조건(condition)의 결과가 True이면 같은 크기로 들여쓰기(indentation) 되어 있는 명령어들(코드 블록)이 실행되고, False이면 else 후에 같은 크기로 들여쓰기 되어 있는 명령어들이 실행
 - 조건이 False일 때 실행해야 할 명령어가 필요 없다면 else 부분은 생략 가능함



조건(condition)

- if 조건문에서 **조건**이란 참과 거짓을 판단하는 명령어를 말함
- 조건에 사용되는 명령어
 - in 연산자, not in 연산자, 관계 연산자, 논리 연산자의 결과
 - 산술 연산식의 결과 값
 - 변수가 참조하는 자료형의 값으로 참과 거짓 판단 가능

자료형	참	거짓
숫자	0이 아닌 숫자	0
문자열	"abc"	""
리스트	[1,2,3]	[]
튜플	(1,2,3)	()
사전	{"a":"b"}	{}

0, 0.0, ""(빈 문자열), 빈 자료형 등은 **False**로 간주하고 그 외의 나머지 값들은 모두 **True**로 간주

조건(condition)

```
test.py - C:\Users\Wadmin\Desktop\script\test.py (3.9.2)
File Edit Format Run Options Window Help
n = int(input("Enter a number : "))
if n % 2: # n이 홀수이면 2로 나눈 나머지가 1이 되어 조건이 True
    s = "odd"
    print("%d is %s." % (n, s))
else: # n이 짝수이면 2로 나눈 나머지가 0이 되어 조건이 False
    s = "even"
    print("%d is %s." % (n, s))
```

동일 결과를 내는 코드

```
if명령어 설명예시코드.py - E...
File Edit Format Run Options Window Help
n = int(input("Enter a number : "))
s = "even"
if n % 2 == 1:
    s = "odd"
print("%d is %s." % (n, s))
```

같은 조건식

```
Enter a number : 15
15 is odd.
>>>
```

```
test.py - C:\Users\Wadmin\Desktop\script\test.py (3.9.2)
File Edit Format Run Options Window Help
n = input("Enter : ")
print(n)
print(n == "")
print(len(n))
```

변수 n에는 빈 문자열이 저장되어 있기 때문에 print(n) 함수 실행으로 줄바꿈만 발생

enter key만 입력

```
Enter :
True
0
>>>
```

n == "" (빈문자열) 이므로 True 출력
빈 문자열에는 데이터가 없으므로 len(n)은 0

조건(condition)

```
x = float(input("Enter a number : "))  
if not x :    # 입력 받은 데이터가 0일 때(x의 값) x는 False이므로 not x 는 True가 됨  
    print("The x's value is zero")  
else :  
    print("The x's value is not zero")
```

```
name = input("Name?: ")  
if name != "":    # 어떤 데이터라도 입력된 경우(스페이스 키만 입력 된 경우도 포함)  
    print("%s is your name" %(name))  
else :    # enter key만 입력된 경우  
    print("No data!")
```

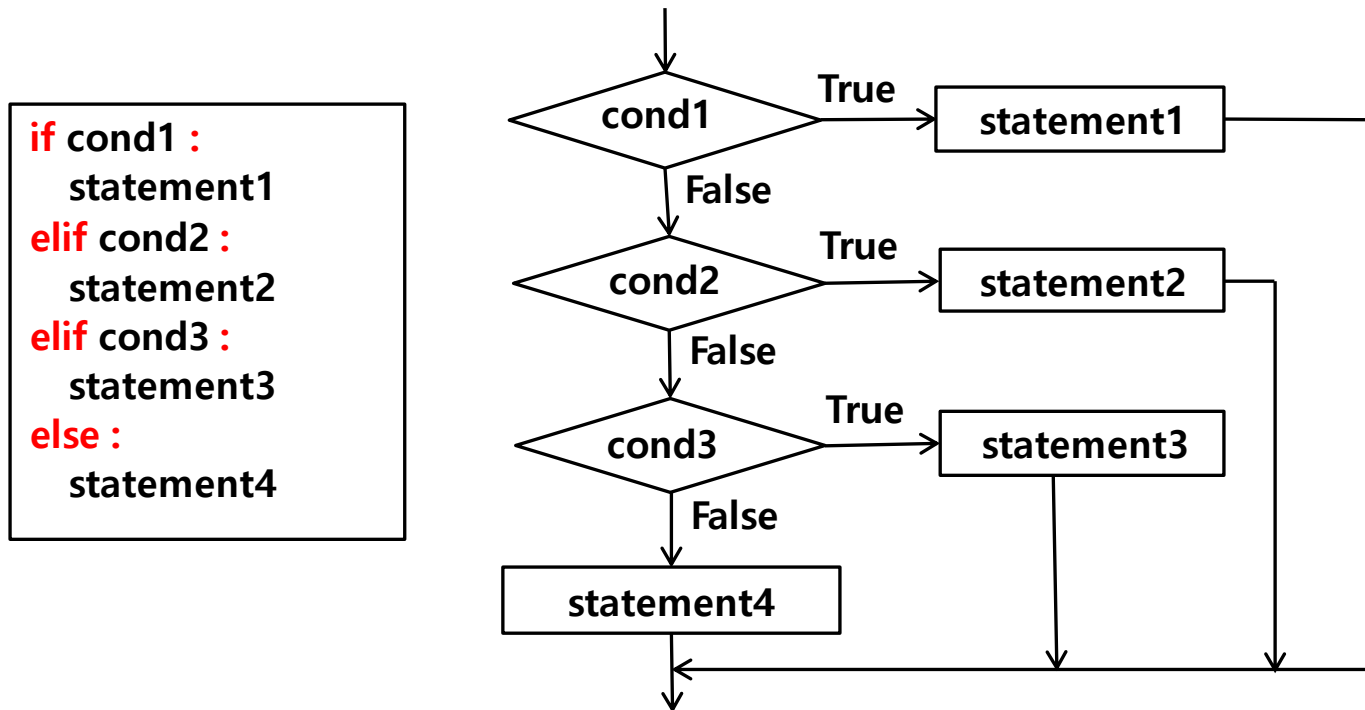
동일 조건식

if name :

if len(name) != 0 :

if – elif 조건문

- 다양한 조건을 판단할 때 사용
 - elif는 앞 조건이 거짓일 때 다시 조건을 검사하는 if문
 - 조건이 True이면 같은 크기로 들여쓰기(indentation) 되어 있는 명령어들(코드 블록) 실행
 - 마지막 else는 필요하지 않으면 생략 가능



if – elif 조건문

- 입력 받은 점수로 grade를 부여하는 코드

```
test.py - C:\Users\Wadmin\Desktop...  
File Edit Format Run Options Window Help  
score = int(input("Enter score : "))  
grade = "F"  
if score >= 90:  
    grade = "A"  
elif score >= 80 :  
    grade = "B"  
elif score >= 70:  
    grade = "C"  
elif score >= 60:  
    grade = "D"  
print("Grade is %s " %grade)  
Ln: 11 Col: 0
```

같은 의미

```
Enter score : 65  
Grade is D  
>>>
```

```
*test.py - C:\Users\Wadmin\Desktop...  
File Edit Format Run Options Window Help  
score = int(input("Enter score : "))  
if score >= 90:  
    grade = "A"  
elif score >= 80 :  
    grade = "B"  
elif score >= 70:  
    grade = "C"  
elif score >= 60:  
    grade = "D"  
else:  
    grade = "F"  
print("Grade is %s " %grade)  
Ln: 13 Col: 0
```

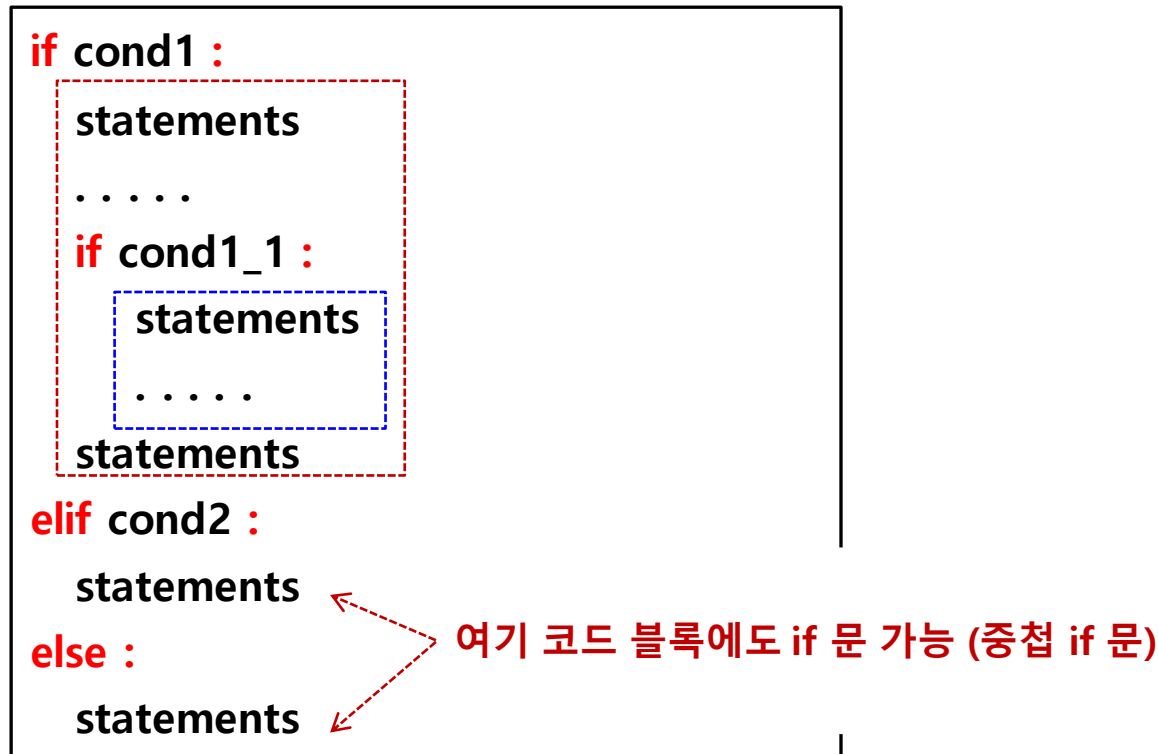
- 입력 받은 정수가 2 또는 3으로 나누어지는지 판정하는 코드

```
test.py - C:\Users\Wadmin\Desktop\script\test.py (3.9.2)  
File Edit Format Run Options Window Help  
n = int(input("Enter a number : "))  
if (n % 2 == 0) and (n % 3 == 0) :  
    print("%d is divided by both 2 and 3." %n)  
elif (n % 2 == 0) and (n % 3 != 0) :  
    print("%d is divided by 2 but not by 3." %n)  
elif (not n % 2 == 0) and (n % 3 == 0) :  
    print("%d is divided by 3 but not by 2." %n)  
else :  
    print("%d is neither divided by 2 nor by 3." %n)  
Ln: 9 Col: 52
```

```
Enter a number : 65  
65 is neither divided by 2 nor by 3.  
>>>
```

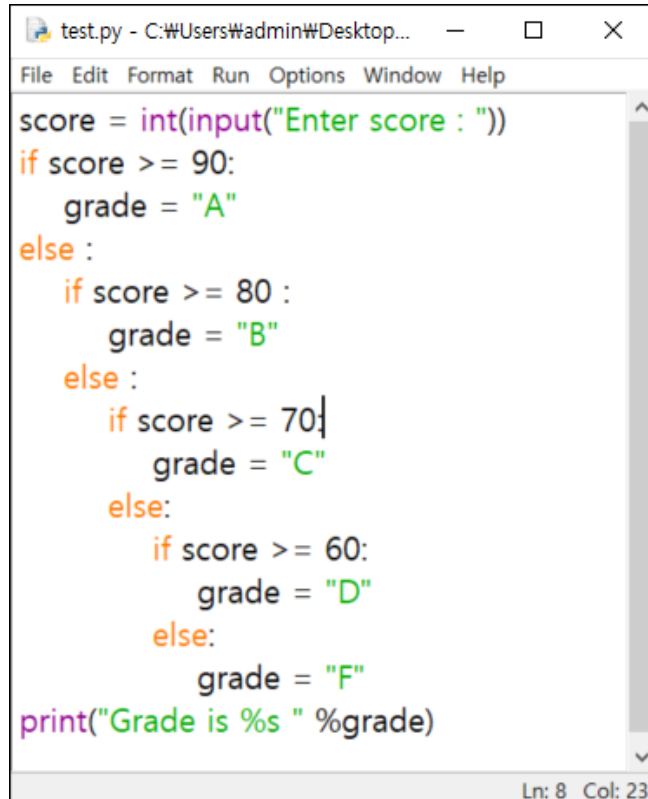

중첩(nested) if 문

- 조건 확인 후 또 다른 조건을 검사해야 하는 경우, 중첩된 if-else 구조를 사용
 - if 문의 코드 블록 안에 또 다른 if 문을 사용 (들여쓰기 유의)

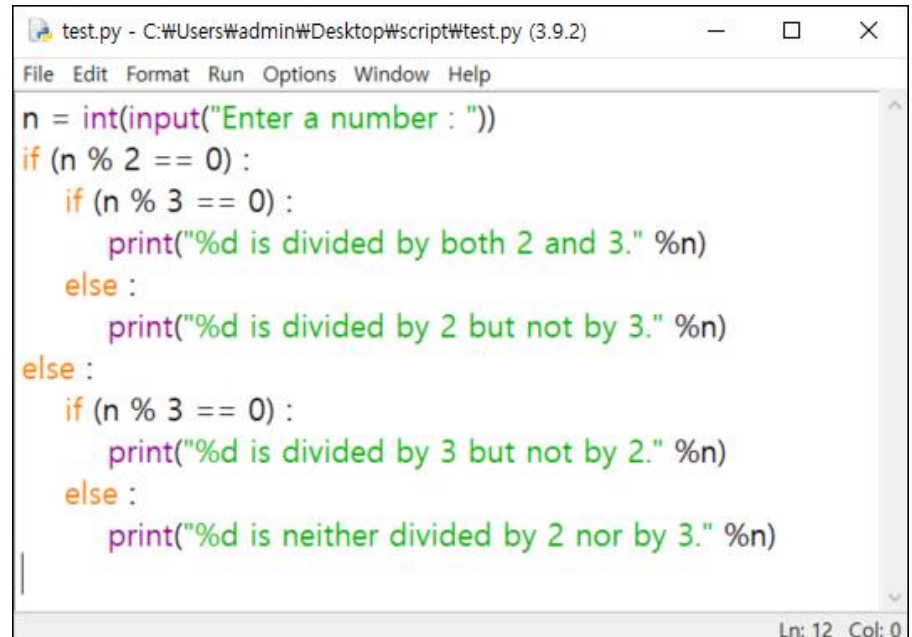


중첩(nested) if 조건문

- if – elif 조건문의 예제 코드를 중첩 if 문으로 변경
 - 조건 검사 단계가 많아지면 한쪽으로 치우친 코드로 작성될 수 있음



```
test.py - C:\Users\admin\Desktop...  
File Edit Format Run Options Window Help  
score = int(input("Enter score : "))  
if score >= 90:  
    grade = "A"  
else :  
    if score >= 80 :  
        grade = "B"  
    else :  
        if score >= 70:  
            grade = "C"  
        else:  
            if score >= 60:  
                grade = "D"  
            else:  
                grade = "F"  
print("Grade is %s " %grade)  
Ln: 8 Col: 23
```



```
test.py - C:\Users\admin\Desktop\script\test.py (3.9.2)  
File Edit Format Run Options Window Help  
n = int(input("Enter a number : "))  
if (n % 2 == 0) :  
    if (n % 3 == 0) :  
        print("%d is divided by both 2 and 3." %n)  
    else :  
        print("%d is divided by 2 but not by 3." %n)  
else :  
    if (n % 3 == 0) :  
        print("%d is divided by 3 but not by 2." %n)  
    else :  
        print("%d is neither divided by 2 nor by 3." %n)  
Ln: 12 Col: 0
```

리스트(list)

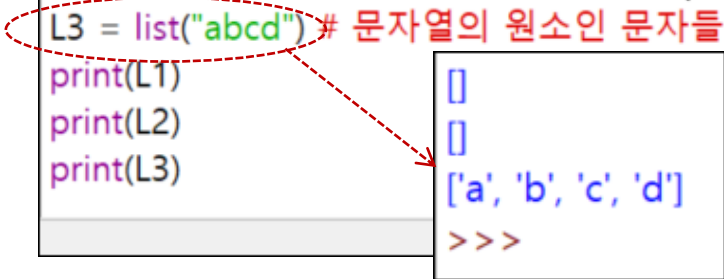
- 같은 의미의 많은 데이터를 하나의 변수명으로 '순서대로' 저장하고 관리해야 할 때 사용
 - 저장하는 순서가 의미가 있는 데이터(sequence data)를 저장하는 객체
 - 따라서, 문자열처럼 인덱싱(indexing)과 슬라이싱(slicing)이 가능
 - 파이썬이 제공하는 모든 자료형의 데이터들은 리스트의 원소 값이 될 수 있음
- 형식(syntax)
 - `list_name = [element1, element2, element3,...]`
 - 대괄호 []안에 콤마로 구분하여 원소들을 표시
 - 리스트의 각 원소는 변수와 같은 의미이며, 그 값은 변경 가능

리스트(list)

```
chap4.py - C:\Users\admin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help

scores = [67, 22, 90, 0]
a = [1, 2.2, "python", [10, 20]] # 리스트도 리스트의 원소가 될 수 있음
L1 = [] # 원소가 없는 빈 리스트 생성
L2 = list() # list() 함수로 빈 리스트 생성
L3 = list("abcd") # 문자열의 원소인 문자들을 원소로 하는 리스트 생성

print(L1)
print(L2)
print(L3)
```



```
[]
[]
['a', 'b', 'c', 'd']
>>>
```

Ln: 4 Col: 34

```
>>> L = list(10)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    L = list(10)
TypeError: 'int' object is not iterable
>>>
```

list() 함수로 리스트 객체 생성

- 함수의 인자는 **iterable한 객체 하나만** 허용
- 그 객체의 원소들이 리스트의 원소로 변환
- 인자를 지정하지 않으면 빈 리스트 생성

리스트 인덱싱

- 인덱싱을 이용해 리스트의 각 원소가 가리키고 있는 값에 접근할 수 있음
- 인덱싱 범위 (문자열과 동일하게 적용)
 - 리스트의 크기(원소 개수)는 내장함수 `len()`으로 구함
 - positive indexing : $0 \sim \text{len}(\text{list객체}) - 1$

```
letters = [ 'A', 'B', 'C', 'D', 'E', 'F' ]  
n = len(letters)    #size of list, n = 6
```

letters
n=6

'A'	'B'	'C'	'D'	'E'	'F'
0	1	2	3	4	5

- 리스트의 원소가 순서 있는 자료형인 경우, 추가 인덱싱 가능

```
chap4.py - C:\Users\Wadmin\Desktop\script\chap4.p...  
File Edit Format Run Options Window Help  
0 1 2 3  
L = [4, 1, 8, 6]  
0 1 2 3 4  
mL = [1, 2.2, "Sogang", "대학교", [8, 4, "Python"]]  
print(len(L), len(mL))  
print(L[0], mL[3], mL[len(mL) - 1])  
print(mL[4][2])  
print(mL[4][2][0])  
Ln: 7 Col: 0
```

4 5
4 대학교 [8, 4, 'Python']
Python
P
>>>

리스트 슬라이싱

- 문자열과 동일하게 적용
- 형식 (L을 n개의 원소를 갖는 리스트라고 가정)

$L[s : e : sp]$ # $s = \text{start}$, $e = \text{end}$, $sp = \text{step}$ 을 의미

- 슬라이싱 규칙 ($s, e \geq 0$ 인 경우만 설명)
 - $sp > 0$ 인 경우 ($s < e$ 이어야 함. 아니면 결과는 빈 리스트 생성)

- index를 s 부터 $e-1$ 까지 sp 씩 증가하며 리스트를 참조
 - s 를 생략하면 0부터, e 를 생략하면 끝까지, sp 를 생략하면 +1씩
- $L[:e:sp]$ $L[s::sp]$ $L[s:e]$

- $sp < 0$ 인 경우 ($s > e$ 이어야 한다. 아니면 결과는 빈 리스트 생성)

- index를 s 부터 $e+1$ 까지 $|sp|$ 씩 감소하며 리스트를 참조
 - s 를 생략하면 끝에서부터 $e+1$ 까지 $|sp|$ 씩 감소
 - e 를 생략하면 s 부터 첫 번째 원소까지 $|sp|$ 씩 감소
- $L[:e:sp]$
- $L[s::sp]$

리스트 슬라이싱

- 리스트 `a = [0, 1, 2, 3, 4, 5, 6, 7, 8]`

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

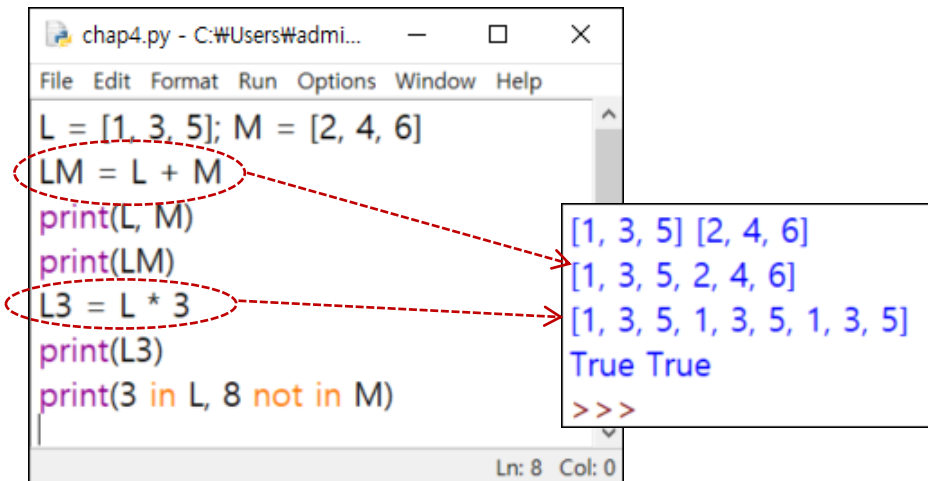
- 리스트 슬라이싱의 결과는 리스트(원소가 없으면 빈 리스트(`[]`), 원소가 한 개일 경우도 리스트(`[data]`))
- 리스트의 일부 원소 값을 참조할 필요가 있는 경우 슬라이싱 사용

```
chap4.py - C:\Users\admin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help
a = [0, 1, 2, 3, 4, 5, 6, 7, 8]
print(a[:6:2])
print(a[1:7:2])
print(a[3:20:3]) # end index가 해당 객체의 크기를 벗어나면 무시하고 끝까지
print(a[4:4])    # [] (end - 1까지 이므로 참조할 데이터가 없기 때문에)
print(a[4:1:1])  # [] (start index > end index이기 때문에)
print(a[4:5])    # index 4의 데이터만 참조
print(a[4])       # indexing(slicing이 아님)
print(a[::-1])   # 끝에서 부터 역순으로 데이터 참조
```

```
[0, 2, 4]
[1, 3, 5]
[3, 6]
[]
[]
[4]
4
[8, 7, 6, 5, 4, 3, 2, 1, 0]
>>>
```

리스트 연산

- Concatenation 연산자 + (리스트의 연결 연산)
 - 두 리스트의 원소들을 연결하여 새로운 리스트 생성
- Repetition 연산자 * (리스트의 반복 연산)
 - **list * n** 은 list의 원소들을 n번 반복하여 새로운 리스트 생성
- Membership 연산자 in, not in (문자열에서와 동일)



The screenshot shows a Python IDE window titled 'chap4.py - C:\Users\Wadmi...'. The code in the editor is as follows:

```
L = [1, 3, 5]; M = [2, 4, 6]
LM = L + M
print(L, M)
print(LM)
L3 = L * 3
print(L3)
print(3 in L, 8 not in M)
```

Red dashed circles highlight the lines `LM = L + M` and `L3 = L * 3`. A callout box on the right displays the output of these operations:

```
[1, 3, 5] [2, 4, 6]
[1, 3, 5, 2, 4, 6]
[1, 3, 5, 1, 3, 5, 1, 3, 5]
True True
>>>
```

리스트 L, M의 원소는 변하지 않고, 연산 결과는 새로운 리스트 객체 생성

리스트 메소드

method	Description (x: 리스트, a: 임의의 객체)
x.append(a)	데이터 a를 리스트 x의 끝에 추가
x.extend(L)	리스트 L의 모든 원소를 리스트 x의 마지막에 추가
x.insert(i, a)	a를 리스트 x의 index i에 추가
x.remove(a)	리스트 x에서 원소 값이 데이터 a인 첫 원소 제거 (반환 값 없음)
x.pop()	x의 마지막 원소 제거 및 반환
x.pop(i)	x[i]를 x에서 제거하고 그 값을 반환
x.clear()	리스트 x의 모든 원소를 삭제. 빈 리스트가 됨
x.index(a)	리스트 x에서 원소 값이 a인 첫 번째 원소의 index를 반환
x.count(a)	리스트 x에서 a 값과 같은 원소의 개수를 반환
x.sort()	x의 원소들을 오름차순으로 정렬 내림차순으로 정렬하려면 x.sort(reverse=True) 사용
x.reverse()	x의 원소들을 역으로 재 배치 (정렬과 다름)
x.copy()	a shallow copy of the list (y = x[:]와 동일)

리스트 메소드 사용 예시

```
chap4.py - C:\Users\admin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help
L = [10, 20, 10, 20, 10]
L.append(100) # 원소 100을 리스트의 끝에 추가. L[len(L):len(L)] = [100]
print(L)
print("*****")
L.insert(2, 100) # 원소 100을 인덱스 2의 위치에 삽입. L[2:2] = [100]
print(L)
print("*****")
data = L.pop() # 마지막 원소 삭제 및 값 반환
print(data)
print(L)
print("*****")
data = L.pop(1) # 인덱스 1의 원소 삭제 및 값 반환
print(data)
print(L)
print("*****")
L.remove(10) # 원소 값 10인 원소 삭제, 여러 개인 경우는 첫번째만 삭제
print(L)
print("*****")
idx = L.index(10) # 원소 값 10인 원소의 인덱스 반환(여러개이면 첫번째 인덱스)
c = L.count(10) # 원소 값 10인 원소의 개수 반환
print(idx)
print(c)
```

```
[10, 20, 10, 20, 10, 100]
*****
[10, 20, 100, 10, 20, 10, 100]
*****
100
[10, 20, 100, 10, 20, 10]
*****
20
[10, 100, 10, 20, 10]
*****
[100, 10, 20, 10]
*****
1
2
>>>
```

리스트 메소드 사용 예시

```
chap4.py - C:\Users\Wadmin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help
L = []
L.append("first")
L.append("second")
print(L) # 추가한 순서대로 원소 저장
print("*****")
L.clear() # 리스트의 모든 원소 삭제
print(L)
print("*****")
L = list("That's good!") # 문자열 객체를 리스트 객체로 변환
print(L)
print("*****")
del L # 변수 정보 삭제
print(L)
```

```
['first', 'second']
*****
[]
*****
['T', 'h', 'a', 't', "'", 's', ' ', 'g', 'o', 'o', 'd', '!']
*****
```

Traceback (most recent call last):

File "C:\Users\Wadmin\Desktop\script\chap4.py", line 13, in <module>
 print(L)

NameError: name 'L' is not defined

>>>

리스트 메소드 사용 예시

```
chap4.py - C:\Users\admin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help
L = [3, 5, 1]; M = [2, 6, 4, 10]; s = "abcd"
L.extend(M) # 리스트 L에 M(iterable 객체만 허용) 원소를 추가
print(L, M)
L.extend(s) # 리스트 L에 s(iterable 객체만 허용) 원소를 추가
print(L, s)
print("*****")
M.sort() # 리스트 M을 오름차순으로 정렬
print(M)
print("*****")
M.sort(reverse = True) # 리스트 M을 내림차순으로 정렬
print(M)
print("*****")
L.reverse() # 리스트 L을 역순으로 재배치
print(L)
```

```
[3, 5, 1, 2, 6, 4, 10] [2, 6, 4, 10]
[3, 5, 1, 2, 6, 4, 10, 'a', 'b', 'c', 'd'] abcd
*****
[2, 4, 6, 10]
*****
[10, 6, 4, 2]
*****
['d', 'c', 'b', 'a', 10, 4, 6, 2, 1, 5, 3]
>>>
```

Ln: 15 Col: 0

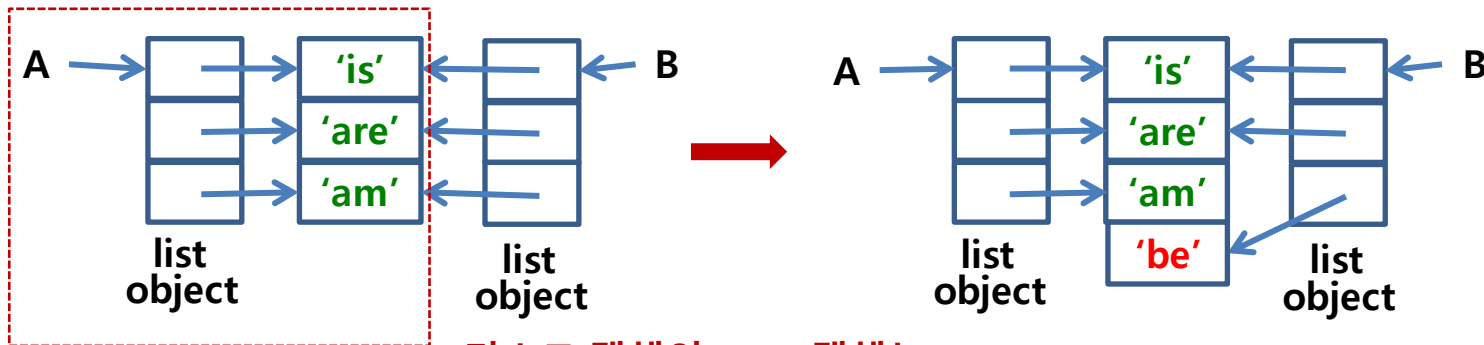
리스트 복사

- 리스트 A를 복사하여 새로운 리스트 B를 생성

```
chap4.py - C:\Users\wadmin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help
A = ['is', 'are', 'am']
B = A.copy()
print(id(A), id(B)) # 변수 A와 B가 참조하는 리스트 객체가 다름(id가 다름). 즉 다른 객체임
B[2] = "be"
print(A, B)
2144361692480 2144354648000
['is', 'are', 'am'] ['is', 'are', 'be']
>>>
```

id() 함수

- 각 객체의 고유번호를 알려주는 내장함수
- 고유번호는 객체가 생성될 때 자동으로 정해지며 삭제될 때 소멸됨



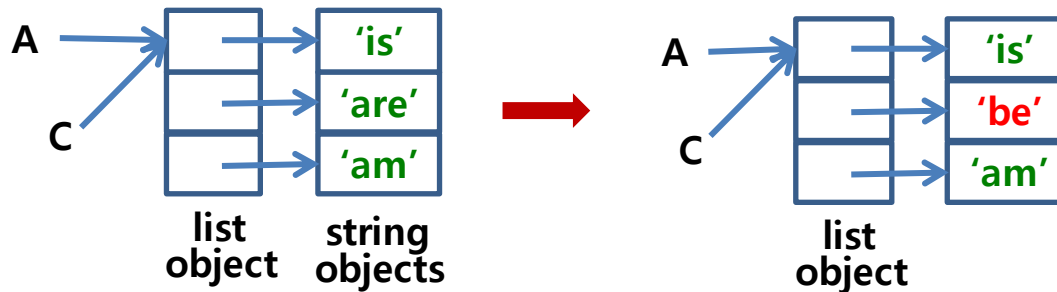
리스트 객체의 구조

리스트 객체의 copy 객체는
다른 리스트 객체가 생성되
고 참조하는 데이터만 같음

리스트 참조 (복사와 다름)

- 같은 리스트 참조하기(= 대입 연산자 이용)
 - 동일한 객체를 다른 이름으로 참조하기

```
chap4.py - C:\Users\admin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help
A = ['is', 'are', 'am']
C = A
print(id(A), id(C)) # 변수 A와 C는 같은 리스트 객체를 참조함(id가 같음).복사가 아님.
C[1] = "be"
print(A, C)
1684636483136 1684636483136
['is', 'be', 'am'] ['is', 'be', 'am']
>>>
```



내장 함수 (함수의 인자가 리스트 자료형일 때)

- 문자열, 튜플, 집합, 사전 등도 동일하게 사용 가능

function	Description(인수 x : 리스트)
all(x)	x의 모든 원소가 True(i.e. != 0)이거나, 또는 x가 빈 리스트이면 True 반환
any(x)	x에 True인(i.e. != 0인) 원소가 한 개라도 있으면 True, 또는 x가 빈 리스트이면 False 반환
enumerate(x)	x의 모든 원소에 대해 튜플 (index, 원소 값)을 얻을 수 있는 enumerate object를 반환. index = 0, 1, 2, ...
len(x)	x의 원소 개수를 반환
list(y)	Iterable한 y를 리스트로 변환하여 반환
max(x)	x의 원소 중 최대 값 반환(비교 불가능하면 TypeError)
min(x)	x의 원소 중 최소 값 반환(비교 불가능하면 TypeError)
sorted(x)	x의 원소를 정렬한 리스트 반환. x는 불변이고 오름차순 또는 내림차순으로 정렬(sorted(x, reverse=True))
sum(x)	x의 원소 합을 반환(더할 수 없으면 TypeError)

내장 함수 사용 예시

- 인자가 리스트인 경우

```
chap4.py - C:\Users\Wadmin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help

L = [1, 9, 4, 2, 8]
M = sorted(L, reverse = True) # 내림차순으로 정렬한 L의 원소들을 원소로 하는 새로운 리스트 생성
print(L)
print(M)
print(sum(L)) # L의 원소들을 합한 값 반환 (원소들이 더하기 연산이 가능할 때만 사용 가능)
L = ["Jan", "Apr", "Sep"]
print(min(L), max(L)) # L의 원소 중 최소, 최대 값인 원소 반환 (문자열인 경우 알파벳 순)
L = sorted(L) # 내장함수로 정렬한 후, 동일한 변수에게 할당 할 수 있음. L.sort()와 동일
print(L)
LL = [1, "2", 3, "4"]
print(min(LL), max(LL)) # 문자열과 정수 원소간에 비교 불가이므로 에러 발생

[1, 9, 4, 2, 8]
[9, 8, 4, 2, 1]
24
Apr Sep
['Apr', 'Jan', 'Sep']
Traceback (most recent call last):
  File "C:\Users\Wadmin\Desktop\script\chap4.py", line 11, in <module>
    print(min(LL), max(LL)) # 문자열과 정수 원소간에 비교 불가이므로 에러 발생
TypeError: '<' not supported between instances of 'str' and 'int'
>>>
```


내장 함수 사용 예시

- 인자가 리스트인 경우

```
chap4.py - C:\Users\admin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help
L = [100, 900, 300, 500]
E = enumerate(L) # L의 각 원소에 대해 생성한 튜플 (index, 원소 값)을 원소로 하는 enumerate 객체를 반환
print(E)
print(type(E))
EtoL = list(E) # enumerate 객체를 사용하기 편리한 리스트 객체로 변환
print(EtoL)
L = ["Jan", "Apr", "Sep"]
M = list(enumerate(L)) # 위의 단계를 한번에 할 수 있음
print(M)
```

<enumerate object at 0x000001ACFF19EC00>
<class 'enumerate'>
[(0, 100), (1, 900), (2, 300), (3, 500)]
>>>

함수 반환값이 enumerate 객체

메소드(함수)의 반환값 처리

- 메소드(함수)는 정의에 따라 반환값이 있는 경우와 반환값 없이 정의된 기능만 실행하는 경우가 있음
- 메소드(함수)의 반환값이 있는 경우는 실행 결과(반환값)를 변수에 할당하여 사용
- 메소드(함수)의 반환값이 없는 경우는 호출만 하고 변수에 할당하지 않음
- 예를 들어 **리스트 메소드 sort()**와 **내장 함수 sorted()**
 - 리스트 메소드 sort()는 리스트 객체 L에 대해 사용되는 경우, L.sort()와 같이 호출하여 리스트 L의 원소들을 정렬만 하고 반환값은 없기 때문에 실행 결과를 변수를 할당하지 않음
 - 내장 함수 sorted()는 인자로 리스트 객체 L을 넘겨 받는 경우, M = sorted(L)과 같이 호출하여 L의 원소들을 정렬하여 새로운 리스트 M에 원소로 저장하여 반환. 해당 값을 변수에 할당하여 사용

메소드(함수)의 반환값 처리

```
chap4.py - C:\Users\admin\Desktop\script\chap4.py (3.9.2)
File Edit Format Run Options Window Help

L = [1, 9, 4, 2, 8]
M = sorted(L, reverse = True)
print("L = ", L, ", ", "M = ", M)
L.sort()
print("L = ", L)
print("#####")
L = [1, 9, 4]
L1 = sorted(L)
L2 = L.sort() # 반환값이 없는 메소드의 실행 결과를 변수에 할당하면 None 값을 가짐
print(L1, ", ", L2, ", ", L) # 반환값이 없는 메소드는 호출 후에 변수에 할당하지 않아야함
print("#####")
s = "Sogang"
L1 = sorted(s)
L2 = list(s)
L2.sort(reverse = True)
print(L1)
print(L2)
print(s)
```

```
L = [1, 9, 4, 2, 8] , M = [9, 8, 4, 2, 1]
L = [1, 2, 4, 8, 9]
#####
[1, 4, 9] , None , [1, 4, 9]
#####
['S', 'a', 'g', 'g', 'n', 'o']
['o', 'n', 'g', 'g', 'a', 'S']
Sogang
>>>
```

Ln: 19 Col: 0