

# 03

## CHAPTER

# 선형 리스트

### 학습목표

- 선형 리스트의 개념을 파악한다.
- 선형 리스트의 데이터 삽입/삭제 원리를 이해한다.
- 파이썬으로 선형 리스트를 조작하는 코드를 작성한다.
- 다항식을 선형 리스트로 표현하는 방법을 학습한다.

SECTION 00 생활 속 자료구조와 알고리즘

SECTION 01 선형 리스트의 기본

SECTION 02 선형 리스트의 간단 구현

SECTION 03 선형 리스트의 일반 구현

SECTION 04 선형 리스트의 응용

연습문제

응용예제



### ■ 선형 리스트란?

- 맛집이나 마트에서 줄을 서는 것처럼 데이터를 일정한 순서로 나열한 것



# Section 01 선형 리스트의 기본

## ■ 선형 리스트의 개념

- 데이터를 일정한 순서로 나열한 자료구조
- 순차 리스트(Ordered List)라고도 함
- 선형 리스트는 입력 순서대로 저장하는 데이터에 적당



그림 3-1 선형 리스트 사용 예

# Section 01 선형 리스트의 기본

## ■ 선형 리스트의 예

- 카톡으로 연락 온 친구를 배열을 이용하여 표현



그림 3-2 배열에 저장된 선형 리스트 예

# Section 01 선형 리스트의 기본

## ■ 선형 리스트의 원리

### ■ 데이터 삽입

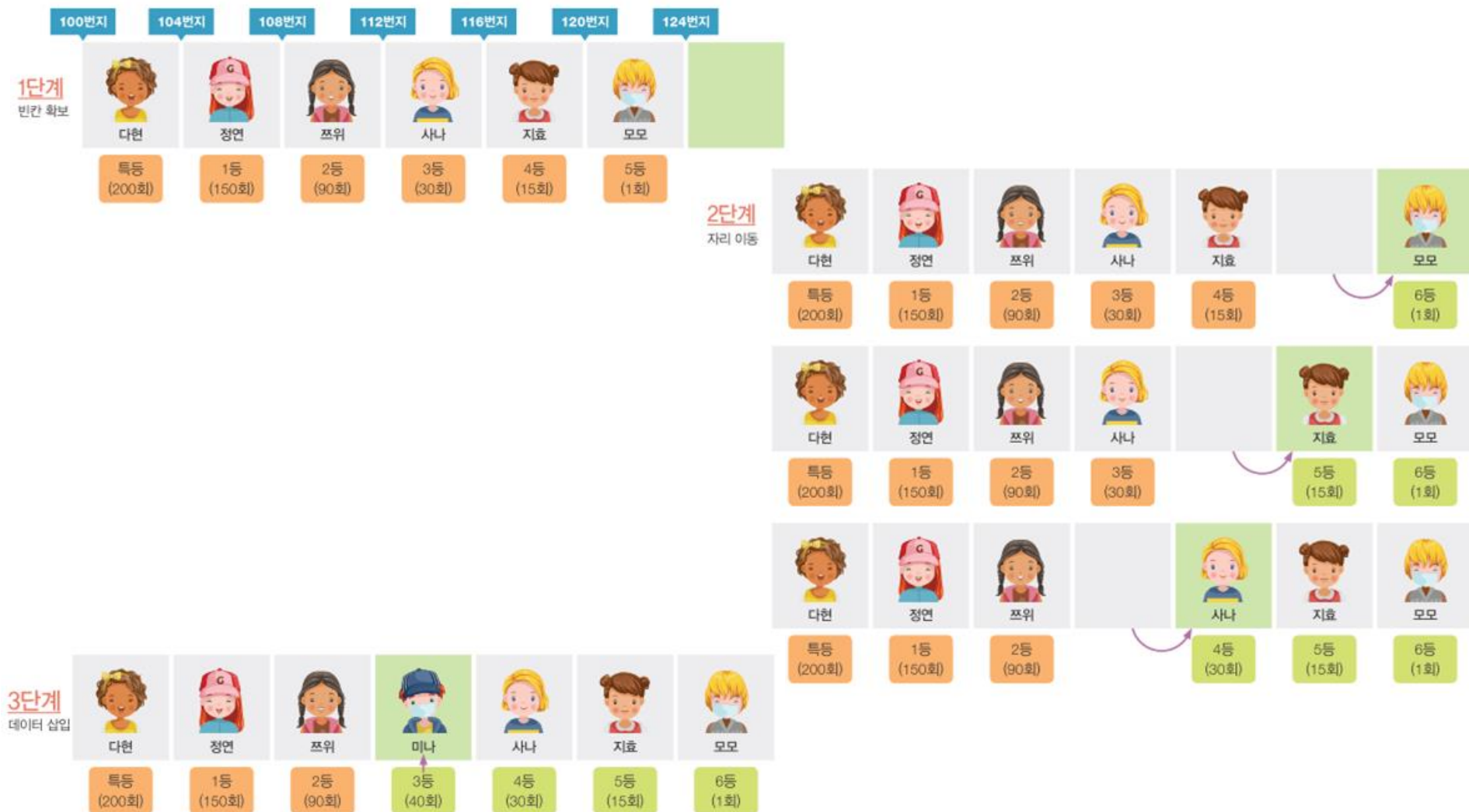


그림 3-3 선형 리스트에서 데이터를 삽입하는 과정

# Section 01 선형 리스트의 기본

여기서 잠깐



파이썬에서  
메모리 주소

실제 메모리에는 [그림 3-2]와 같이 물리적 순서가 아닌 개념적 순서로 저장된다. 메모리의 물리적인 실제 주소는 `id()` 함수로 확인할 수 있는데, 다음 예처럼 주소가 일정하지 않고 파이썬을 실행할 때마다 달라진다.

```
id(katok[0])  
id(katok[1])  
id(katok[2])
```

실행 결과

```
20731568  
20731472  
54080432
```

# Section 01 선형 리스트의 기본

## ■ 선형 리스트의 원리

### ■ 데이터 삭제



그림 3-4 선형 리스트에서 데이터 삭제 과정



## Section 02 선형 리스트의 간단 구현

### ■ 데이터가 5개인 선형 리스트 생성



그림 3-5 생성할 선형 리스트 예

```
katok = ["다현", "정연", "쯔위", "사나", "지효"]  
print(katok)
```

데이터가 5개 있는 배열로 생성

실행 결과

```
['다현', '정연', '쯔위', '사나', '지효']
```

```
print(katok[0]) # 특등  
print(katok[4]) # 4등
```

특등은 첫 번째, 1등은 두 번째, 2등은 세 번째 순으로 들어감

실행 결과

```
다현  
지효
```



## Section 02 선형 리스트의 간단 구현

### ■ 데이터 삽입 : 끝에 삽입

❶ 선형 리스트에 빈칸을 추가한다.



❷ 추가한 빈칸에 모모 데이터를 삽입한다.

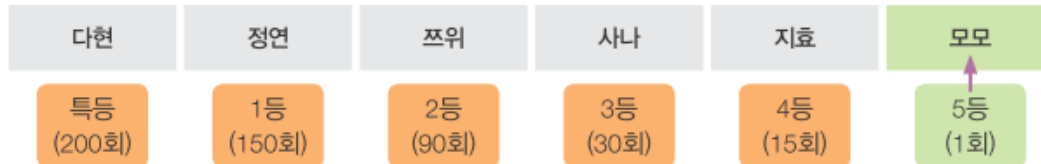


그림 3-6 선형 리스트의 끝에 데이터 삽입

```
❶ katok.append(None)  
print(katok)
```

실행 결과

```
['다현', '정연', '쑤위', '사나', '지효', None]
```

```
❷ katok[5] = "모모"  
print(katok)
```

실행 결과

```
['다현', '정연', '쑤위', '사나', '지효', '모모']
```

# Section 02 선형 리스트의 간단 구현

## ■ 데이터 삽입 : 중간에 삽입

❶ 선형 리스트에 빈칸을 추가한다.

100번지	104번지	108번지	112번지	116번지	120번지	124번지
다현	정연	쯔위	사나	지효	모모	
특등 (200회)	1등 (150회)	2등 (90회)	3등 (30회)	4등 (15회)	5등 (1회)	

❷ 5등 데이터를 마지막 칸인 6등 자리로 옮기고, 원래 5등 자리는 빈칸으로 만든다.

다현	정연	쯔위	사나	지효		모모
특등 (200회)	1등 (150회)	2등 (90회)	3등 (30회)	4등 (15회)		6등 (1회)

❸ 4등 데이터를 5등 자리로 옮기고, 원래 4등 자리는 빈칸으로 만든다.

다현	정연	쯔위	사나		지효	모모
특등 (200회)	1등 (150회)	2등 (90회)	3등 (30회)		5등 (15회)	6등 (1회)

❹ 3등 데이터를 4등 자리로 옮기고, 원래 3등 자리는 빈칸으로 만든다.

다현	정연	쯔위		사나	지효	모모
특등 (200회)	1등 (150회)	2등 (90회)		4등 (30회)	5등 (15회)	6등 (1회)

❺ 3등 자리에 미나를 삽입한다.

다현	정연	쯔위	미나	사나	지효	모모
특등 (200회)	1등 (150회)	2등 (90회)	3등 (40회)	4등 (30회)	5등 (15회)	6등 (1회)

그림 3-7 선형 리스트에 중간 데이터 삽입

```
❶ katok.append(None)
   print(katok)
```

실행 결과

['다현', '정연', '쯔위', '사나', '지효', '모모', None]

```
katok[6] = katok[5]
❷ katok[5] = None # 빈칸으로 만들기
   print(katok)
```

실행 결과

['다현', '정연', '쯔위', '사나', '지효', None, '모모']

```
❸ katok[5] = katok[4]
   katok[4] = None
❹ katok[4] = katok[3]
   katok[3] = None
   print(katok)
```

실행 결과

['다현', '정연', '쯔위', None, '사나', '지효', '모모']

```
❺ katok[3] = "미나"
   print(katok)
```

실행 결과

['다현', '정연', '쯔위', '미나', '사나', '지효', '모모']

# Section 02 선형 리스트의 간단 구현

## ■ 데이터 삭제 : 중간 삭제

❶ 4등 자리에 있던 데이터를 삭제한다.



```
❶ katok[4] = None  
print(katok)
```

실행 결과

['다현', '정연', '쑤위', '미나', None, '지효', '모모']

❷ 5등 자리에 있던 데이터를 4등 자리로 옮긴다.



```
❷ katok[4] = katok[5]  
katok[5] = None  
❸ katok[5] = katok[6]  
katok[6] = None  
print(katok)
```

실행 결과

['다현', '정연', '쑤위', '미나', '지효', '모모', None]

❸ 6등 자리에 있던 데이터를 5등 자리로 옮긴다.



```
❹ del(katok[6])  
print(katok)
```

실행 결과

['다현', '정연', '쑤위', '미나', '지효', '모모']

❹ 빈 6등 자리를 삭제한다.



그림 3-8 선형 리스트의 데이터 삭제

## Section 03 선형 리스트의 일반 구현

### ■ 배열을 이용한 선형 리스트 생성

#### ■ 배열 생성

```
katok = []
```

#### ■ 첫 번째 데이터 입력



❶ 배열 맨 뒤에 빈칸 추가    ❷ 배열 길이 확인 → 1개    ❸ (배열 길이 - 1) 위치에 데이터 삽입

그림 3-9 단순 연결 리스트의 첫 번째 데이터 입력

```
❶ katok.append(None)
❷ 배열길이 = len(katok)
❸ katok[배열길이-1] = '다현'
print(katok)
```

실행 결과

['다현']

## Section 03 선형 리스트의 일반 구현

### ▪ 두 번째 데이터 입력

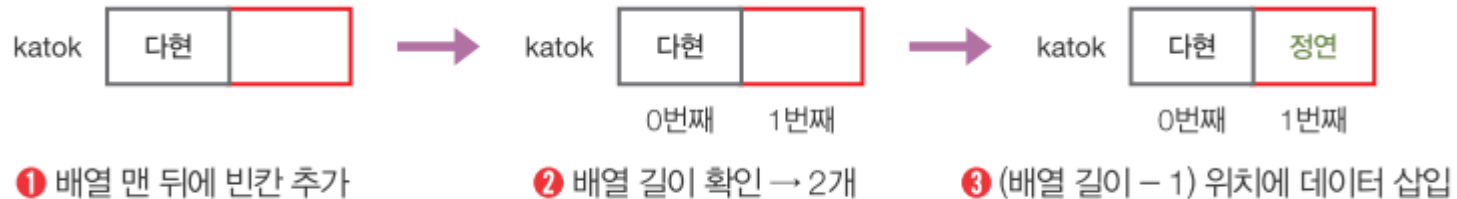


그림 3-10 단순 연결 리스트의 두 번째 데이터 입력

```
① katok.append(None)
② 배열길이 = len(katok)
③ katok[배열길이-1] = '정연'
print(katok)
```

#### 실행 결과

['다현', '정연']

## Section 03 선형 리스트의 일반 구현

### ■ 선형 리스트 생성 함수의 완성

katok	다현	정연	쯔위	사나	지효
-------	----	----	----	----	----

그림 3-11 선형 리스트 예

Code03-01.py 선형 리스트를 생성하는 함수

```
1  katok = []   # 빈 배열
2
3  def add_data(friend) :
4
5      katok.append(None)
6      kLen = len(katok)
7      katok[kLen-1] = friend
8
9  add_data('다현')
10 add_data('정연')
11 add_data('쯔위')
12 add_data('사나')
13 add_data('지효')
14
15 print(katok)
```

실행 결과

['다현', '정연', '쯔위', '사나', '지효']

## Section 03 선형 리스트의 일반 구현

### ■ 데이터 삽입 : 중간 데이터 삽입

0 데이터 삽입 전 초기 상태



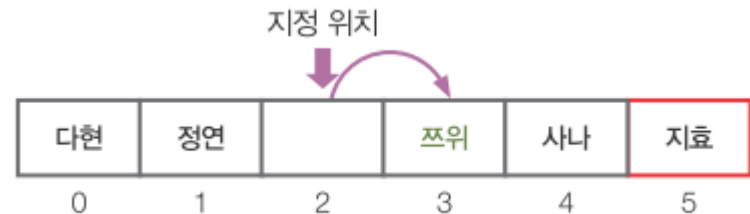
1 선형 리스트에 빈칸을 추가한다.



2 마지막 자리(5번)에 바로 앞 자리(4번)를 넣고, 원래 4번 자리는 빈칸으로 만든다.



3 다시 4번 자리에 3번 자리를 넣고, 원래 3번 자리는 빈칸으로 만든다. 지정 위치에 있던 데이터가 다음 칸으로 이동할 때까지



2 단계를 반복한 후 멈춘다.



## Section 03 선형 리스트의 일반 구현

4 지정 위치(2번)에 술라를 추가한다.



1 `katok.append(None)`

2~3 for 현재위치 in range(마지막위치, 지정위치, -1) :

`katok[현재위치] = katok[현재위치-1]`

`katok[현재위치-1] = None`

4 `katok[지정위치] = friend`

## Section 03 선형 리스트의 일반 구현

### ■ 데이터 삽입 : 맨 끝 데이터 삽입

1 선형 리스트에 빈칸을 추가한다.



2 지정 위치(6번 자리)에 문별을 추가한다.



1 `katok.append(None)`

```
2 for 현재위치 in range(마지막위치, 지정위치, -1) :    # 어차피 작동하지 않음
    katok[현재위치] = katok[현재위치-1]
    katok[현재위치-1] = None
```

```
katok[지정위치] = friend
```

## Section 03 선형 리스트의 일반 구현

### ■ 데이터 삽입 함수의 완성

Code03-02.py 데이터를 삽입하는 함수

```
1  katok = ["다현", "정연", "쯔위", "사나", "지효"]
2
3
4  def insert_data(position, friend) :
5
6      if position < 0 or position > len(katok) :
7          print("데이터를 삽입할 범위를 벗어났습니다.")
8          return
9
10     katok.append(None)          # 빈칸 추가
11     kLen = len(katok)          # 배열의 현재 크기
12
13     for i in range(kLen-1, position, -1) :
14         katok[i] = katok[i-1]
15         katok[i-1] = None
16
17     katok[position] = friend    # 지정한 위치에 친구 추가
18
19
20 insert_data(2, '솔라')
21 print(katok)
22 insert_data(6, '문별')
23 print(katok)
```

**실행 결과**

['다현', '정연', '솔라', '쯔위', '사나', '지효']  
['다현', '정연', '솔라', '쯔위', '사나', '지효', '문별']

## Section 03 선형 리스트의 일반 구현

여기서 잠깐



반복문이 한 번도  
실행되지 않는  
경우

다음과 같이 for 문에서 range(값1, 값2, 값3)의 값 1과 값 2가 같으면 한번도 수행하지 않는다.

```
for i in range(6, 6, -1) :  
    print("OK!")
```

```
for i in range(6, 6, 1) :  
    print("Ok!")
```

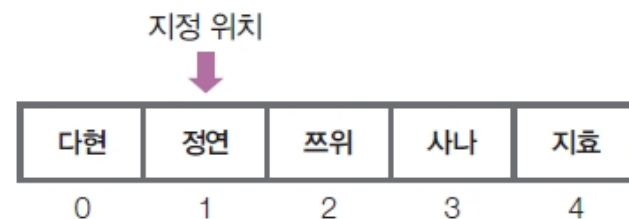
**실행 결과**

아무것도 출력 안 됨

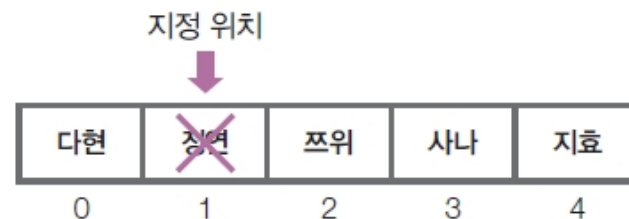
## Section 03 선형 리스트의 일반 구현

### ■ 데이터 삭제 : 중간 데이터 삭제

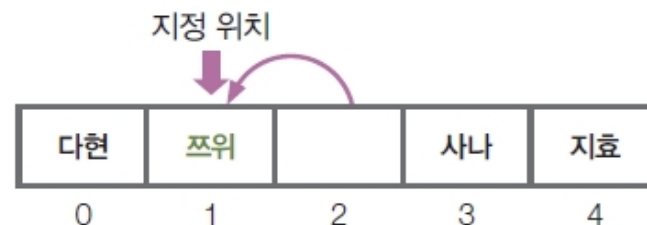
0 중간에 데이터가 삭제되기 전 초기 상태



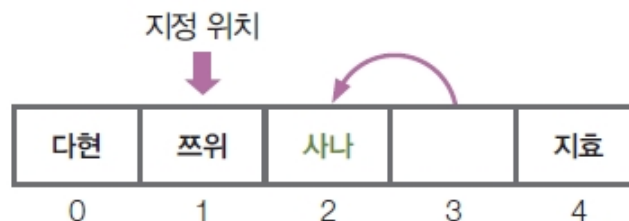
1 지정 위치의 데이터를 삭제한다.



2 지정 위치 다음(2번 자리)에 있는 데이터를 지정 위치(1번 자리)로 이동하고, 2번은 빈칸으로 만든다.



3 다시 3번 자리를 2번 자리로 이동하고, 3번 자리는 빈칸으로 만든다. 배열의 맨 마지막 위치가 앞 칸으로 이동할 때까지 2 단계를 반복한 후 멈춘다.



## Section 03 선형 리스트의 일반 구현

4 배열 맨 마지막 위치를 완전히 제거한다.



```
1 katok[지정위치] = None

2~3 for 현재위치 in range(지정위치+1, 마지막위치+1) :
    katok[현재위치-1] = katok[현재위치]
    katok[현재위치] = None

4 del(katok[지정위치])
```

## Section 03 선형 리스트의 일반 구현

### ■ 데이터 삭제 : 맨 마지막 데이터 삭제

**0** 마지막 데이터가 삭제되기 전 초기 상태



**1** 지정 위치의 데이터를 삭제한다.



**2** 지정 위치(=배열의 맨 마지막 위치)를 완전히 제거한다.



**1** `katok[지정위치] = None`

```
for 현재위치 in range(지정위치+1, 마지막위치+1) :  
    katok[현재위치-1] = katok[현재위치]  
    katok[현재위치] = None
```

**2** `del(katok[지정위치])`



## Section 03 선형 리스트의 일반 구현

### ■ 데이터 삭제 : 데이터 삭제 함수의 완성

Code03-03.py 데이터를 삭제하는 함수

```
1 katok = ['다현', '정연', '쯔위', '사나', '지효']
2
3
4 def delete_data(position) :
5
6     if position < 0 or position > len(katok) :
7         print("데이터를 삭제할 범위를 벗어났습니다.")
8         return
9
10    kLen = len(katok)
11    katok[position] = None    # 데이터 삭제
12
13    for i in range(position+1, kLen) :
14        katok[i-1] = katok[i]
15        katok[i] = None      # 배열의 맨 마지막 위치 삭제
16
17    del(katok[kLen-1])
18
19
20 delete_data(1)
21 print(katok)
22 delete_data(3)
23 print(katok)
```

#### 실행 결과

['다현', '쯔위', '사나', '지효']

['다현', '쯔위', '사나']

## Section 03 선형 리스트의 일반 구현

### ■ 기본 선형 리스트의 완성

Code03-04.py 선형 리스트 처리 프로그램

```
1  ## 함수 선언 부분 ##
2  def add_data(friend) :
3
4      katok.append(None)
5      kLen = len(katok)
6      katok[kLen-1] = friend
7
8
9  def insert_data(position, friend) :
10
11      # 생략(Code03-02.py의 6~17행과 동일)
12
13
14
15
16
17
18
19
20
21
22
23
24
25  def delete_data(position) :
26
27      # 생략(Code03-03.py의 6~17행과 동일)
28
29
30
31
32
33
34
35
36
37
38
39
40
41  ## 전역 변수 선언 부분 ##
42  katok = []
43  select = -1  # 1: 추가, 2: 삽입, 3: 삭제, 4: 종료
44
45
```

## Section 03 선형 리스트의 일반 구현

```
46 ## 메인 코드 부분 ##
47 if __name__ == "__main__" :
48
49     while (select != 4) :
50
51         select = int(input("선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> "))
52
53         if (select == 1) :
54             data = input("추가할 데이터--> ")
55             add_data(data)
56             print(katok)
57         elif (select == 2) :
58             pos = int(input("삽입할 위치--> "))
59             data = input("추가할 데이터--> ")
60             insert_data(pos, data)
61             print(katok)
62         elif (select == 3) :
63             pos = int(input("삭제할 위치--> "))
64             delete_data(pos)
65             print(katok)
66         elif (select == 4) :
67             print(katok)
68             exit
69         else :
70             print("1~4 중 하나를 입력하세요.")
71             continue
```

## Section 03 선형 리스트의 일반 구현

### 실행 결과

선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> 1

추가할 데이터--> 다현

['다현']

선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> 1

추가할 데이터--> 정연

['다현', '정연']

선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> 1

추가할 데이터--> 찰위

['다현', '정연', '찰위']

선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> 2

삽입할 위치--> 1

추가할 데이터--> 하나

['다현', '하나', '정연', '찰위']

선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> 2

삽입할 위치--> 0

추가할 데이터--> 문별

['문별', '다현', '하나', '정연', '찰위']

선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> 3

삭제할 위치--> 3

['문별', '다현', '하나', '찰위']

선택하세요(1: 추가, 2: 삽입, 3: 삭제, 4: 종료)--> 4

['문별', '다현', '하나', '찰위']

## Section 04 선형 리스트의 응용

### ■ 다항식의 선형 리스트 표현

- 선형 리스트의 가장 많은 응용 중 하나는 다항식을 저장하고 활용하는 것
- $P(x)$ 를 다항식이라고 하며  $a, b, c, \dots, z$  등을 계수,  $x$ 를 변수,  $x$ 의  $1, 2, \dots, n$  등을 지수

$$P(x) = a + bx + cx^2 + dx^3 + \dots + zx^n$$

### ■ 최고차 항부터 나열

①  $P(x) = 7x^3 - 4x^2 + 5$

②  $P(x) = 7x^3 - 4x^2 + 0x^1 + 5x^0$

①에서 생략한 항까지 모두 표현하면 ②와 같음

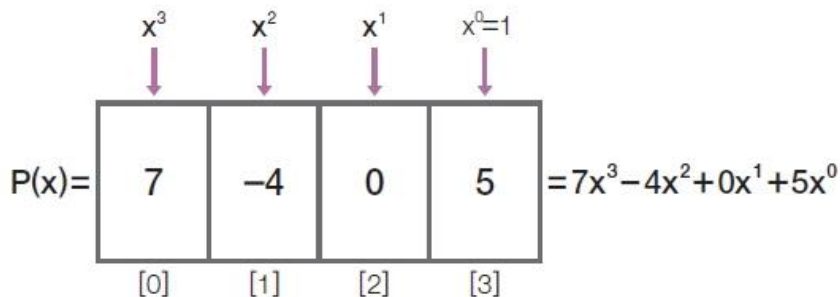


그림 3-12 다항식 예의 선형 리스트로 표현

## Section 04 선형 리스트의 응용

```
px = [7, -4, 0, 5]
print(px)
```

실행 결과

[7, -4, 0, 5]

```
x = 2
pxVal = 7*x**3 - 4*x**2 + 0*x**1 + 5*x**0
print(pxVal)
```

x값이 2라면 파이썬 수식으로 계산 가능

실행 결과

45

### ■ 다항식 형태 출력

```
px = [7, -4, 0, 5]
polyStr = "P(x) = "
polyStr += " + " + str(px[0]) + "x^" + str(3)
polyStr += " + " + str(px[1]) + "x^" + str(2)
polyStr += " + " + str(px[2]) + "x^" + str(1)
polyStr += " + " + str(px[3]) + "x^" + str(0)
print(polyStr)
```

실행 결과

P(x) = + 7x^3 + -4x^2 + 0x^1 + 5x^0

## Section 04 선형 리스트의 응용

### ■ 다항식의 선형 리스트 표현과 계산 프로그램

- printPoly( ) 함수와 calcPoly( ) 함수를 생성해서 코드 작성

Code03-05.py 다항식 선형 리스트 표현과 계산 프로그램

```
1  ## 함수 선언 부분 ##
2  def printPoly(p_x) :
3      term = len(p_x) - 1      # 최고차항 숫자 = 배열 길이-1
4      polyStr = "P(x) = "
5
6      for i in range(len(p_x)) :
7          coef = p_x[i]        # 계수
8
9          if (coef >= 0) :
10             polyStr += "+"
11             polyStr += str(coef) + "x^" + str(term) + " "
12             term -= 1
13
14     return polyStr
15
16
17 def calcPoly(xVal, p_x) :
18     retValue = 0
19     term = len(p_x) - 1      # 최고차항 숫자 = 배열 길이-1
20
```



## Section 04 선형 리스트의 응용

```
21     for i in range(len(px)) :
22         coef = p_x[i]          # 계수
23         retValue += coef * xValue ** term
24         term -= 1
25
26     return retValue
27
28
29 ## 전역 변수 선언 부분 ##
30 px = [7, -4, 0, 5]             # = 7x^3 -4x^2 +0x^1 +5x^0
31
32
33 ## 메인 코드 부분 ##
34 if __name__ == "__main__" :
35
36     pStr = printPoly(px)
37     print(pStr)
38
39     xValue = int(input("X 값-->"))
40
41     pxValue = calcPoly(xValue, px)
42     print(pxValue)
```

실행 결과

$P(x) = + 7x^3 - 4x^2 + 0x^1 + 5x^0$

X 값-->2

45

## Section 04 선형 리스트의 응용

### SELF STUDY 3-2

Code03-05.py의 printPoly( ) 함수를 수정해서 계수가 0인 항은 출력되지 않도록 하자. 또 첫 번째 항의 + 부호도 출력되지 않도록 한다. 즉, 다음 결과를 만들자.

실행 결과

$$P(x) = 7x^3 - 4x^2 + 5x^0$$

## Section 04 선형 리스트의 응용

### ■ 특수 다항식 처리 프로그램

- 지수가 상당히 큰 다항식의 처리

❶  $P(x) = 7x^{300} - 4x^{20} + 5$

❶에서 생략한 항까지 모두 표현하면 ❷와 같음

❷  $P(x) = 7x^{300} + 0x^{299} + 0x^{298} + 0x^{297} + \dots + 0x^{21} - 4x^{20} + 0x^{19} + 0x^{18} + 0x^{17} + \dots + 5x^0$

`px = [7, 0, 0, 0, ..., -4, 0, 0, 0, ..., 5]`

# 총 301개

Code03-05.py 코드로 적용

`tx = [300, 20, 0]`

# 항 차수

`px = [7, -4, 5]`

# 각 항 위치의 계수

0이 아닌 계수와  
항의 차수만 저장

Code03-06.py 특수 다항식의 선형 리스트 표현과 계산 프로그램

```
1  ## 함수 선언 부분 ##
2  def printPoly(t_x, p_x):
3      polyStr = "P(x) = "
4
5      for i in range(len(p_x)):
6          term = t_x[i]          # 항 차수
7          coef = p_x[i]          # 계수
8
9          if (coef >= 0):
10             polyStr += "+"
11             polyStr += str(coef) + "x^" + str(term) + " "
12
```

## Section 04 선형 리스트의 응용

```
13     return polyStr
14
15
16 def calcPoly(xVal, t_x, p_x) :
17     retValue = 0
18
19     for i in range(len(px)) :
20         term = t_x[i]          # 항 차수
21         coef = p_x[i]          # 계수
22         retValue += coef * xValue ** term
23         term -= 1
24
25     return retValue
26
27
28 ## 전역 변수 선언 부분 ##
29 tx = [300, 20, 0]
30 px = [7, -4, 5]
31
32
33 ## 메인 코드 부분 ##
34 if __name__ == "__main__" :
35
36     pStr = printPoly(tx, px)
37     print(pStr)
38
39     xValue = int(input("X 값-->"))
```

## Section 04 선형 리스트의 응용

```
40
41     pxValue = calcPoly(xValue, tx, px)
42     print(pxValue)
```

### 실행 결과

$P(x) = +7x^{300} - 4x^{20} + 5x^0$

X 값-->1

8

# 응용예제 01 카톡 친구 자동 삽입하기

난이도★☆☆☆

## 예제 설명

카톡 친구 이름과 카톡 횟수를 입력하면 자동으로 위치를 찾아 삽입하는 프로그램이다. 카톡 친구의 초기 정보는 [그림 3-2]의 정보를 모두 저장하도록 ('친구이름', 연락횟수) 튜플 리스트로 시작한다.

```
[('다현', 200), ('정연', 150), ('쯔위', 90), ('사나', 30), ('지효', 15)]
```

새로운 친구 '미나'와 40회를 입력하면 자동으로 자신의 순위에 해당하는 위치를 찾아 삽입된다. 동일한 연락 횟수라면 새로운 친구를 앞 순위로 지정한다.

```
[('다현', 200), ('정연', 150), ('쯔위', 90), ('미나', 40), ('사나', 30), ('지효', 15)]
```

## 실행 결과



```
Python Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookData\WEx03-01.py =====
추가할 친구--> 미나
카톡 횟수--> 40
[('다현', 200), ('정연', 150), ('쯔위', 90), ('미나', 40), ('사나', 30), ('지효', 15)]
추가할 친구--> 해리
카톡 횟수--> 200
[('해리', 200), ('다현', 200), ('정연', 150), ('쯔위', 90), ('미나', 40), ('사나', 30), ('지효', 15)]
추가할 친구--> |
```

## 예제 설명

Code03-06.py '특수 다항식의 선형 리스트 표현과 계산 프로그램'에서 29~30행에 있는 차수 배열과 계수 배열을 2차원 배열로 지정한 후 결과가 동일하게 나오도록 하자.

## 실행 결과

A screenshot of a Python shell window titled 'Python'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following content: a restart message 'RESTART: C:\CookData\WEx03-02.py', a polynomial definition 'P(x) = +7x^300 -4x^20 +5x^0', an input prompt 'X 값-->1', the input value '8', and the prompt '>>>'. The status bar at the bottom right indicates 'Ln: 8 Col: 4'.

```
Python
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookData\WEx03-02.py =====
P(x) = +7x^300 -4x^20 +5x^0
X 값-->1
8
>>>
```