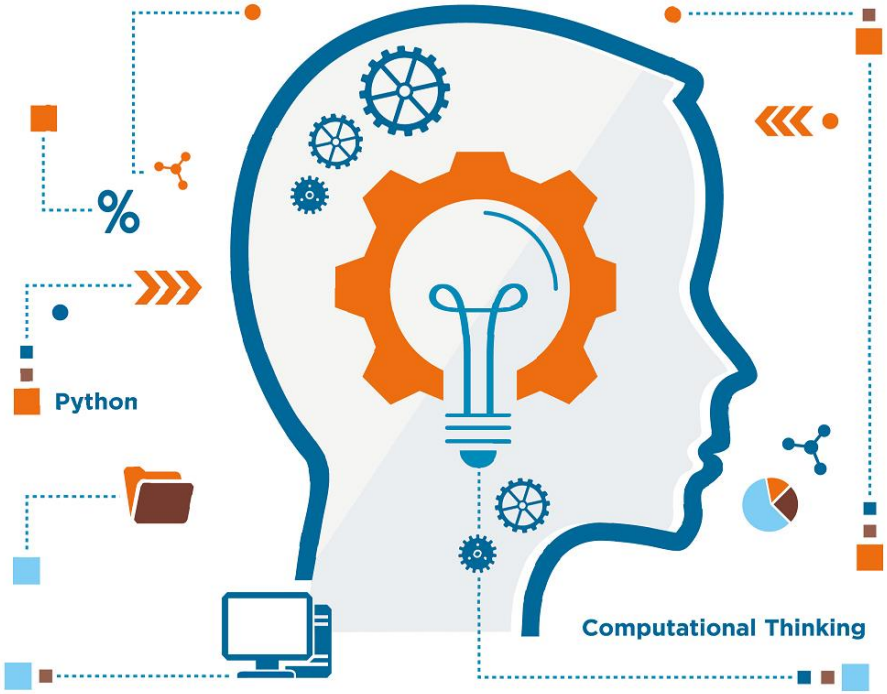


Chapter 07. 리스트

목차

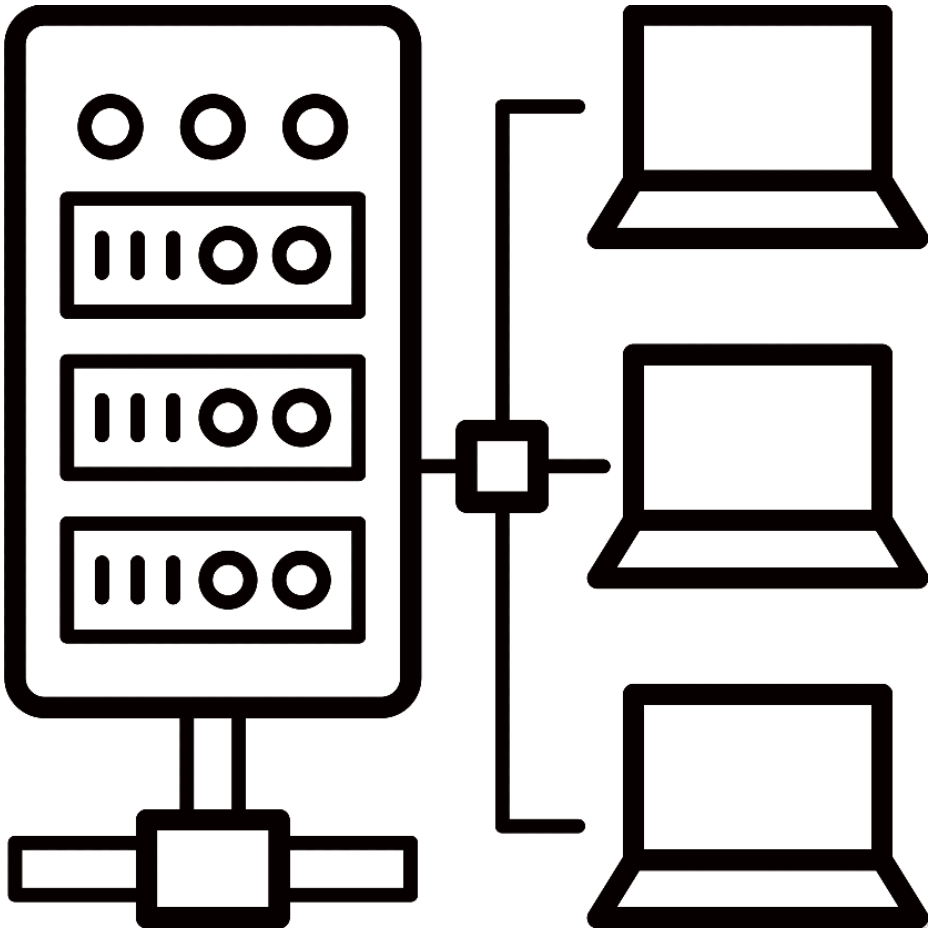
01

리스트의 개념

01. 리스트의 개념

[데이터 관리의 중요성]

- 많은 데이터를 간편하게 다루기 위한 데이터 구조가 필요하다.
- 파이썬에는 리스트와 튜플이 있다.



01. 리스트의 개념

I. 리스트의 필요성

- 학생이 10명이면 10개의 변수가 필요하고, 이름도 각각 다르게 만들어야 함
- 학생이 100명이라면? 저장할 값이 많아질수록 이런 방식으로 변수를 늘려가는 방법은 사용하기 어려워짐 → 변수를 묶어서 관리하는 리스트가 필요!

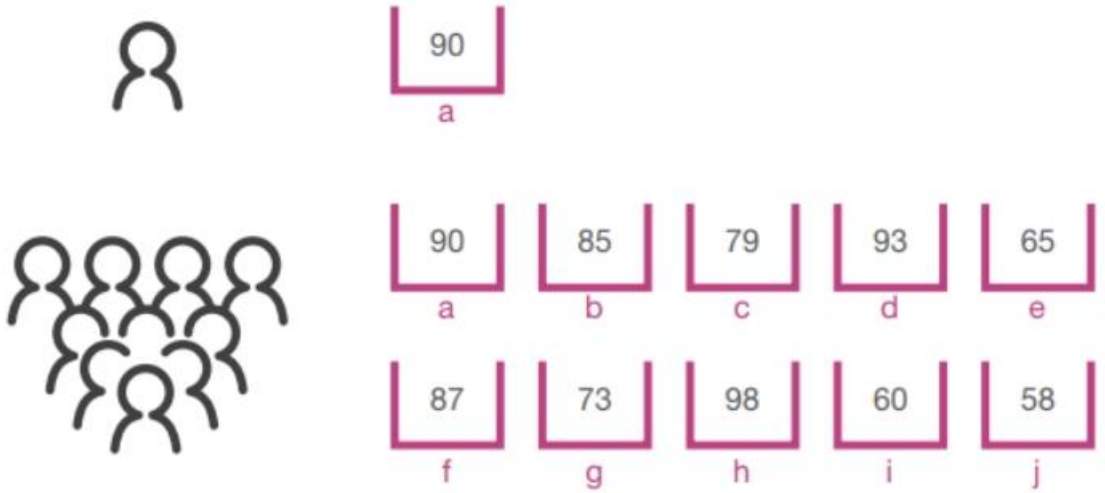


그림 7-1 리스트의 필요성

01. 리스트의 개념

II. 리스트의 구조

- 컬렉션 : 많은 데이터를 처리할 때 변수처럼 개별 요소로 처리하는 대신, 그룹으로 묶어서 처리하기 위한 데이터 구조를 지원
- 리스트, 튜플, 딕셔너리 등이 있음
- 리스트는 여러 개의 데이터를 하나의 이름으로 저장하는 가장 일반적인 형태
- 리스트에 저장되는 각각의 데이터를 항목 또는 원소라고 하며, 항목은 숫자나 문자 , 다른 리스트 등 다양한 종류로 구성
- 리스트를 만드는 방법은 대괄호([ ])를 붙이고 항목 간에는 쉼표(,)로 구분해서 나열
- 각 항목은 순서대로 인덱스라고 하는 번호가 정해지며 0부터 시작

```
리스트명 = [항목1, 항목2, 항목3, 항목4, 항목5, ... 항목n]
인덱스   → [0]    [1]    [2]    [3]    [4]    ... [n-1]
```

```
a = [90, 85, 79, 93, 65, 87, 73, 98, 60, 58]
b = ['홍길동', '김유신', 3.5, 6, ['a', 'b']]
```

II. 리스트의 구조

실습 7-1 베스트셀러 도서의 판매 합계와 평균 구하기 code07-01.py

- 베스트셀러 도서 5권의 판매 수량 합계와 평균을 계산

① 변수 이름을 각각 다르게 정의해야 하고, 합 계를 구할 때는 모든 변수를 덧셈 연산자로 연결

```
01  b1 = 234
02  b2 = 82
03  b3 = 128
04  b4 = 50
05  b5 = 155
06
07  total = b1 + b2 + b3 + b4 + b5          # 변수를 이용한 합계 구하기
08  print("판매 수량 합계 =", total)
09  print("판매 수량 평균 =", total / 5)
```

01. 리스트의 개념

II. 리스트의 구조

실습 7-1 베스트셀러 도서의 판매 합계와 평균 구하기 code07-01.py

② 변수 대신 리스트를 사용하면 이름을 일일이 정의하는 번거로움을 줄이고, 합계를 구할 때도 반복문을 사용

```
01  books = [234, 82, 128, 50, 155]
02  total = 0
03
04  for x in books :                    # 리스트를 이용한 합계 구하기
05      total += x
06  print("판매 수량 합계 =", total)
07  print("판매 수량 평균 =", total / len(books))
```

③

```
판매 수량 합계 = 649
판매 수량 평균 = 129.8
```

01. 리스트의 개념

II. 리스트의 구조

여기서 잠깐 **in 연산자**

- in 연산자는 단독으로 사용되기도 하는데, 다음과 같이 특정 항목이 리스트에 있는지를 검사해서 True나 False로 결과를 알려줌

```
>>> alist = ['홍길동', '김유신', 3]
>>> '이순신' in alist          # '이순신'이 리스트에 있으면(in) True를 반환
False
>>> 5 not in alist             # 5가 리스트에 없으면(not in) True를 반환
True
```

02

리스트의 사용

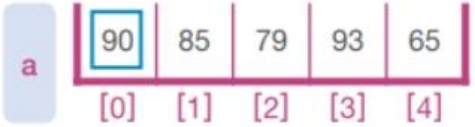
02. 리스트의 사용

I. 리스트 인덱싱

- 인덱스를 이용하여 해당 항목의 값을 가 져오거나 수정하는 작업

```
>>> a = [90, 85, 79, 93, 65]
```


```
>>> a[0]
90
```




```
>>> a[3] = 'xy'
>>> a
[90, 85, 79, 'xy', 65]
```

그림 7-3 인덱스를 사용한 리스트 항목의 참조

실습 7-2 리스트의 인덱싱 다양하게 활용하기

① 항목을 입력해서 리스트를 만들고, 만든 리스트의 데이터형을 type( ) 함수로 확인

```
>>> alist = ['홍길동', '김유신', 3.5, 6, ['a', 'b']]     # 또 다른 리스트인 ['a', 'b']도 포함
>>> type(alist)
```

02. 리스트의 사용

I. 리스트 인덱싱

실습 7-2 리스트의 인덱싱 다양하게 활용하기

② 인덱스를 이용해 항목의 값을 출력

```
>>> alist[1]                # 두 번째 항목의 참조
'김유신'
>>> len(alist)              # 전체 항목 개수
5
>>> alist[5]                # 인덱스의 범위(0~4)를 벗어난 사용으로 에러 발생
Traceback (most recent call last):
    File "<pyshell#16>", line 1, in <module>
        alist[5]
IndexError: list index out of range
```

③ alist의 마지막 항목은 리스트 유형

```
>>> alist[4]
['a', 'b']
>>> alist[4][0]
'a'
>>> alist[4][1]
'b'
```

그림 7-4 리스트 중첩

02. 리스트의 사용

I. 리스트 인덱싱

실습 7-2 리스트의 인덱싱 다양하게 활용하기

④ 인덱스는 음수를 사용할 수도 있는데, 마지막 항목의 인덱스는 [–1]

```
>>> alist[-1]
['a', 'b']
>>> alist[-2]
6
>>> alist[-5]
'홍길동'
```

그림 7–5 음수를 사용한 인덱스

⑤ 인덱싱으로 항목의 값을 바꾸거나 수식에도 사용

```
>>> alist[2] = 7.8
>>> alist[2] + alist[3]
13.8
```

그림 7–6 항목의 값 바꾸기

02. 리스트의 사용

I. 리스트 인덱싱

실습 7-2 리스트의 인덱싱 다양하게 활용하기

⑥ 덧셈(+)과 곱셈(*) 연산자에서는 리스트 자체를 피연산자로 하는 연산을 수행

```
>>> alist + alist
['홍길동', '김유신', 7.8, 6, ['a', 'b'], '홍길동', '김유신', 7.8, 6, ['a', 'b']]
>>> alist * 3
['홍길동', '김유신', 7.8, 6, ['a', 'b'], '홍길동', '김유신', 7.8, 6, ['a', 'b'], '홍길동', '김유
신', 7.8, 6, ['a', 'b']]
```

⑦ 리스트를 출력하려면 리스트 이름을 사용해서 전체 항목을 한 번에 출력하거나 반복문으로 항목을 하나씩 출력

```
>>> print(alist)
['홍길동', '김유신', 7.8, 6, ['a', 'b']]
>>> for x in alist :
        print(x)
홍길동
김유신
7.8
6
['a', 'b']
```

print(x) 입력 후 Enter 를 2번 누르면 실행

02. 리스트의 사용

I. 리스트 인덱싱

여기서 잠깐 **2차원 리스트**

- 2차원 리스트를 for 반복문으로 참조할 때는 다음과 같이 2개의 변수를 사용할 수 있음

	열	
행	'A'	3
	'B'	5
	'C'	7

```
>>> multi = [['A', 3], ['B', 5], ['C', 7]]
>>> for x, y in multi :
        print(x, '=', y)
A = 3
B = 5
C = 7
```

그림 7-7 2차원 리스트 사용법

02. 리스트의 사용

II. 항목의 추가와 삭제

- 항목 추가 메소드

```
리스트명.append(항목)                    # 리스트의 맨 뒤에 삽입
리스트명.insert(위치, 항목)              # 특정 위치에 항목을 삽입
```

- 항목 위치 검색 메소드

```
리스트명.index(항목)                     # 특정 항목의 위치 찾기
```

- 항목 삭제 메소드

```
리스트명.remove(항목)                    # 특정 항목(값)을 삭제
del(리스트명[위치]) 또는 del 리스트명[위치]  # 특정 위치에 있는 항목을 삭제
리스트명.pop()                           # 맨 뒤의 항목 하나를 삭제
리스트명.clear()                         # 모든 항목 삭제
```

02. 리스트의 사용

II. 항목의 추가와 삭제

실습 7-3 리스트에 항목 추가하기

① 빈 리스트를 만들고 리스트의 내용을 확인

```
>>> alist = []
>>> alist
[]
```

그림 7-8 빈 리스트 생성

② 리스트에 항목을 추가하고, 변경된 리스트를 확인

```
>>> alist.append(30)
>>> alist.append('홍길동')
>>> alist
[30, '홍길동']
```

그림 7-9 실행 순서대로 항목 추가

③ 리스트의 특정 위치에 항목을 추가하려면 insert( ) 메소드를 사용

```
>>> alist.insert(1, '김유신')
>>> alist
[30, '김유신', '홍길동']
>>> alist.index('김유신')
1
```

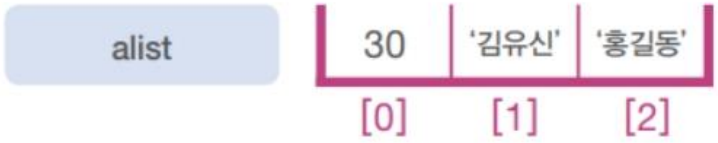


그림 7-10 위치를 지정하여 항목 추가

II. 항목의 추가와 삭제

실습 7-4 리스트에서 항목 삭제하기

① remove( ) 메소드를 사용하면 리스트에서 특정 항목을 삭제

```
>>> alist.remove('홍길동')
>>> alist
[30, '김유신']
```

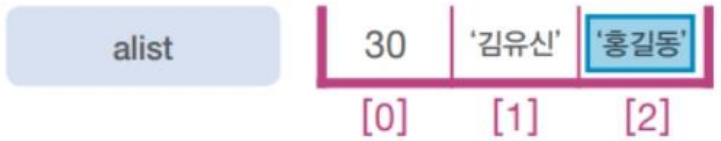


그림 7-11 특정 항목 삭제

② 리스트에서의 위치를 이용해 항목을 삭제하려면 del( ) 메소드를 사용

```
>>> del(alist[0])
>>> alist
['김유신']
```

그림 7-12 특정 위치에 있는 항목 삭제

II. 항목의 추가와 삭제

실습 7-4 리스트에서 항목 삭제하기

③ pop( ) 메소드는 리스트의 가장 마지막 항목을 삭제하는데, 마지막 항목을 꺼내어 다른 변수에 저장

```
>>> deleted = alist.pop()
>>> alist
[]
>>> deleted          # 삭제한 항목이 저장된 변수
'김유신'
```

그림 7-13 마지막 항목 삭제

④ 리스트의 모든 항목을 한 번에 삭제하려면 clear( ) 메소드를 사용

```
>>> alist = [1, 2, 3, 4, 5]
>>> alist.clear()
>>> alist
[]
```

II. 항목의 추가와 삭제

여기서 잠깐 **clear()와 del()의 차이**

- 리스트 항목을 모두 삭제하여 빈 리스트로 만드는 clear()와는 달리 del() 함수는 객체 자체를 메모리에서 삭제

```
>>> alist = [1, 2, 3, 4, 5]
>>> del(alist)
>>> alist
Traceback (most recent call last):
    File "<pyshell#13>", line 1, in <module>
        alist
NameError: name 'alist' is not defined
```

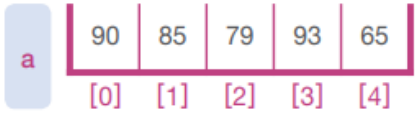
02. 리스트의 사용

III. 리스트 슬라이싱

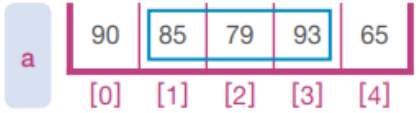
- 범위로 여러 개의 항목을 참조하는 것, 범위 표시에는 콜론(:) 기호를 사용

```
리스트명[인덱스1:인덱스2] # 인덱스1부터 (인덱스2 앞)까지 추출
리스트명[인덱스1:]        # 인덱스1부터 마지막까지 추출
리스트명[:인덱스2]        # 첫 번째부터 (인덱스2 앞)까지 추출
리스트명[:]               # 리스트의 전체 항목 추출
```

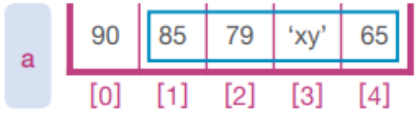
```
>>> a = [90, 85, 79, 93, 65]
```



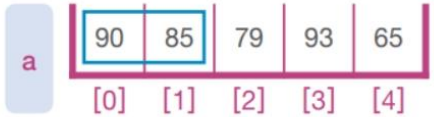

```
>>> a[1:4]
[85, 79, 93]
```




```
>>> a[1:]
[85, 79, 93, 65]
```




```
>>> a[:2]
[90, 85]
```




```
>>> a[:]
[90, 85, 79, 93, 65]
```

그림 7-14 다양한 슬라이싱 방법

02. 리스트의 사용

III. 리스트 슬라이싱

실습 7-5 리스트 슬라이싱으로 항목 추출, 변경, 삭제하기

① 항목을 입력해서 리스트를 만들고, 여러 방법으로 슬라이싱

```
>>> blist = ['정약용', '유관순', 50]
>>> blist[1:2]
['유관순']
>>> blist[:2]
['정약용', '유관순']
>>> blist[2:]
[50]
>>> blist[:]
['정약용', '유관순', 50]
```

슬라이싱의 결과는 항목이 하나이더라도 항상 리스트임
참고로, blist[1]로 인덱싱하면 문자열 '유관순'으로 출력

III. 리스트 슬라이싱

실습 7-5 리스트 슬라이싱으로 항목 추출, 변경, 삭제하기

② 슬라이싱을 사용해서 리스트 항목을 다른 값으로 변경 가능

```
>>> blist; len(blist)              # 현재의 리스트 값과 개수 확인
['정약용', '유관순', 50]
3
>>> blist[1:2] = [30, '이순신']    # '유관순'을 2개의 항목으로 변경
>>> blist
['정약용', 30, '이순신', 50]
>>> len(blist)                     # 항목의 개수가 4로 증가
4
```

③ 슬라이싱을 이용하면 여러 항목을 한 번에 삭제할 수도 있음

```
>>> blist                          # 현재의 리스트 값 확인
['정약용', 30, '이순신', 50]
>>> del(blist[2:])                 # 세 번째 항목부터 끝까지 삭제
>>> blist                          # 삭제 후 리스트 값 확인
['정약용', 30]
```

02. 리스트의 사용

IV. 리스트 정렬

- 글자 순서나 값의 크기를 기준으로 나열

```
리스트명.sort()                  # 오름차순 정렬(1, 2, 3 혹은 'a', 'b', 'c')
리스트명.sort(reverse=True)      # 내림차순 정렬(3, 2, 1 혹은 'c', 'b', 'a')
sorted(리스트명)                 # 정렬된 리스트를 반환
```

실습 7-6 리스트에서 sort() 함수 활용하기

① 리스트를 만들고 데이터를 저장, sort( ) 메소드로 항목을 정렬한 다음 결과를 확인

```
>>> contact = ['정약용', '이순신', '김유신', '유관순']
>>> contact.sort()
>>> contact
['김유신', '유관순', '이순신', '정약용']
```

② 리스트의 정렬 방식은 오름차순을 기본으로 하므로, 내림차순으로 정렬하려면 'reverse' 옵션을 사용

```
>>> contact.sort(reverse=True)
>>> contact
['정약용', '이순신', '유관순', '김유신']
```

IV. 리스트 정렬

실습 7-6 리스트에서 sort() 함수 활용하기

③ sorted( ) 함수는 인수로 사용한 numbers 리스트는 바꾸지 않고 다른 리스트 alist에 정렬 결과를 저장

```
>>> numbers = [78, 45, 90, 88, 62]
>>> alist = sorted(numbers)
>>> numbers                          # 항목이 정렬되지 않은 원본 리스트
[78, 45, 90, 88, 62]
>>> alist                            # 정렬된 결과를 반환받은 리스트
[45, 62, 78, 88, 90]
```

03

리스트의 활용

03. 리스트의 활용

실습 7-7 리스트로 점수의 평균 구하기 code07-07.py

- 리스트를 이용하여 수강생 5명의 점수를 입력받아 평균을 출력하는 프로그램

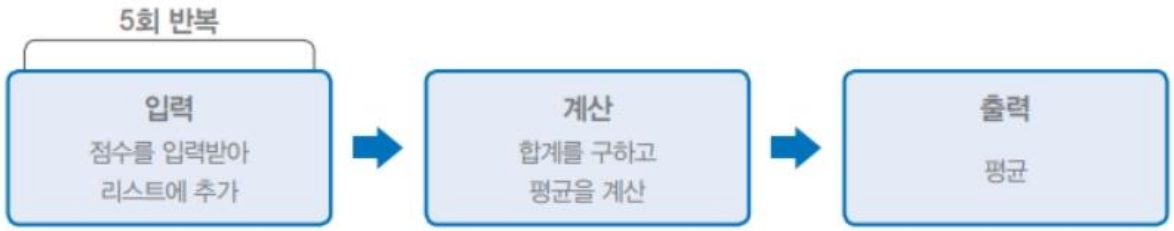
① 

그림 7-15 5명의 점수 입력 후 평균 구하기

② 리스트의 합계를 구하는 함수 sum( )과 리스트 항목의 개수를 구하는 len( )을 사용해서 간단하게 평균을 구할 수 있음

```
01  numbers = []
02
03  for _ in range(5):
04      numbers.append(int(input("점수 입력 : ")))
05
06  print("평균 =", sum(numbers) / len(numbers))
```

③ 프로그램 실행하고 결과 확인

03. 리스트의 활용

실습 7-8 도서 목록 만들기 code07-08.py

- 읽은 도서명을 모두 입력하면 정렬해서 가나다순으로 출력하는 프로그램

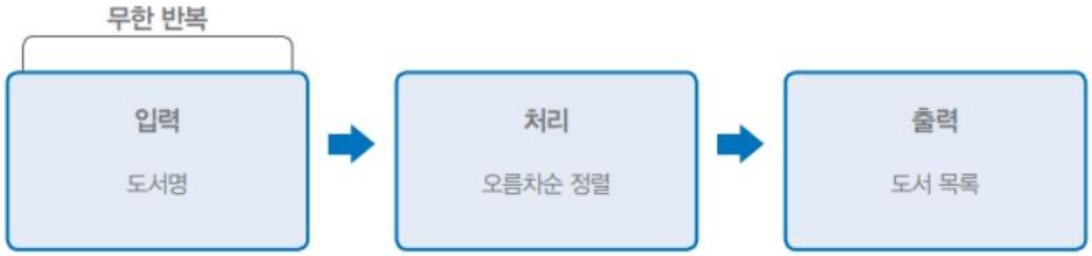
① 

그림 7-17 도서 목록 만들고 정렬하기

② 무한 반복되는 입력 과정을 종료하기 위해 입력 값이 빈 문자열인지 판단하는 if 문을 사용

```
01  book = ''                       # 입력하는 도서명을 저장하는 변수
02  bookList = []                   # 도서 목록을 저장하는 리스트
03  number = 0                      # 출력할 때 표시할 순서 번호
04
05  print("입력을 종료하려면 [Enter] 키를 누르세요.")
06  print('=' * 30)
07
08  while True :
09      book = input("도서명 입력 : ")
10      if book == '' :             # 도서명을 입력하지 않고 Enter를 눌렀을 때, 빈 문자열로 판단
```

03. 리스트의 활용

실습 7-8 도서 목록 만들기 code07-08.py

② 무한 반복되는 입력 과정을 종료하기 위해 입력 값이 빈 문자열인지 판단하는 if 문을 사용

```
11            break
12        bookList.append(book)
13
14    bookList.sort()
15
16    print('=' * 30)
17    for b in bookList :
18        number += 1
19        print(number, ':', b)
```

③ 결과 출력

```
입력을 종료하려면 [Enter] 키를 누르세요.
==============================
도서명 입력 : 여행의 이유
도서명 입력 : 소년이로
도서명 입력 : 희랍인 조르바
도서명 입력 : 세 여자
도서명 입력 : 아픔이 길이 되려면
도서명 입력 : 관찰의 인문학
도서명 입력 :
==============================
1 : 관찰의 인문학
2 : 세 여자
3 : 소년이로
4 : 아픔이 길이 되려면
5 : 여행의 이유
6 : 희랍인 조르바
```

03. 리스트의 활용

실습 7-9 스크린 세이버 수정하기 code07-09.py

① choice( )는 리스트에서 하나의 항목을 랜덤으로 선택하는 메소드

```
import random
colorList = ['red', 'yellow', 'green', 'orange', 'blue']
random.choice(colorList)
```

②

```
01  from turtle import *
02  from random import *
03
04  x, y, radius = 0, 0, 0
05  colorList=['red', 'yellow', 'green', 'orange', 'blue', 'violet',
06             'tan', 'brown', 'navy', 'cyan']
07
08  setup(1200, 800)                # 창 크기
09  bgcolor("black")                # 배경 색상
10  speed(0)                        # 그리기 최고 속도
11
12  for i in range(30):
13      x = randint(-500, 500)
14      y = randint(-400, 300)
15      radius = randint(80, 130)
```

03. 리스트의 활용

실습 7-9 스크린 세이버 수정하기 code07-09.py

②

```
16      penup()
17      goto(x, y)
18      pendown()
19      color(choice(colorList))    # 리스트의 색상 하나를 선택해서 채우기 색으로 설정
20      begin_fill()
21      circle(radius)
22      end_fill()
```

③

그림 7-19 스크린 세이버 수정 결과

03. 리스트의 활용

실습 7-10 버킷 리스트 만들기 code07-10.py

① 추가(1)나 삭제(2) 메뉴를 선택하면 리스트 추가나 삭제 작업 후, 목록을 출력

메뉴를 잘못 입력했을 때 다시 선택할 수 있도록 처리하고, 종료(0) 메뉴를 선택하는 경우 프로그램을 종료

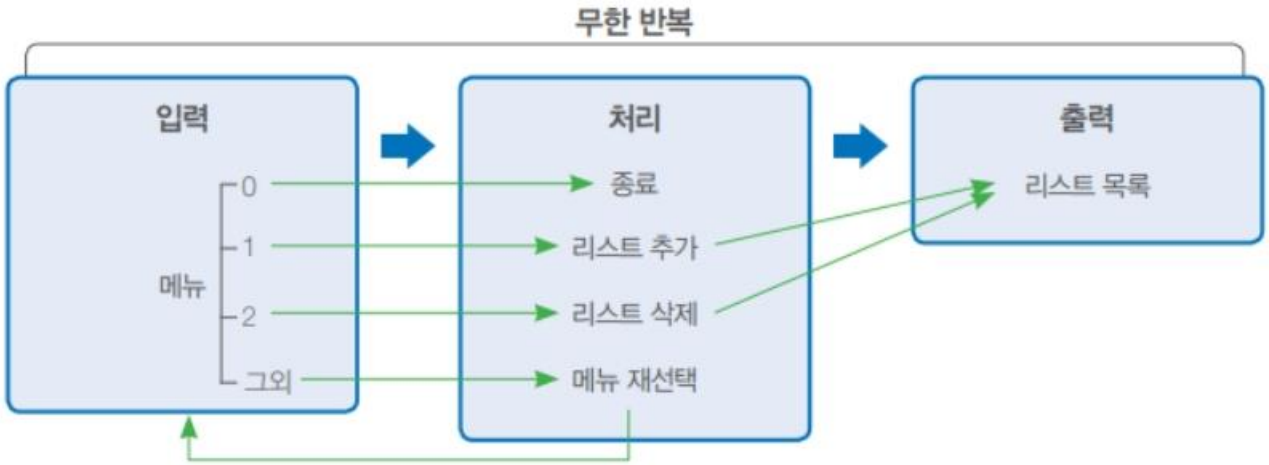


그림 7-20 버킷 리스트 프로그램의 실행 순서

03. 리스트의 활용

실습 7-10 버킷 리스트 만들기 code07-10.py

②

```
01  menu = 0
02  bucket = []
03
04  while True :
05      menu = int(input("메뉴 선택(1.추가 2.삭제 0.종료) : "))
06      if menu == 1 :
07          bucket.append(input("추가할 내용 : "))
08      elif menu == 2 :
09          bucket.remove(input("삭제할 내용 : "))
10      elif menu == 0 :
11          print("프로그램을 종료합니다.")
12          break
13      else :
14          print("메뉴 선택 오류입니다. 다시 선택하세요.")
15          continue
16      print('*' * 30)
17      for x in bucket :
18          print(x)
19      print('*' * 30)
```

03. 리스트의 활용

실습 7-10 버킷 리스트 만들기 code07-10.py

③ 프로그램을 실행하고 버킷 리스트를 추가

```
메뉴 선택(1.추가 2.삭제 0.종료) : 1
추가할 내용 : 번지 점프
******************************
번지 점프
******************************
메뉴 선택(1.추가 2.삭제 0.종료) : 1
추가할 내용 : 혼자 여행
******************************
번지 점프
혼자 여행
******************************
메뉴 선택(1.추가 2.삭제 0.종료) : 1
추가할 내용 : 독서 백권
******************************
번지 점프
혼자 여행
독서 백권
```

03. 리스트의 활용

실습 7-10 버킷 리스트 만들기 code07-10.py

④ 버킷 리스트에 저장된 항목을 삭제

```
메뉴 선택(1.추가 2.삭제 0.종료) : 2
삭제할 내용 : 혼자 여행
******************************
번지 점프
독서 백권
******************************
```

⑤ 메뉴를 잘못 입력하거나 프로그램 종료 메뉴도 선택

```
메뉴 선택(1.추가 2.삭제 0.종료) : 4
메뉴 선택 오류입니다. 다시 선택하세요.
메뉴 선택(1.추가 2.삭제 0.종료) : 0
프로그램을 종료합니다.
>>>
```