

# 파이썬 프로그래밍

18차시

리스트 활용 방법



## ⚠ 학습개요

- ... 리스트 항목 정렬
- ... 리스트 컴프리헨션
- ... 리스트에서 대입 연산자 =
- ... 깊은 복사(deep copy)
- ... 연산자 is

## ⚠ 학습목표

- ... 리스트 항목을 정렬할 수 있다.
- ... 조건을 활용해 리스트를 간결히 생성할 수 있다.
- ... 리스트를 얇게, 깊게 대입할 수 있다.
- ... 연산자 is로 동일 객체임을 확인할 수 있다.

## Chapter 1.

# 리스트 항목 정렬

P Y T H O N   P R O G R A M M I N G

## ⚠ 메소드 reverse()와 sort()

### + 메소드 reverse()

- 리스트 항목 순서를 뒤집음

```
>>> one = '잣밤배굴감'
>>> wlist = list(one)
>>> print(wlist)
['잣', '밤', '배', '굴', '감']

>>> wlist.reverse()
>>> print(wlist)
['감', '굴', '배', '밤', '잣']
```

## ⚠ 메소드 reverse()와 sort()

### + 메소드 sort()

- 리스트 항목 순서를 정렬

```
>>> one = '잣밤배굴감'
>>> wlist = list(one)
>>> wlist.sort()
>>> print(wlist)
['감', '굴', '밤', '배', '잣']
>>> wlist.sort(reverse=True)
>>> print(wlist)
['잣', '배', '밤', '굴', '감']
```



## ⚠ 내장 함수 sorted()

✦ 리스트 항목의 순서를 정렬한 리스트를 반환하는 내장 함수 sorted()

```
>>> fruit = ['사과', '귤', '복숭아', '파인애플']
>>> s_fruit = sorted(fruit)
>>> print(s_fruit)
['귤', '복숭아', '사과', '파인애플']
>>> print(fruit)
['사과', '귤', '복숭아', '파인애플']
```

```
>>> rs_fruit = sorted(fruit, reverse = True)
>>> print(rs_fruit)
['파인애플', '사과', '복숭아', '귤']
```

## ⚠ 일상 코딩: 한 글자 단어 5개와 과일 4개로 구성되는 리스트 생성과 정렬

[코딩실습] 한 글자 단어와 과일의 정렬

난이도 기본

```
1. word = list('삶꿈정')
2. word.extend('복빛')
3. print(word)
4. word.sort()
5. print(word)
6.
7. fruit = ['복숭아', '자두', '골드키위', '귤']
8. print(fruit)
9. fruit.sort(reverse=True)
10. print(fruit)
11.
12. mix = word + fruit
13. print(sorted(mix))
14. print(sorted(mix, reverse=True))
```



## ⚠ 일상 코딩- 한 글자 단어 5개와 과일 4개로 구성되는 리스트 생성과 정렬

주의	2번 줄의 '복빛'은 리스트로 이전의 리스트 '삶꿈정' 뒤에 추가된다.
결과	<pre>[ '삶', '꿈', '정', '복', '빛' ] [ '꿈', '복', '빛', '삶', '정', ] [ '복숭아', '자두', '골드키위', '귤' ] [ '자두', '복숭아', '귤', '골드키위' ] [ '골드키위', '귤', '꿈', '복', '복숭아', '빛', '삶', '자두', '정' ] [ '정', '자두', '삶', '빛', '복숭아', '복', '꿈', '귤', '골드키위' ]</pre>

## Chapter 2.

# 리스트 컴프리헨션

P Y T H O N   P R O G R A M M I N G

## ⚠ 리스트 컴프리헨션

### + 조건을 만족하는 항목으로 리스트를 간결히 생성하는 컴프리헨션

- even은 2에서 10까지의 짝수 항목으로 구성된 리스트
- 컴프리헨션은 함축, 축약, 내포, 내장 등으로도 불림

```
>>> even = []
>>> for i in range(2, 11, 2):
...     even.append(i)
...
>>> print(even)
[2, 4, 6, 8, 10]
```

#### 리스트 컴프리헨션

```
>>> even = [i for i in range(2, 11, 2)]

>>> print(even)
[2, 4, 6, 8, 10]
```

리스트 = [항목연산식 for 항목 in 시퀀스 if 조건식]

리스트 = []  
 for 항목 in 시퀀스 :  
 if 조건식 :  
 리스트.append(항목연산식)

## ⚠ 홀수의 제곱을 구하는 리스트 컴프리헨션

### + 0에서 9까지의 리스트와 10 이하의 홀수의 제곱

[코딩실습] 간단한 리스트 컴프리헨션

난이도 기본

```
1. # for문으로 리스트 생성
2. a = []
3. for i in range(10):
4.     a.append(i)
5. print(a)
6.
7. # 컴프리헨션으로 리스트 생성
8. seq = [i for i in range(10)]
9. print(seq)
10.
11. # for문으로 리스트 생성
12. s = []
13. for i in range(10):
14.     if i%2 == 1:
15.         s.append(i**2)
16. print(s)
```

## ⚠ 홀수의 제곱을 구하는 리스트 컴프리헨션

### + 0에서 9까지의 리스트와 10 이하의 홀수의 제곱

[코딩실습] 한 글자 단어와 과일의 정렬

난이도 기본

```
17.  
18. #컴프리헨션으로 리스트 생성  
19. squares = [i**2 for i in range(10) if i%2 == 1]  
20. print(squares)
```

주의 컴프리헨션에는 콜론이 제거된다는 점에 주의하자.

결과

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[1, 9, 25, 49, 81]  
[1, 9, 25, 49, 81]
```

## Chapter 3.

# 리스트에서 대입 연산자 =

P Y T H O N   P R O G R A M M I N G



## ⚠ 리스트 대입에 의한 동일 리스트의 공유

### + 리스트에서 대입 연산자 =

- 얇은 복사(shallow copy)
- 대입되는 변수가 동일한 시퀀스를 가리킴
- 결국 변수 f2와 f1은 하나의 같은 리스트

```
>>> f1 = ['사과', '귤', '복숭아', '파인애플']
>>> f2 = f1
>>> f2.pop()
'파인애플'
>>> print(f1)
['사과', '귤', '복숭아']
>>> print(f2)
['사과', '귤', '복숭아']
```

## ⚠ 리스트 대입에 의한 동일 리스트의 공유

### + 리스트에서 대입 연산자 =

- 얕은 복사(shallow copy)
- 대입되는 변수가 동일한 시퀀스를 가리킴
- 결국 변수 f2와 f1은 하나의 같은 리스트

#### Python 3.6

```
1. f1 = ['사과', '귤', '복숭아', '파인애플']
2. f2 = f1
3. f2.pop()
```

[Edit this code](#)

#### Frames

Global frame  
f1  
f2

#### Objects list

0	1	2	3
"사과"	"귤"	"복숭아"	"파인애플"

마지막 항목인 '파인애플' 삭제

- line that has just executed
- next line to execute

## ⚠ 리스트 대입에 의한 동일 리스트의 공유

### + 파이썬 실행 시 메모리 모습을 볼 수 있음

- 파이썬 튜터 홈페이지
  - [www.pythontutor.com](http://www.pythontutor.com)

## Chapter 4.

# 깊은 복사 (deep copy)

P Y T H O N   P R O G R A M M I N G

## ⚠ 리스트의 깊은 복사에 의한 대입으로 새로운 리스트의 생성

### + 깊은 복사(deep copy)

- 새로운 리스트를 만들어 복사
- 슬라이스 [:]
- copy()
- 또는 list() 함수

```
>>> f1 = ['사과', '귤', '복숭아', '파인애플']
>>> f2 = f1[:]
>>> f2.pop(1)
'귤'
>>> print(f1, f2)
['사과', '귤', '복숭아', '파인애플'] ['사과', '복숭아', '파인애플']
```

## ⚠ 리스트의 깊은 복사에 의한 대입으로 새로운 리스트의 생성

### + 깊은 복사(deep copy)

- 새로운 리스트를 만들어 복사
- 슬라이스 [:]
- copy()
- 또는 list() 함수

```
>>> f3 = f1.copy()
>>> f3.pop()
'파인애플'
>>> print(f1, f3)
['사과', '귤', '복숭아', '파인애플'] ['사과', '귤', '복숭아']
```

```
>>> f4 = list(f1)
>>> f4.append('감')
>>> print(f1, f4)
['사과', '귤', '복숭아', '파인애플'] ['사과', '귤', '복숭아', '파인애플', '감']
```



## ⚠ 리스트의 깊은 복사에 의한 대입으로 새로운 리스트의 생성

### + 깊은 복사(deep copy)

- 다음 그림과 같이 f1과 f2는 항목이 같지만 전혀 다른 리스트다.

#### Python 3.6

```
1. f1 = ['사과', '귤', '복숭아', '파인애플']
2. f2 = f1[:]
```

[Edit this code](#)

- line that has just executed
- next line to execute

Click a line of code to set a breakpoint;  
use the Back and Forward buttons to jump there.

#### Frames

Global frame  
f1  
f2

#### Objects

0	1	2	3
“사과”	“귤”	“복숭아”	“파인애플”
0	1	2	3
“사과”	“귤”	“복숭아”	“파인애플”

[그림18-2] 리스트의 깊은 복사 대입으로 인한 다른 리스트의 생성

## Chapter 5.

# 연산자 is

P Y T H O N   P R O G R A M M I N G

## ⚠ 변수의 동일 객체 여부를 검사하는 is

### + 문장 is

- 피연산자인 변수 2개가 동일한 메모리를 공유하는지 검사
  - 같으면 True, 다르면 False를 반환

```
>>> f1 = ['사과', '귤', '복숭아', '파인애플']
>>> f2 = f1
>>> print(f1 is f2)
True
>>> f3 = f1[:]
>>> print(f1 is f3)
False
```

## ⚠ 리스트 항목 정렬

- ... 메소드 `reverse()`와 `sort()`
- ... 내장 함수 `sorted()`

## ⚠ 리스트 컴프리헨션

- ... 조건을 만족하는 항목으로 리스트를 간결히 생성하는 컴프리헨션
- ... 컴프리헨션은 함축, 축약, 내포, 내장 등으로도 불림

## ⚠ 리스트에서 대입 연산자 =

... 얇은 복사(shallow copy)

## ⚠ 깊은 복사(deep copy)

- ... 새로운 리스트를 만들어 복사
- ... 슬라이스 [:]
- ... `copy()`
- ... 또는 `list()` 함수

# SUMMARY



동양미래대학교  
DONGYANG MIRAE UNIVERSITY

## ⚠ 연산자 is

... A is B