

5주차
파이썬 기초 프로그래밍



리스트

- 리스트란?
- 리스트 메소드
- 스택과 큐
- 중첩 리스트
- 리스트 복사

리스트란?

- [] 안에 둘러싸인 객체들의 모임
- 하나의 이름에 여러 데이터 저장
- 시퀀스 자료형 이면서 변경 가능

시퀀스 자료형

1. 여러 개의 객체 저장
2. 각 객체는 순서를 가짐
3. 인덱싱, 슬라이싱, 연결, 반복, 멤버 검사, 길이정보 가능
4. 문자열, 리스트, 튜플, 바이트 시퀀스

시퀀스 자료형(1)

- `L = [1,2,3,4,5,6]`

| 구분 | 연산 | 설명 | 예 | 결과 |
|-------|----------------------|----------------------------|----------------------------|--------------------------|
| 인덱싱 | <code>[k]</code> | k번 위치의 값 | <code>L[0]</code> | 1 |
| 슬라이싱 | <code>[s:t:p]</code> | s부터 t 사이 구간의 값을 p 간격으로 취한다 | <code>L[1:5:2]</code> | <code>[2,4]</code> |
| 연결하기 | <code>+</code> | 두 시퀀스 연결 | <code>[1,2]+[3,4,5]</code> | <code>[1,2,3,4,5]</code> |
| 반복하기 | <code>*</code> | 반복 | <code>[1]*3</code> | <code>[1,1,1]</code> |
| 멤버 검사 | <code>in</code> | 어떤 값이 시퀀스 자료형에 속하는지 검사 | <code>3 in L</code> | True |
| 길이 정보 | <code>len()</code> | 크기 | <code>len(L)</code> | 6 |

시퀀스 자료형(2)

```
>>> a = []                # 빈 리스트
>>> a = [1,2,3]
>>> print (a[0], a[-1])   # 인덱싱
1, 3
>>> print (a[1:3], a[:])  # 슬라이싱
[2, 3] [1, 2, 3]
>>> L = list(range(10))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> L[::2]                # 확장 슬라이싱
[0, 2, 4, 6, 8]
```

```
>>> a * 2                # 반복
[1, 2, 3, 1, 2, 3]
>>> a + [4, 5, 6]        # 연결
[1, 2, 3, 4, 5, 6]
>>> 4 in L               # 멤버 검사
True
>>> len(a)               # 길이 정보
3
```

리스트 생성

- 직접 정의

- $L = [1, 2, 3, 4]$

- 리스트 내장 사용

- $L = [k * k \text{ for } k \text{ in range}(10)]$

출력식 변수의 범위

- $L = [k * k \text{ for } k \text{ in range}(10) \text{ if } k \% 2 == 1]$

조건부 서식

참일 때만 출력 수식 적용

- 리스트 객체 생성

- $L = \text{list}(\text{range}(10))$

리스트 수정(1)

- 리스트 데이터 변경하기
 - 리스트[index] = 새로운 값
 - 예 : L[0] = 9
friendList[index] = newName
- 리스트 데이터 삭제하기
 - del L[2]
 - L.pop(2)

```
>>> L = [1,2,3,4,5]
>>> L[0] = 9 → 첫번째 값 수정
>>> print(L)
[9, 2, 3, 4, 5]
>>> del L[3] → 네번째 값 삭제
>>> L
[9, 2, 3, 5]
```

리스트 수정(2)

- `a = ['spam', 'egg', 123, 1234]` 일 때,

- 리스트 치환

 - `a[0:2] = [1, 12]`

(`a[0:2] = [1]`)

- 리스트 일부 삭제

 - `a[0:2] = []`

크기가 달라도 됨

- 리스트 추가

 - `a[1:1] = ['spam', 'ham']`

- 리스트 값 삭제

 - `del a[0]`

 - `del a[1:]`

리스트에 있는 자료 사용하기

- for 문으로 리스트의 값 하나씩 꺼내기

```
L = [i for i in range(100) if i % 3 == 0]
for i in L :
    print(i, end=' ')
```

- for 문으로 인덱스 생성 : 리스트의 값 변경할 때

```
myList = list(range(4))
for i in range(len(myList)) :
    myList[i] *= 3
```

리스트 내포(comprehension)

```
>>> number1 = [1, 3, 5, 7]
>>> number2 = [2, 4, 6, 8]
>>> print(i + j for (i, j) in zip(number1, number2))
```

output :

```
[3, 7, 11, 15]
```

실습 01

- 예약어(keyword)는 변수 이름으로 사용할 수 없다.
사용자로 부터 변수 이름을 입력 받아 keyword인지를
판별하는 프로그램을 작성하시오.
 - 힌트 : keyword list 알아보기 라이브러리

```
>>> import keyword  
>>> keyword.kwlist
```

리스트 조작 함수(메소드)

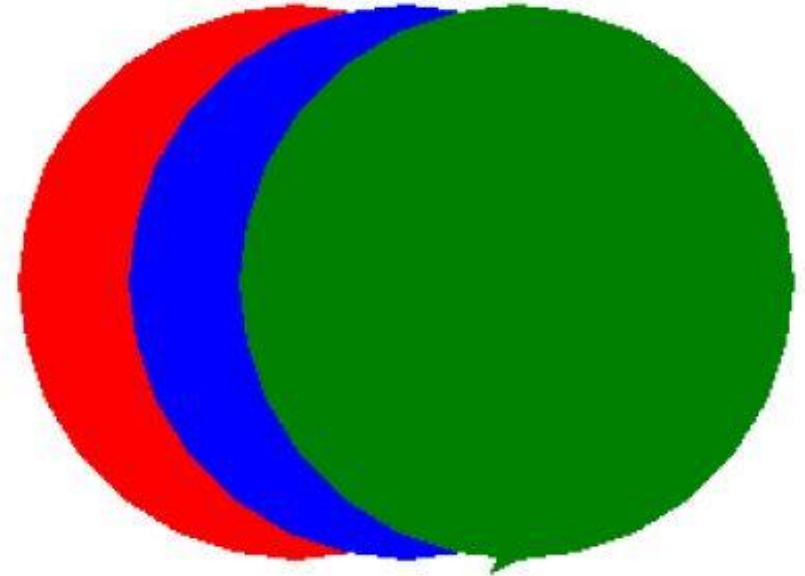
• L = [1,2,3,4,5,6] 일 때

| 메서드 | 설명 | 사용법 | 결과 |
|---------|--|---------------------|-----------------------|
| append | 끝에 데이터 추가 | L.append(7) | [1,2,3,4,5,6,7] |
| count | 지정한 값의 개수 반환 | L.count(3) | 1 |
| extend | 리스트 추가. + 연산과 동일 | L.extend([8,9]) | [1,2,3,4,5,6,7,8,9] |
| index | 지정한 값의 첫번째 인덱스 반환 | L.index(4) | 3 |
| insert | 지정한 위치에 값을 삽입 | L.insert(3,0) | [1,2,3,0,4,5,6,7,8,9] |
| pop | 지정한 위치 의 항목을 반환하고 그 값은 리스트에서 삭제 . 위치를 지정 안하면 마지막 항목 반환 | L.pop() L.pop(3) | 9 0 |
| remove | 지정한 값을 삭제 . 값이 여러 개면 첫번째 값만 삭제 | L.remove(2) | [1,3,4,5,6,7,8] |
| reverse | 리스트의 순서를 역순으로 바꾼다 | L.reverse() | [8,7,6,5,4,3,1] |
| sort | 리스트 정렬 | L.sort() | [1,3,4,5,6,7,8] |

실습 02

- 사용자로 부터 3가지 색깔(R,G,B)을 입력 받아, 리스트에 저장한 후, 반지름 100인 원을 그리고 리스트에 있는 색상으로 원 안을 채우시오. 이때 3개의 원은 조금씩 겹치도록 한다.
 - 원 채우기

```
color('yellow')  
begin_fill()  
circle(50)  
end_fill()
```



- turtle graphic API 참고 : <https://url.kr/Wik2uj>

리스트를 스택으로 사용하기

- 스택
 - 나중에 넣은 데이터를 먼저 꺼내 사용하는 자료구조
 - Last In, First Out : LIFO
 - 예 : 접시 쌓기
 - 연산
 - push : 데이터 넣기 (**append()** 메소드 사용)
 - pop : 데이터 꺼내기 (**pop()** 메소드 사용)

```
>>> s = [10, 20, 30, 40, 50]
>>> s.append(60)          # push 연산
[10, 20, 30, 40, 50, 60]
>>> s.pop()              # pop 연산
60
>>> s
[10, 20, 30, 40, 50]
```

리스트를 큐으로 사용하기

- 큐(Queue)?
 - 먼저 들어온 데이터를 먼저 꺼내 사용하는 자료구조
 - First In, last Out : FIFO
 - 예 : 버스 정류장의 줄
 - 연산
 - put(혹은 insert) : 데이터 넣기 (**append()** 메소드 사용)
 - get(혹은 delete) : 데이터 꺼내기 (**pop(0)** 메소드 사용)

```
>>> s = [10, 20, 30, 40, 50]
>>> s.append(60)           # insert 연산
[10, 20, 30, 40, 50, 60]
>>> s.pop(0)              # delete 연산
10
>>> s
[20, 30, 40, 50, 60]
```

중첩 리스트(nested list)

- 리스트 안에 또 다른 리스트가 포함되어 있는 경우

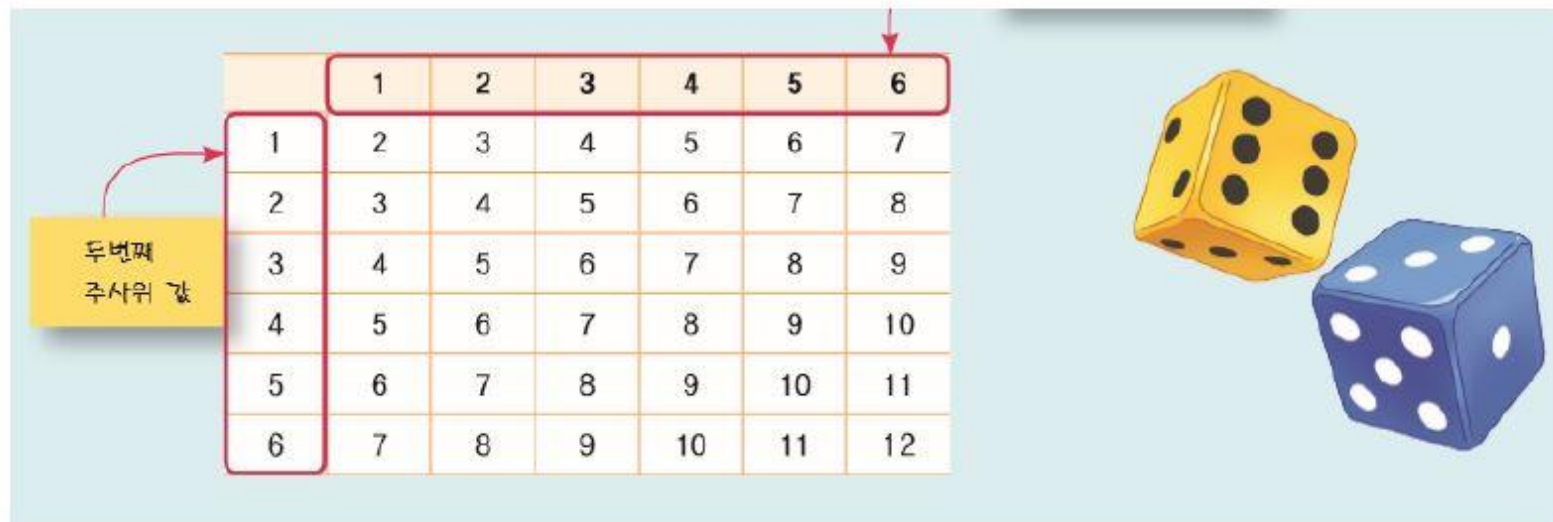
```
>>> s = [1, 2, 3]
>>> t = ['begin', s, 'end']    # 중첩 리스트
>>> t
['begin', [1, 2, 3], 'end']
>>> t[1][1]
2
```

- 리스트는 다른 객체를 직접 저장하지 않고 객체들의 참조(주소)만을 저장

중첩 리스트(nested list)

- 2차원 리스트

- 2개의 주사위를 굴리면 다음 표와 같은 36가지의 결과가 나온다.
이것을 6X6 크기의 2차원 리스트로 생성하여 보자



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |

결과

```
[[2, 3, 4, 5, 6, 7], [3, 4, 5, 6, 7, 8], [4, 5, 6, 7, 8, 9],  
[5, 6, 7, 8, 9, 10], [6, 7, 8, 9, 10, 11], [7, 8, 9, 10, 11, 12]]
```

```
rows = 6  
cols = 6  
table = [ ]
```

```
# 2차원 리스트를 생성한다.  
for row in range(rows):  
    table += [[0]*cols]
```

```
# 2차원 리스트의 각 요소에 rows와 cols 값을 더하여 저장한다.  
for row in range(rows):  
    for col in range(cols):  
        table[row][col] = (row+1+col+1)
```

```
print(table)
```

중첩 리스트

- 오류기 그리기

```
from turtle import *

reset()
positions = [[-120, 60, 'blue'], [0, 60, 'black'], [120,60,'red'],
             [-60, 0, 'yellow'], [60, 0, 'green']]

for x, y, c in positions :
    up()
    goto(x, y)
    down()

    color(c)

    begin_fill()
    circle(50)
    end_fill()
```

리스트 복사

- 얇은 복사
 - 주소만 복사
 - 같은 객체 공유
- 깊은 복사
 - 모든 객체 복제
 - 다른 객체를 가르킴

```
>>> L = [1, 2, 3, 4, 5]
>>> a = L
>>> a
[1, 2, 3, 4, 5]
>>> a[2] = 99
>>> a
[1, 2, 99, 4, 5]
>>> L
[1, 2, 99, 4, 5]
```

```
>>> b = list(L)
>>> b
[1, 2, 99, 4, 5]
>>> b[2] = 3
>>> b
[1, 2, 3, 4, 5]
>>> L
[1, 2, 99, 4, 5]
>>> a
[1, 2, 99, 4, 5]
>>>
```