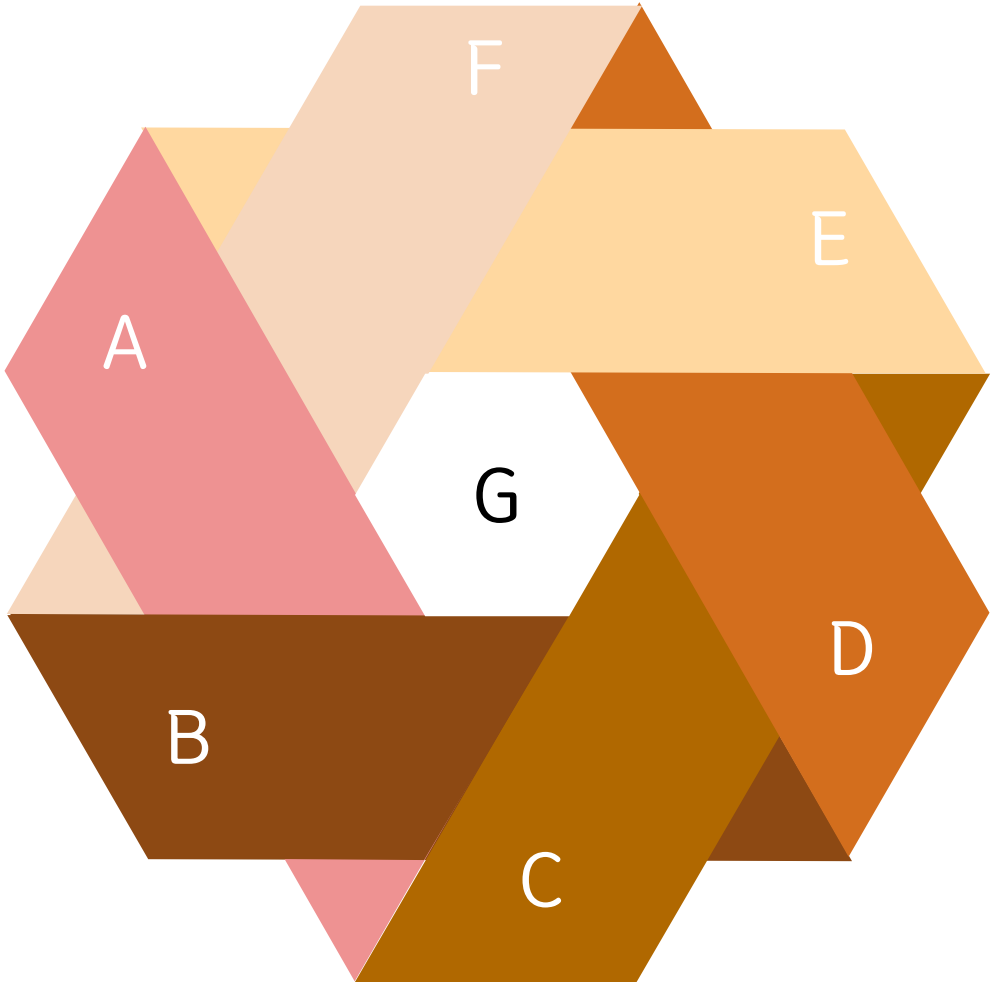
Python

리스트 자료형

A 리스트

B 숫자

C 문자열



튜플 D

딕셔너리 E

집합 F

불 G

리스트는 어떻게 만들고 사용할까?

▶기본문법

리스트명 = [요소 1, 요소 2, 요소 3, · · ·]

▶ 예시

```
>>>a = [ ]
>>>b = [ 1 , 2 , 3 ]
>>>c = [ 'Life' , 'is' , 'too' , 'short' ]
>>>d = [ 1 , 2 , 'Life' , 'is' ]
>>>e = [ 1 , 2 , [ 'Life' , 'is'] ]
```

빈 리스트
숫자를 요소로 갖는 리스트
문자열을 요소로 갖는 리스트
숫자와 문자열을 요소로 갖는 리스트
숫자와 리스트를 요소로 갖는 리스트

리스트 안에는 어떠한 자료형도 포함시킬 수 있다.
빈 리스트는 a =list() 로 생성할 수도 있다.

리스트의 인덱싱

리스트 a = [1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9]

인덱스	0	1	2	3	4	5	6	7	8
리스트 a	1	2	3	4	5	6	7	8	9

리스트 요소 하나씩 접근

리스트 인덱스도 문자열 인덱스처럼 0 부터 시작

리스트의 인덱싱

```
>>> a = [ 1 , 2 , 3]
>>> a
```

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

```
>>> a [ 0 ] + a [ 2 ]
```

```
>>> a [ -1 ]
```

리스트의 인덱싱

```
>>> a = [ 1 , 2 , 3]
>>> a
[ 1 , 2 , 3 ]
```

```
>>> a [ 0 ]
1
```

```
>>> a [ 0 ] + a [ 2 ]
4
```

```
>>> a [ -1 ]
3
```

리스트의 인덱싱

```
>>> a = [ 1 , 2 , 3 , [ ‘a’ , ‘b’ , ‘c’ ] ]
>>> a[0]
```

```
>>> a [ -1 ]
```

```
>>> a [ 3 ]
```

```
>>> a [ -1 ][0]
```

리스트의 인덱싱

```
>>> a = [ 1 , 2 , 3 , [ ‘a’ , ‘b’ , ‘c’ ] ]
>>> a[0]
1
```

```
>>> a [ -1 ]
[ ‘a’ , ‘b’ , ‘c’ ]
```

```
>>> a [ 3 ]
[ ‘a’ , ‘b’ , ‘c’ ]
```

```
>>> a [ -1 ][0]
‘a’
```

리스트의 인덱싱

```
>>> a = [ 1 , 2  , [ ‘a’ , ‘b’ , [ ‘Life’, ‘is’ ] ] ]
>>> a [ 2 ][ 2 ][ 0 ]
```

리스트의 인덱싱

```
>>> a = [ 1 , 2  , [ ‘a’ , ‘b’ , [ ‘Life’, ‘is’ ] ] ]
>>> a [ 2 ][ 2 ][ 0 ]
```

‘Life’

삼중 구조의 리스트에서 ‘Life’ 문자열 접근-> a[2][2][0]
리스트 a의 세번째 요소인 [‘a’ , ‘b’ , [‘Life’, ‘is’]]
[‘a’ , ‘b’ , [‘Life’, ‘is’]]의 세번째 요소인 [‘Life’, ‘is’]
[‘Life’, ‘is’]의 첫번째 요소인 ‘Life ' 를 나타낸다.

리스트의 슬라이싱

리스트 a = [1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9]

인덱스	0	1	2	3	4	5	6	7	8
리스트 a	1	2	3	4	5	6	7	8	9

a[1:7]

연속적인 객체(문자열, 리스트) 범위를 지정해 선택하는 표기법

리스트의 슬라이싱

```
>>> a = [1,2,3,4,5]
>>> a[0:2]
```

```
>>> a = [1,2,3,4,5]
>>> b = [a:2]
>>> c = [2:]
>>> b
```

```
>>> c
```

리스트의 슬라이싱

```
>>> a = [1,2,3,4,5]
>>> a[0:2]
```

```
[1,2]
```

```
>>> a = [1,2,3,4,5]
>>> b = [a:2]
>>> c = [2:]
>>> b
```

```
[1,2]
```

```
>>> c
```

```
[3,4,5]
```

리스트의 슬라이싱

```
>>> a = [1, 2, 3, ['a', 'b', 'c'], 4, 5]
>>> a[2:5]
```

```
>>> a[3][:2]
```

리스트의 슬라이싱

```
>>> a = [ 1 , 2 , 3 , [ ‘a’ , ‘b’ , ‘c’] , 4 , 5 ]
>>> a [ 2 : 5 ]
[ 3 , [ ‘a’ , ‘b’ , ‘c’] , 4 ]
```

```
>>> a [ 3 ][ : 2 ]
[ ‘a’ , ‘b’ ]
```

a [3] 은 [‘a’ , ‘b’ , ‘c’]를 나타낸다.
a [3][: 2]은 [‘a’ , ‘b’ , ‘c’]에서
첫번째 요소부터 3번째요소 직전까지의 값을 의미한다.

리스트 연산하기

더하기

```
>>> a = [ 1 , 2 , 3 ]
>>> b = [ 4 , 5 , 6 ]
>>> a + b
```

[1 , 2 , 3 , 4 , 5 , 6]

반복하기

```
>>> a = [ 1 , 2 , 3]
>>> a * 3
```

[1 , 2 , 3 , 1 , 2 , 3 , 1 , 2 , 3]

길이 구하기

```
>>> a = [ 1 , 2 , 3]
>>> len ( a )
```

3

하기 쉬운 리스트 연산 오류

```
>>> a = [ 1 , 2 , 3 ]
>>> a [ 2 ] + “hi”
```

리스트 연산하기

더하기

```
>>> a = [ 1 , 2 , 3 ]
>>> b = [ 4 , 5 , 6 ]
>>> a + b
```

[1 , 2 , 3 , 4 , 5 , 6]

반복하기

```
>>> a = [ 1 , 2 , 3]
>>> a * 3
```

[1 , 2 , 3 , 1 , 2 , 3 , 1 , 2 , 3]

길이 구하기

```
>>> a = [ 1 , 2 , 3]
>>> len ( a )
```

3

하기 쉬운 리스트 연산 오류

```
>>> a = [ 1 , 2 , 3 ]
>>> str ( a [ 2] ) + “hi”
```

리스트의 수정과 삭제

값 수정하기

```
>>> a = [ 1 , 2 , 3 ]
>>> a [ 2 ] = 4
>>> a
```

```
[ 1 , 2 , 4 ]
```

del함수 사용해 요소 삭제하기

```
>>> a = [ 1 , 2 , 3 ]
>>> del a [ 1 ]
>>> a
```

```
[ 1 , 3 ]
```

리스트의 수정과 삭제

값 수정하기

```
>>> a = [ 1 , 2 , 3 ]
>>> a [ 2 ] = 4
>>> a
```

```
[ 1 , 2 , 4 ]
```

del함수 사용해 요소 삭제하기

```
>>> a = [ 1 , 2 , 3 ]
>>> del a [ 1 ]
>>> a
```

```
[ 1 , 3 ]
```

del함수는 파이썬이
자체적으로 가지고 있는
삭제함수!
기본 구조 -> del 객체
* 객체 = 모든 자료형

리스트의 수정과 삭제

값 수정하기

```
>>> a = [ 1 , 2 , 3 ]
>>> a [ 2 ] = 4
>>> a
```

```
[ 1 , 2 , 4 ]
```

del함수 사용해 요소 삭제하기

```
>>> a = [ 1 , 2 , 3 , 4 , 5]
>>> del a [ 2 : ]
>>> a
```

```
[ 1 , 2 ]
```

리스트 관련 함수들

추가
append

```
>>> a = [ 1 , 2 , 3]
>>> a.append(4)
>>>a
[ 1 , 2 , 3 , 4 ]

>>> a.append([5,6])
>>>a
[ 1 , 2 , 3 , 4 , [ 5 , 6 ] ]
```

정렬
sort

```
>>> a = [ 1 , 4 , 3 , 2 ]
>>> a.sort()
>>> a
[ 1 , 2 , 3 , 4 ]

>>> a = [ 'a' , 'c' , 'b' ]
>>> a. sort()
>>>a
['a' , 'b' ,c']
```

뒤집기
reverse

```
>>> a = [ 'a' , 'c' , 'b' ]
>>> a. reverse()
>>>a
['b' , 'c' , 'a']
```

위치반환
index

```
>>> a = [ 1 , 2 , 3 ]
>>> a.index(3)
2

>>> a. index(0)
ValudeError:0 is not in list
```

삽입
insert

```
>>> a = [ 1 , 2 , 3 ]
>>> a. insert(0,4)
>>>a
 [4 , 1 , 2 , 3 ]

>>> a. insert(3,5)
 [4 , 1 , 2 , 5 , 3 ]
```

제거
remove

```
>>> a = [ 1 , 2 , 3 , 1 , 2 , 3 ]
>>> a.remove(3)
>>>a
[ 1 , 2 , 1 , 2 , 3]
```

꺼내기
pop

```
>>> a = [ 1 , 2 , 3 ]
>>> a. pop()
3
>>> a
[1 , 2 ]

>>> a. pop(1)
2
>>> a
[1 , 3 ]
```

개수세기
count

```
>>> a = [ 1 , 2 , 3 , 1 ]
>>> a.count(1)
2
```

확장
expend

```
>>> a = [ 1 , 2 , 3 ]
>>> a. extend([4,5])
>>> a
[1 , 2 , 4 , 5 ]

>>> b = [ 6 , 7 ]
>>> a.extend(b)
>>> a
[1 , 2 , 4 , 5 , 6 , 7]
```

서식지정
formatting

```
>>> print("integer :%d" % (100))
Integer : 100

>>> print("Hi I'm %s." % "abc")
Hi I'm abc.

>>> print("integer ={}".format(100))
Integer : 100
```

성능 : % > format

리스트 내포 List comprehension

List의 [ ]괄호 안에 for루프를 사용하여 반복적으로 표현식(expression)을 실행해 리스트 요소들을 정의하는 특별한 용법

▶기본문법
[표현식 for 항목 in 반복가능 객체 if 조건문]
▶예시
>>> list = [n**2 for n in range(10) if n % 3 == 0]
>>>print(list)
[0,9,36,81]

복잡하지만 for문을 여러 개 사용할 수 있다.
[표현식 for항목1 in 반복가능객체1 if 조건문1
for 항목2 in 반복가능객체2 if 조건문2
. . .
for 항목n in 반복가능객체n if 조건문n]

추가	리스트.append(요소)
정렬	리스트.sort()
역순	리스트.reverse()
위치반환	리스트.index(요소)
삽입	리스트.insert(인덱스,요소)
제거	리스트.remove(요소)
꺼내기	리스트.pop()
개수세기	리스트.count(요소)
확장	리스트.extend(리스트)
내포	리스트 = [표현식 항목 in 반복가능 객체 if 조건문]

1. 리스트에 추가할 때 쓰는 함수는?

2. 리스트를 정렬할 때 쓰는 함수는?

3. 리스트의 특정 위치에 특정 값을 삽입할 때 쓰는 함수는?

4. 리스트의 특정 값의 개수를 셀 때 쓰는 함수는?

5. 리스트의 특정 값의 위치를 반환할 때 쓰는 함수는?

연습문제

range(5), range(1,5), range(0,5,2)를 리스트로 변환 후 출력한다.

빈 리스트에 1부터 10까지 추가한 후 홀수를 제거한다. 인덱스 0에 20을 삽입한후 정렬해서 출력한다.

리스트 a = [2,4,8,16,32]일 때, 임의의 값이 리스트 요소인지를 판별해서 출력한다.

리스트[3,7,5,2,9,1,4]를 역순으로 정렬해서 출력한다.

리스트 [1,2,3,4,5,6,7,8,9]에서 4번째 요소를 출력한 후 4번째 요소에 10을 넣는다. (리스트개수는 9로 고정)

리스트a[1,2,3]과 리스트b [4,5,6]을 만들어 리스트 a에 리스트 b를 붙인 후 5의 위치를 출력한다.

끝