

## ❖ 자료구조(data structure)

정보(데이터)를 메모리에 효율적으로 저장하고 사용하는 방법을 정의한 것으로, 데이터를 관리하는 방식이다. 대용량일수록 메모리에 빨리 저장하고 빠르게 검색하여, 데이터를 효율적으로 사용하고 실행 시간을 줄일 수 있게 해 준다.

20 YELLOW PAGE 더라이프 전화번호부

|   |  |   |
|---|--|---|
| <b>재용국 한국영사관</b><br>북경대사관영사관 010-6532-6774<br>성도출영사관 029-8816-5800<br>홍콩출영사관 00852-2529-4141<br>상해출영사관 021-6295-5000<br>천도출영사관 0532-8897-6001<br>심양출영사관 024-2385-3388<br>광둥출영사관 020-3887-0555<br>시안출영사관 029-8835-1001 | 책임직원 방계 조직원 보호계<br>의무직업 전문직계 개업소<br><b>아이나루 어린이집</b><br>교육심사: 185-8079-2116<br>아이나루 어린이집 185-8079-2116<br>중경사무원국대학교 131-9315-7911<br>중경제국대학교 158-2384-9863<br>중경한글학교 150-0235-4052<br>KASVLL국제/장년대학 173-6522-6094<br>한나어린이집 136-3798-5335  | <b>음식점</b><br>강남아저씨 186-9650-0675<br>고함집(高峯) 137-7318-1808<br>내고향 공방집 023-6721-0099<br>내고향 황근방집 023-6710-7519<br>내고향 치보집 136-1822-4115<br>내고향 흥후집 023-6739-4989<br>떡볶이 181-7564-8006<br>푸공회(阜公) 139-8338-2107<br>미아정육식당 136-2834-0525<br>전통음식 백화 023-6305-9288<br>복자산 126-4055-9991<br>서울집 186-9885-3758<br>스카이 피자국밥 185-8485-1153<br>애당양꼬치 186-1111-3050<br>일가 소마리아집 135-0127-6665<br>청춘 156-9210-5002<br>취향 중화요리 131-4021-0035<br>한복가(韓服) 157-3046-9099<br>한국강남스타일 159-9891-3032<br>한성집 159-2277-9826<br>한주 채주집 023-6703-4432 |
| <b>기 관</b><br>KOTRA 중경무역관 6039-1005<br>중소기업진흥공단(중경) 023-6705-2399<br>중경일시정부청사 6382-0753<br>중경한인(韓)회 185-8006-9033   | <b>호텔 / 민방</b><br>동태호텔스타 136-0016-1377<br>리호호텔 157-3046-9099<br>해디슨호텔(海帝申) 186-9675-3116<br>해디슨호텔(海帝申) 181-8311-3395<br>크리스호텔(晶水) 185-8003-8464<br>푸공호텔(阜公) 182-0301-6668<br>영원호텔 150-8672-0586<br>장수사우나 157-3046-9099<br>중경강남타워 131-1014-2603<br>중경서울타워 183-8158-4114<br>중경비행(中京)제자전소 177-8356-3352<br>화일민박 183-7566-1445<br>한국부호집 131-1014-2603<br>호텔한학빌(北漢) 133-1117-8000<br>힐튼호텔 157-3014-6066 | <b>마음 / 식품 / 판매업</b><br>능달집지 186-0234-0145<br>포갈마트 186-8888-2723<br>롯데마트(大井) 153-1028-8885<br>메리안자네일샵 182-2508-7772<br>모아TV 185-2382-7979   |
| <b>진급 생활 전화</b><br>공안/사건사고 110<br>전파고장신고 112<br>국내장가(國內)인내 113<br>국제전화안내 115<br>화재전화번호 116<br>시간안내 117<br>화재신고 119<br>구급센터 120<br>날씨안내 121<br>교통사고 122<br>우편번호안내 184  | <b>기업 / 건설업</b><br>거성가게 186-1048-3282<br>다중대업 186-1162-3396<br>비밀가게가게 6734-3488<br>아시아나항공 6796-2600 / 2033<br>아틀라스 컴퍼니 177-8319-6880   | <b>부동산</b><br>상해중경114부동산 133-2195-2346<br>중경서울부동산 187-2336-0523<br>중경114부동산 186-0218-9115<br>한나부동산 136-2165-5550<br>한신부동산 183-5801-1818   |
| <b>미용 / 패션</b><br>미용(美容) 185-8019-4590<br>미용(美容) 185-8019-4590  | <b>미용 / 패션</b><br>미용(美容) 185-8019-4590<br>미용(美容) 185-8019-4590   | <b>미용 / 패션</b><br>미용(美容) 185-8019-4590<br>미용(美容) 185-8019-4590  |

(a) 전화번호부

9:58

그룹 연락처

검색

나성엽

도민준

라인업

마성지

(b) 휴대전화의 연락처

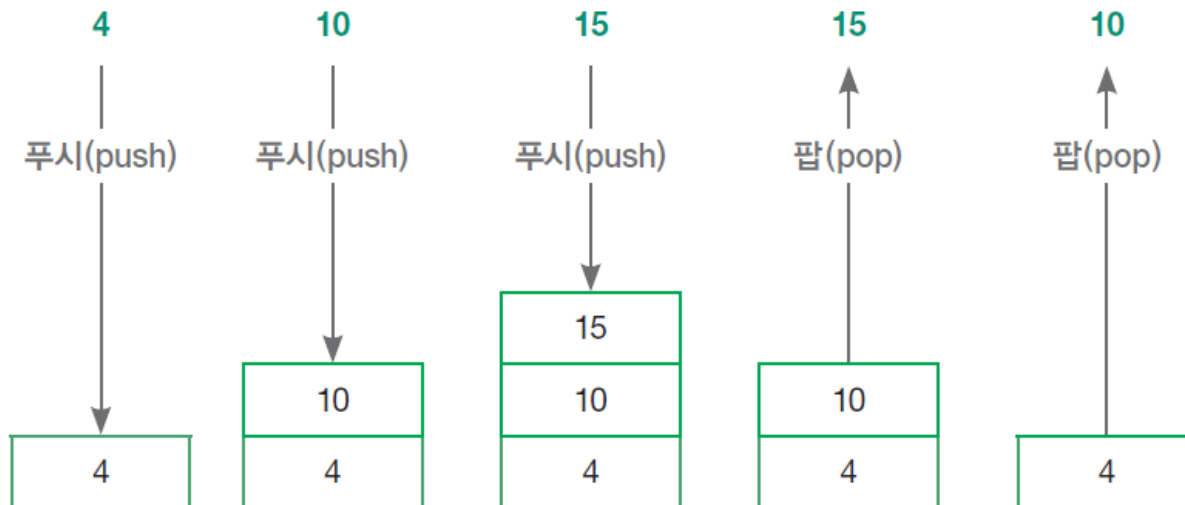
## ❖ 파이썬에서 지원하는 자료구조의 종류

| 자료구조명                | 특징   |
|----------------------|--|
| 스택(stack)            | 나중에 들어온 값을 먼저 나갈 수 있도록 해 주는 자료구조(last in first out)                        |
| 큐(queue)             | 먼저 들어온 값을 먼저 나갈 수 있도록 해 주는 자료구조(first in first out)                        |
| 튜플(tuple)            | 리스트와 같지만, 데이터의 변경을 허용하지 않는 자료구조  |
| 세트(set)              | 데이터의 중복을 허용하지 않고, 수학의 집합 연산을 지원하는 자료구조                                     |
| 딕셔너리<br>(dictionary) | 전화번호부와 같이 키(key)와 값(value) 형태의 데이터를 저장하는 자료구조, 여기서 키값은 다른 데이터와 중복을 허용하지 않음 |
| collections 모듈       | 위에 열거된 여러 자료구조를 효율적으로 사용할 수 있도록 지원하는 파이썬 내장(built-in) 모듈                   |

} 리스트

## ❖ 스택

- **스택(stack)**: 자료구조의 핵심 개념 중 하나로, 'Last In First Out(LIFO)'으로 정의할 수 있다.
- 마지막에 들어간 데이터가 가장 먼저 나오는 형태로, 데이터의 저장 공간을 구현하는 것이다.
- 데이터를 저장하는 공간으로, 리스트와 비슷하지만 저장과 꺼내는 순서가 바뀌는 형태를 스택 자료구조(stack data structure)라고 한다.
- 스택에서 데이터를 저장하는 것을 푸시(push), 데이터를 추출하는 것을 팝(pop)이라고 한다.



## ❖ 스택

- 파이썬에서는 리스트를 사용하여 스택을 구현할 수 있다.
- 리스트라는 저장 공간을 만든 후, `append()` 함수로 데이터를 저장(push)하고 추출(pop)한다.

```
>>> a = [1, 2, 3, 4, 5]
>>> a.append(10)
>>> a
[1, 2, 3, 4, 5, 10]
>>> a.append(20)
>>> a
[1, 2, 3, 4, 5, 10, 20]
>>> a.pop()
20
>>> a.pop()
10
```

- 먼저 변수 `a`에는 `[1, 2, 3, 4, 5]`가 할당된다.
- 변수 `a`에 10과 20을 추가하면, 변수 `a`에는 `[1, 2, 3, 4, 5, 10, 20]`이 할당된다.
- `pop()` 함수를 처음 실행하면, 가장 마지막에 저장된 20이 추출되면서 화면에 출력되고, 동시에 변수 `a`의 값은 `[1, 2, 3, 4, 5, 10]`으로 변한다.
- 다시 `pop()` 함수를 실행하면, 마지막에 저장된 10이 추출되면서 화면에 출력되고, 동시에 변수 `a`의 값은 `[1, 2, 3, 4, 5]`로 변한다.

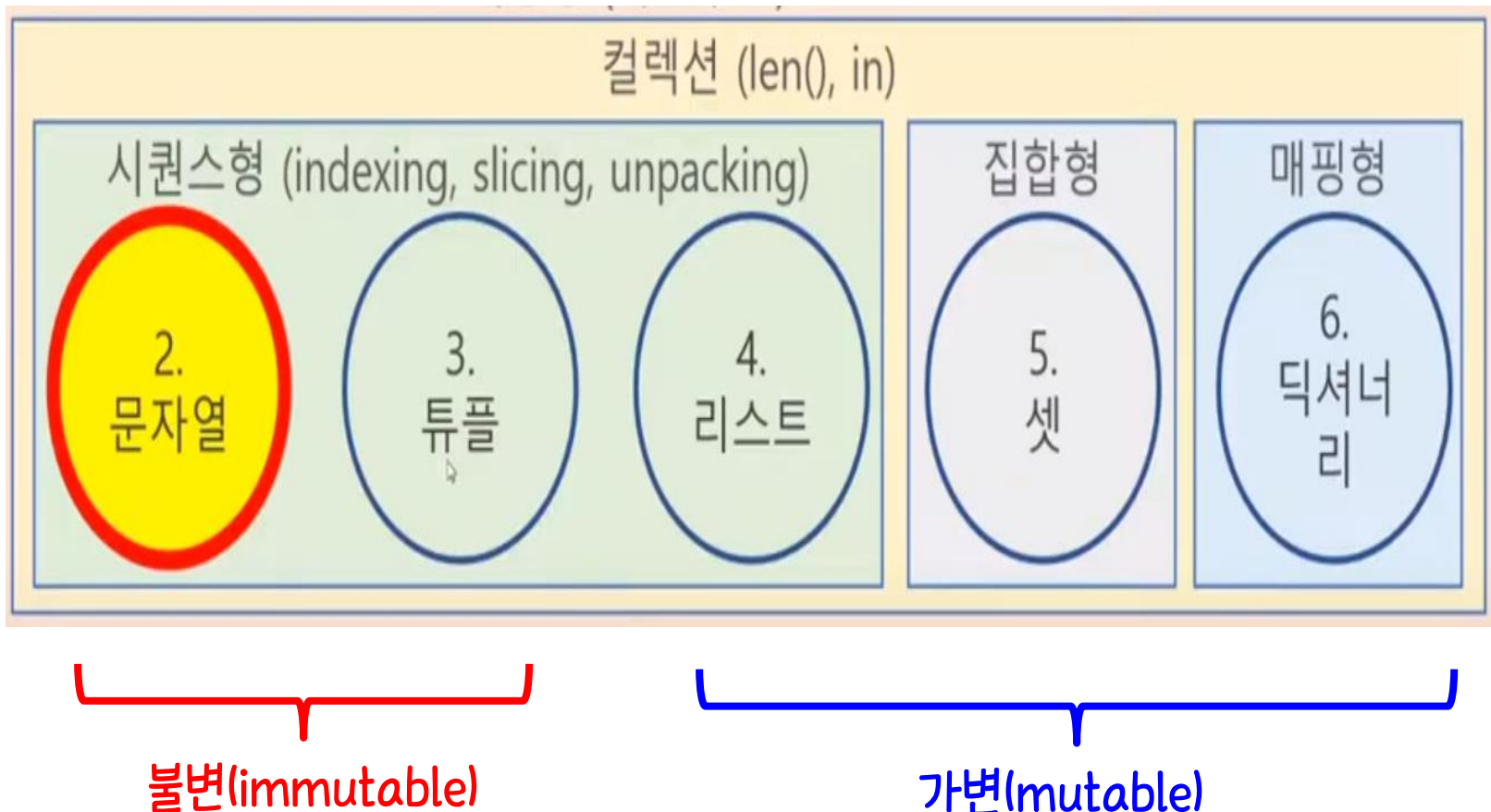
## ❖ 큐

- 큐(queue) : 먼저 들어간 데이터가 먼저 나오는 'Fist in First Out(FIFO)'의 메모리 구조를 가지는 저장 체계이다.
- 파이썬에서 큐를 구현하는 것은 `pop()` 함수를 사용할 때 인덱스가 0번째인 값을 쓴다는 의미로 `pop(0)`을 사용하면 된다.
- `pop()` 함수가 리스트의 마지막 값을 가져온다고 하면, `pop(0)`은 맨 처음 값을 가져온다는 뜻이다.

```
>>> a = [1, 2, 3, 4, 5]
>>> a.append(10)           # a = [1, 2, 3, 4, 5, 10]
>>> a.append(20)          # a = [1, 2, 3, 4, 5, 10, 20]
>>> a.pop(0)
1
>>> a.pop(0)
2
```

# 파이썬의 내장 자료구조

자료구조란 자료를 효율적으로 이용할 수 있도록, 컴퓨터에 저장하는 방법이다.  
데이터 값의 모임, 데이터 간의 관계, 그리고 데이터에 적용할 수 있는 함수로 구성



## ❖ 시퀀스(여러 자료를 순서대로 넣는다는 뜻)

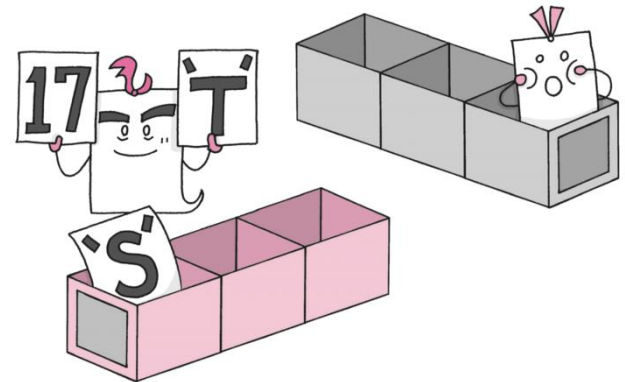
정수로 인덱싱(indexing)될 수 있는 유한한 길이의 순서가 있는 집합이며 슬라이싱도 지원한다.

- $x \text{ in } s; x \text{ not in } s$
- $s1 + s2$
- $s * n$
- 인덱싱 :  $s[0], s[1], s[2], \dots s[-1]$
- 슬라이싱 :  $s[i:j], s[i:j:z]$
- 패킹과 언패킹 : 개별 데이터로 시퀀스를 생성하고 시퀀스를 각각의 개별 데이터로 만드는 것
- 길이 :  $\text{len}(s)$
- 최소값 :  $\text{min}(s)$
- 최대값 :  $\text{max}(s)$
- 지정된 값의 개수 :  $s.\text{count}(x)$
- for문으로 순회하여 개별 문자 순서대로 꺼냄

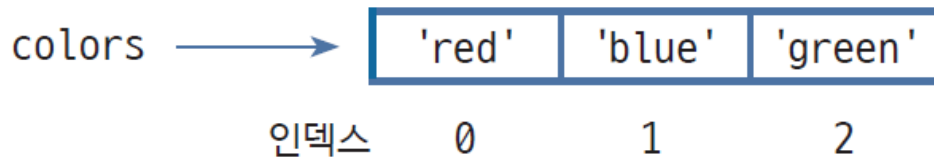
# 리스트

## 순서를 적용해서 여러 개의 데이터를 담는 상자

- 다른 언어에서는 배열이라고 함
- 리스트는 여러 데이터를 모아서 관리하는 파이썬의 자료형
- 리스트의 각 데이터를 요소라 함
- 각 요소는 순서에 따라 인덱스가 부여됨
- 하나의 리스트에 다양한 자료형의 데이터를 저장할 수 있음



```
colors = ['red', 'blue', 'green']
```





# 리스트

```
>>> cities = ['서울', '부산', '인천', '대구', '대전', '광주', '울산', '수원']
```

|     |  |   |   |   |   |   |   |   |
|-----|--|---|---|---|---|---|---|---|
| 값   | ['서울', '부산', '인천', '대구', '대전', '광주', '울산', '수원'] |   |   |   |   |   |   |   |
| 인덱스 | 0  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

|     |  |    |    |    |    |    |    |    |
|-----|--|----|----|----|----|----|----|----|
| 값   | ['서울', '부산', '인천', '대구', '대전', '광주', '울산', '수원'] |    |    |    |    |    |    |    |
| 인덱스 | -8   | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

## ❖ 이차원 리스트

- 리스트를 효율적으로 활용하기 위해 여러 개의 리스트를 하나의 변수에 할당하는 이차원 리스트를 사용할 수 있다.
- 이차원 리스트는 다음과 같은 표의 값들을 다루고자 할 때 사용하는 자료구조이다.

| 학생    | A  | B  | C  | D   | E   |
|-------|----|----|----|-----|-----|
| 국어 점수 | 49 | 79 | 20 | 100 | 80  |
| 수학 점수 | 43 | 59 | 85 | 30  | 90  |
| 영어 점수 | 49 | 79 | 48 | 60  | 100 |

## ❖ 이차원 리스트 생성

- 이차원 리스트를 하나의 변수로 표현하기 위해서는 다음과 같이 코드를 작성한다.

```
>>> kor_score = [49, 79, 20, 100, 80]
>>> math_score = [43, 59, 85, 30, 90]
>>> eng_score = [49, 79, 48, 60, 100]
>>> midterm_score = [kor_score, math_score, eng_score]
>>> midterm_score
[[49, 79, 20, 100, 80], [43, 59, 85, 30, 90], [49, 79, 48, 60, 100]]
```

- 이차원 리스트에 인덱싱하여 값에 접근하기 위해서는 대괄호 2개를 사용한다.

```
>>> print(midterm_score[0][2])
20
```

앞의 대괄호는 행에 대한 인덱스 뒤의 대괄호는 열에 대한 인덱스로 사용된다.

## ❖ 이차원 리스트 생성

- 이차원 리스트를 하나의 변수로 표현하기 위해서는 다음과 같이 코드를 작성한다.

```
>>> kor_score = [49, 79, 20, 100, 80]
>>> math_score = [43, 59, 85, 30, 90]
>>> eng_score = [49, 79, 48, 60, 100]
>>> midterm_score = [kor_score, math_score, eng_score]
>>> midterm_score
[[49, 79, 20, 100, 80], [43, 59, 85, 30, 90], [49, 79, 48, 60, 100]]
```

- 이차원 리스트에 인덱싱하여 값에 접근하기 위해서는 대괄호 2개를 사용한다.

```
>>> print(midterm_score[0][2])
20
```

앞의 대괄호는 행에 대한 인덱스 뒤의 대괄호는 열에 대한 인덱스로 사용된다.

- 튜플(tuple)은 리스트와 유사하나, 읽기 전용 임
- 읽기 전용인 만큼 제공되는 함수도 리스트에 비해 적지만, 속도가 빠름
- 튜플에서 제공되는 메소드 : count, index
- 튜플을 구성하는 각 요소 사이에는 ,를 붙인다. 하나의 요소로 튜플을 생성하려는 경우 요소뒤에 ,를 붙인다.
- 튜플(tuple)은 ' ( ' 과 ' ) ' 으로 둘러싼다.
- 튜플은 적은 메모리 공간을 사용하므로 메모리를 절약할 수 있음
- 함수의 인자들은 튜플로 전달된다.
- 위경도 좌표나 RGB 색상처럼 작은 규모의 자료 구조를 구성하기에 적합
- 리스트는 주로 동일한 자료형으로 이루어진 항목을 순차적으로 추출하는 용도로 사용되며, 튜플을 서로 다른 종류의 데이터형의 항목을 변수에 바로 풀어 쓰는 unpacking 혹은 색인을 매기는 용도로 사용

# 딕셔너리

- 딕셔너리(dictionary) : 전화번호부와 같이 키(key)와 값(value) 형태로 데이터 저장

딕셔너리 변수 = {키 1:값 1, 키 2:값 2, 키 3:값 3, ...}

| 학번(키)    | 이름(값)    |
|----------|----------|
| 20140012 | Janhyeok |
| 20140059 | Jiyong   |
| 20150234 | JaeHong  |
| 20140058 | Wonchul  |

```
>>> student_info = {20140012:'Sungchul', 20140059:'Jiyong', 20140058:'JaeHong'}
```

- 특정 값을 호출하는 방법 : 해당 값의 키를 대괄호 [ ] 안에 넣어 호출할 수 있다.

```
>>> student_info[20140012]
'Sungchul'
```

- 재할당과 데이터 추가이다

```
>>> student_info[20140012] = 'Janhyeok'
>>> student_info[20140012]
'Janhyeok'
>>> student_info[20140039] = 'Wonchul'
>>> student_info
{20140012: 'Janhyeok', 20140059: 'Jiyong', 20140058: 'JaeHong', 20140039: 'Wonchul'}
```

# 딕셔너리

- 딕셔너리의 내용을 얻기 위해서는 `items()`, `keys()`, `values()` 메소드를 사용
- `items()` 는 딕셔너리의 모든 키와 값을 튜플로 묶어서 반환
- `keys()` 는 키, `values()`는 값만 반환
- key는 정렬 안된 상태이며, 색인으로 value를 선택할 수 없다
- key로 검색해서 읽기 위해 유일성을 유지해야 하므로 키를 생성할 때 hash 알고리즘을 통해 유일한 값만 구성해준다.
- key는 변경이 불가능한 자료형들 (int, float, tuple, str)으로만 만들어진다.

```
>>> color = {'apple':'red', 'banana':'yellow'}
>>> print( type (color) )
>>> color[ "cherry" ] = "red"
>>> print( color )
>>> d = dict ( a=1, b=3, c=5 )
>>> print ( type ( d ) )
```



# 집합(세트)

## 순서는 없고 중복되지 않는 여러 개의 데이터를 담는 상자

- 세트(set) : 값을 순서 없이 저장하면서 중복을 불허하는 자료형이다.
- 세트는 튜플과 다르게 삭제나 변경이 가능하며, 다양한 집합 연산 (교집합, 합집합) 을 제공한다.
- 세트(set)는 수학시간에 배운 집합과 동일
- 값은 버리고 키만 남은 딕셔너리와 같다
- 세트는 리스트와 마찬가지로 값들의 모임이며, 인덱스에 의한 사용 불가
- set 생성 : set() : 리스트, 튜플, 문자열 등의 하나의 데이터셋만 지정 가능

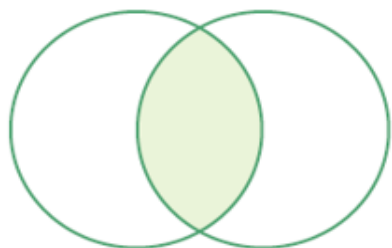
중괄호{ }안에 콤마로 구분된 하나 이상의 데이터 값들을 지정 가능

- 딕셔너리에 set()을 사용하면 키만 사용한다.
- 빈 세트형은 반드시 set()함수로만 가능

# 집합(셋)

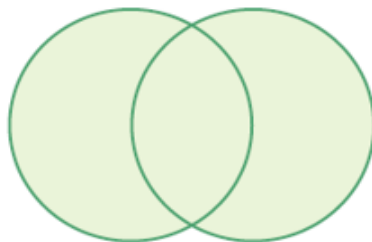


| 연산  | 함수           | 기호 | 예시                           |
|-----|--------------|----|------------------------------|
| 합집합 | union        |    | s1.union(s2), s1   s2        |
| 교집합 | intersection | &  | s1.intersection(s2), s1 & s2 |
| 차집합 | difference   | -  | s1.difference(s2), s1 - s2   |



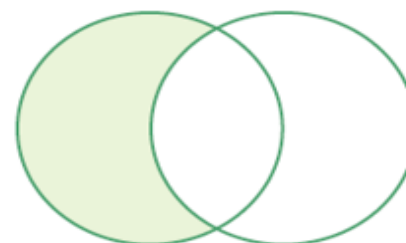
집합 1    집합 2

(a) 교집합



집합 1    집합 2

(b) 합집합



집합 1    집합 2

(c) 차집합

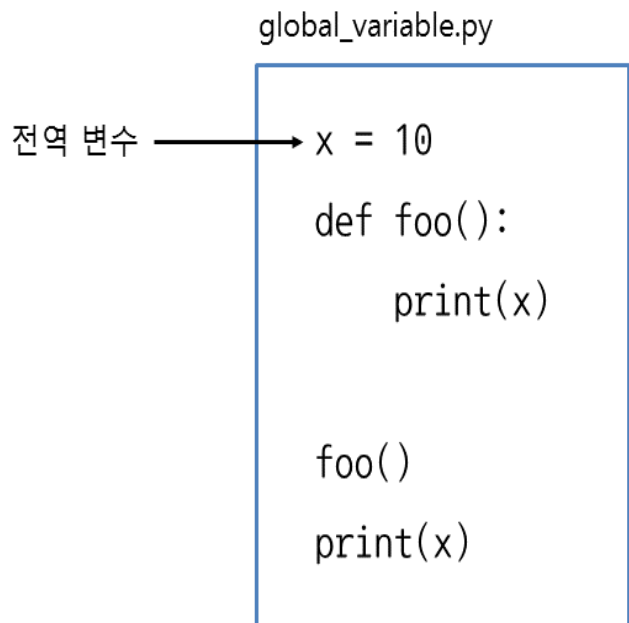
## ■ 변수의 사용 범위

- 변수의 사용 범위(scoping rule) : 변수가 코드에서 사용되는 범위
- 지역 변수(local variable) : 함수 안에서만 사용
- 전역 변수(global variable) : 프로그램 전체에서 사용

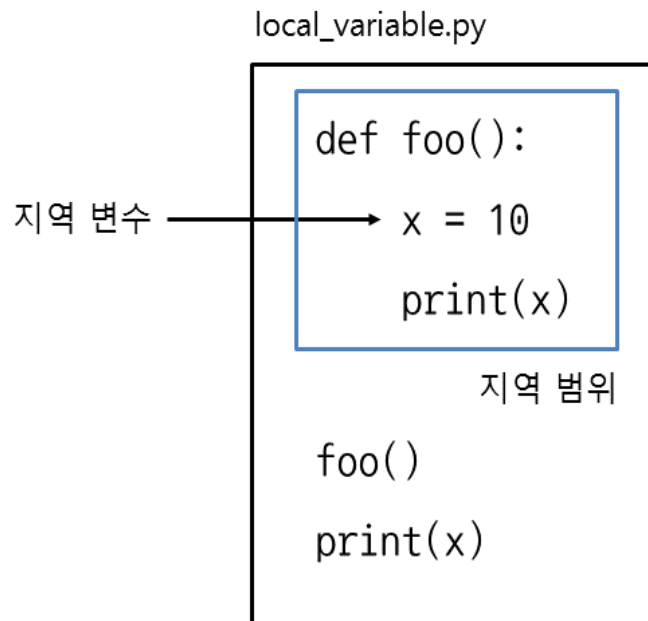
```
1 def test(t):  
2     print(x)  
3     t = 20  
4     print("In Function:", t)  
5  
6 x = 10  
7 test(x)  
8 print("In Main:", x)  
9 print("In Main:", t)
```

```
10  
In function: 20  
In Main: 10  
Traceback (most recent call last):  
  File "scoping_rule.py", line 9, in <module>  
    print("In Main:", t)  
NameError: name 't' is not defined
```

## ■ 변수의 사용 범위



전역 범위



지역 범위

## ■ 변수의 사용 범위

```
1 def f():  
2     s = "I love London!"  
3     print(s)  
4  
5 s = "I love Paris!"  
6 f()  
7 print(s)
```



```
I love London!  
I love Paris!
```

함수 안과 밖의 `s`는 동일한 이름을 가졌지만, 다른 메모리 주소를 가진 전혀 다른 변수이다  
함수 안의 `s`는 해당 함수가 실행되는 동안에만 메모리에 있다가 함수가 종료되는 순간 사라진다.  
함수 밖의 `s`와는 메모리 주소가 달라 서로 영향을 주지 않는다.

## ■ 변수의 사용 범위 :

- 함수 내에서 전역 변수로 선언된 변수를 사용하기 위해서는 **global**이라는 파이썬에서 제공하는 키워드를 사용해야 한다.

```
1 def f():  
2     global s  
3     s = "I love London!"  
4     print(s)  
5  
6 s = "I love Paris!"  
7 f()  
8 print(s)
```

```
I love London!  
I love London!
```



## 재귀 함수

- 재귀 함수(recursive function) : 함수가 자기 자신을 다시 부르는 함수이다.

$$\begin{aligned} 1! &= 1 \\ 2! &= 2(1) = 2 \\ 3! &= 3(2)(1) = 6 \\ 4! &= 4(3)(2)(1) = 24 \\ 5! &= 5(4)(3)(2)(1) = 120 \end{aligned}$$
$$n! = n \cdot (n-1) \cdots 2 \cdot 1 = \prod_{i=1}^n i$$

팩토리얼(factorial) 함수이다.

‘n!’로 표시하면  $n! = n \times (n-1)!$ 로 선언할 수 있다. 자신의 숫자에서 1씩 빼면서 곱하는 형식이다.

## 재귀 함수

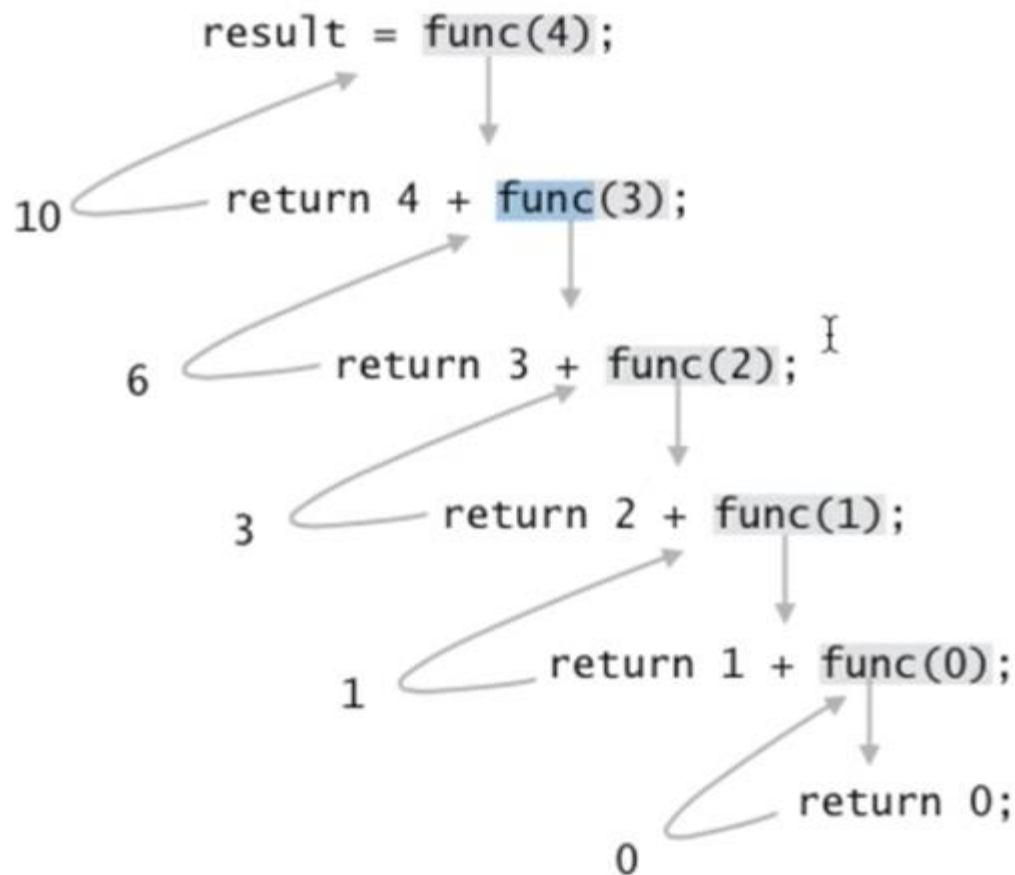
- factorial( ) 함수는 n의 변수를 입력 매개변수로 받은 후  $n == 1$ 이 아닐 때까지 입력된 n과 n에서 1을 뺀 값을 입력값으로 하여 자신의 함수인 factorial( )로 다시 호출한다.

```
1 def factorial(n):
2     if n == 1:
3         return 1
4     else:
5         return n * factorial(n - 1)
6
7 print(factorial(int(input("Input Number for Factorial Calculation: "))))
```

```
5 * factorial(5 - 1)
= 5 * 4 * factorial(4 - 1)
= 5 * 4 * 3 * factorial(3 - 1)
= 5 * 4 * 3 * 2 * factorial(2 - 1)
= 5 * 4 * 3 * 2 * 1
```



## 재귀 함수



## ■ 키워드 아규먼트

- 함수 호출시 전달되는 아규먼트들은 정의 위치에 따라서 매개변수에 전달됨. 아규먼트 앞에 변수명을 지정하면 위치에 관계없이 전달 가능

```
def normalfn(p1, p2, p3) : pass
```

```
normalfn(10,20,30) # 포지션 아규먼트
```

```
normalfn(p3=10,p1=20,p2=30) # 키워드 아규먼트
```

## ■ 디폴트 매개변수(선택적 아규먼트)

- 디폴트 인수(default arguments) : 매개변수에 기본값을 지정하여 사용하고, 아무런 값도 인수로 넘기지 않으면 지정된 기본값을 사용하는 방식

```
def defaultfn1(p1=10, p2="abc", p3=True) : pass
```

```
def defaultfn1(p1, p2="abc", p3=True) : pass
```

## 가변 아규먼트

- 함수의 매개변수 개수를 정하지 않고 진행하는 경우, 사용하는 것이 가변 아규먼트(variable-length arguments)이다. 가변 아규먼트는 매개변수명 앞에 \* 을 붙여서 사용하며 디폴트 매개변수는 관계없지만 위치 매개변수(기본값이 없는 매개변수)의 경우엔 뒤에 와야 한다.

```
def printdeco(*p, deco="@") : pass  
def getlist1(times, *nums) : pass
```

```
printdeco("가", "나")  
printdeco(True, "A", 10, deco="$")  
  
print(getlist1(2, 1,2,3,4,5))
```



## ■ 키워드 가변 아규먼트

- 키워드 가변 아규먼트를 전달받는 매개변수는 **\*\*** 를 변수 앞에 붙인다.
- 키워드 가변 아규먼트를 전달받는 매개변수는 반드시 모든 매개변수의 맨 마지막 선언 한다.

```
def calcstep(**args): pass  
def asterisk_test(a, **kargs) : pass
```

```
calcstep(begin=3, end=5, step=1)
```

```
calcstep(step=1, end=5, begin=3)
```

```
asterisk_test(1, b=2, c=3, d=4, e=5, f=6)
```

## ■ docstring

- 함수 선언문과 본체 사이에 작성하는 문자열
- 함수의 사용법, 인수의 의미, 주의사항 등 설명 작성
- 실행에는 영향 없음

```
def calcsun(n):  
    """1 ~ n까지의 합계를 구해 리턴한다."""  
    sum = 0  
    for i in range(n+1):  
        sum += i  
    return sum  
  
help(calcsun)
```

## ■ 좋은 코드의 의미

- 프로그래밍은 팀플레이(team play)이다. 좋은 프로그래밍을 하는 규칙이 있어야 한다.
- 가독성 좋은 코드를 작성하기 위해서는 여러 가지 필요하지만, 일단 여러 사람의 이해를 돕기 위한 규칙이 필요하다. 프로그래밍에서는 이 규칙을 일반적으로 코딩 규칙(coding convention)이라고 한다.

"컴퓨터가 이해할 수 있는 코드는 어느 바보나 다 짤 수 있다. 좋은 프로그래머는 사람이 이해할 수 있는 코드를 짤다."  
- 마틴 파울러

## ■ 코딩 규칙

- 들여쓰기는 4 스페이스
- 한 줄은 최대 79자까지
- 불필요한 공백은 피함
- PEP 8 (Python Enhance Proposal 8) : 파이썬 개발자들이 앞으로 필요한 파이썬의 기능이나 여러 가지 부수적인 것을 정의한 문서이다.



## ■ PEP 8의 코딩 규칙

- = 연산자는 1칸 이상 띄우지 않는다

```
variable_example = 12      # 필요 이상으로 빈칸이 많음  
variable_example = 12      # 정상적인 띄어쓰기
```

- 주석은 항상 갱신하고, 불필요한 주석은 삭제한다.

```
lI00 = "Hard to Understand"    # 변수를 구분하기 어려움
```

- 소문자 l, 대문자 O, 대문자 I는 사용을 금한다.
- 함수명은 소문자로 구성하고, 필요하다면 밑줄로 나눈다.



## ■ 함수 개발 가이드라인 : 함수 이름

- 함수는 가능하면 짧게 작성할 것(줄 수를 줄일 것)
- 함수 이름에 함수의 역할과 의도를 명확히 드러낼 것

```
def print_hello_world():  
    print("Hello, World")  
def get_hello_world():  
    return "Hello, World"
```





## ■ 함수 개발 가이드라인 : 함수의 역할

- 함수는 한 가지 역할을 명확히 해야 한다.
- 이름에 맞는 최소한의 역할을 할 수 있도록 작성해야 한다.

```
def add_variables(x, y):  
    return x + y
```

```
def add_variables(x, y):  
    print(x, y)  
    return x + y
```

## ■ 함수 개발 가이드라인 : 함수를 만드는 경우

- 공통으로 사용되는 코드를 함수로 변환

```
1 a = 5
2 if (a > 3):
3     print("Hello World")
4     print("Hello TEAMLAB")
5 if (a > 4):
6     print("Hello World")
7     print("Hello TEAMLAB")
8 if (a > 5):
9     print("Hello World")
10    print("Hello TEAMLAB")
```



```
1 def print_hello():
2     print("Hello World")
3     print("Hello TEAMLAB")
4
5 a = 5
6 if (a > 3):
7     print_hello()
8
9 if (a > 4):
10    print_hello()
11
12 if (a > 5):
13    print_hello()
```

## ■ 함수 개발 가이드라인 : 함수를 만드는 경우

- 복잡한 로직이 사용되었을 때, 식별 가능한 이름의 함수로 변환

```
1 import math
2 a = 1; b = -2; c = 1
3
4 print((-b + math.sqrt(b ** 2 - (4 * a * c))) / (2 * a))
5 print((-b - math.sqrt(b ** 2 - (4 * a * c))) / (2 * a))
```



```
1 import math
2
3 def get_result_quadratic_equation(a, b, c):
4     values = []
5     values.append((-b + math.sqrt(b ** 2 - (4 * a * c))) / (2 * a))
6     values.append((-b - math.sqrt(b ** 2 - (4 * a * c))) / (2 * a))
7     return values
8
9 print(get_result_quadratic_equation(1,-2,1))
```