List 다루기

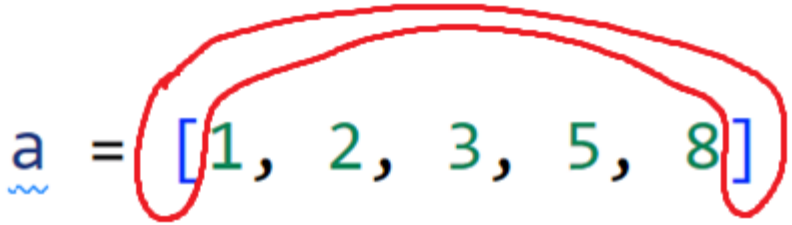
민코딩

List

◉ **여러 Elements를 저장하고 싶을 때 사용**

- 1, 2, 3, 5, 8 는 각각 Element(=요소) 라고 한다.
- **a는 리스트를 할당한다. 그리고,**
  할당된 리스트에는 1, 2, 3, 5, 8 이 담겨있는 것이다.

- **쉽게 이해할수있는 예시**
  - **a는 나이키 박스(리스트) 자체에 이름을 붙인 것(할당)이다.**
  - **나이키 박스 안에 들어있는 신발을 할당한 것이 아니다.**



a에는 [ ] 라는 리스트가 할당 된 것이지,
리스트안에 담겨있는 값이 할당된 것이 아니다.

List 원리

◉ **변수 a에는 리스트를 할당한다.**

- 할당된 리스트 안에 값을 변경할 수 있다.

```
a = [1, 2, 3, 5, 8]

a[3] = 100

print(a)
```

```
[1, 2, 3, 100, 8]
```

List 원리

◉ **할당된 리스트 안에 값을 추가 / 삭제가 가능하다.**

- a에 빈 리스트을 할당해둔다.
- 할당된 리스트에 값을 추가 / 삭제가 가능하다.

```
a = []
a.append(3)
a.append(5)
a.append(6)
a.remove(3)
print(a)
```

```
[5, 6]
```

[도전] Quiz

◉ **총 몇번 할당이 이뤄질까?**

- 그리고 출력 결과는?

```
a = []
a = [1, 1, 1]
a.append(1)
a = [2, 2, 2, 6]
a.append(5)
print(a)
```

index로 접근 = indexing

- arr[index 번호] 로 element를 접근할 수 있다.
- arr 리스트는 [0] 부터 [3] 번 index까지 element가 존재한다.

```
arr = [1, 5, 4, 7]

print(arr[1] + arr[2])
```

```
9
```

배열 전체 순회하기

◉ **List 전체 값 순회하면서 출력하기**

- 의사가 병실을 순회하는 것 처럼
  for문으로 각 리스트의 Element들을 순회하면서 출력하는 코드

◉ **for 순회방식**

- indexing 방식 for 순회 : index로 순회
- iterator 방식 for 순회 : 값 자체를 직접 순회

◉ **파이썬에서 권장되는 방식은**
iterator(이터레이터) 방식이 더 권장된다.

- 파이썬 진영에서는 간결성을 강조함.

```
arr = [1, 5, 4, 7]

for i in range(4):
    print(arr[i], end = '')
print()

for num in arr:
    print(num, end = '')
```

```
1547
1547
```

[도전] 리스트 순회하기

◉ 리스트를 먼저 초기화한다.

1	5	2	7	3	6

◉ 리스트에서 맨 앞과 맨 끝수 출력

◉ 그 다음줄에 리스트에서 홀수 번째만 출력해보기

◉ 그 다음줄에 리스트에서 3 ~ 6 사이 수만 출력하기

```
arr = [1, 5, 4, 7]

#마지막 값을 출력하는 두 가지 방법
a = arr[len(arr) - 1]
a = arr[-1]

print(a)
```

위 도전문제를 풀기 위한 힌트

리스트 초기화 방법

- **총 6칸으로 구성된 List를 만들어야 하는 경우가 있다.**
- **어떤 코드가 더 파이써닉(Pythonic)한가?**
  - 파이써닉 : Java, C++ 등 다른 언어스럽게 코딩하는 것이 아닌,
    파이썬스럽게 코딩하는지를 나타냄 (사랑스럽다.. 이런 느낌으로 파이써닉하다 이렇게 사용)
  - 파이썬 개발자 진영에서 자주 사용되는 말
  - 파이써닉한 코드 = 간결하면서, 명확하게 의도가 표현되는 코드

0	0	0	0	0	0

```
arr = []
for i in range(6):
    arr.append(0)
```

VS

```
arr = [0] * 6
```

[참고] 파이써닉한 코드 작성하기

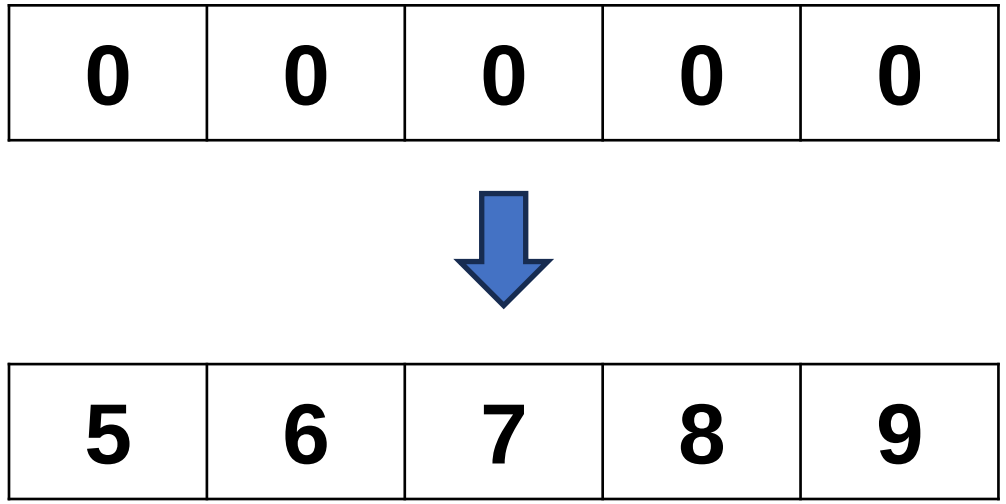
◉ 파이썬을 다룰 때
다른 언어를 개발한 경험이 있는 사람들은
본인이 아는 언어와 비슷하게 코딩하곤한다.

◉ 파이썬 진영에서는
파이썬스럽게 코딩하는 것을 강조하고 있으며,
조금 더 간결한 문법 / 가독성이 좋은 문법 / 더 의도가 잘 들어나는 명확함을
고민하여 쓰자는것을 강조한다.
→ 파이썬 철학

◉ 파이썬 스럽게 코딩하는것. 이를 파이써닉(Pythonic) 하다 라고 표현한다.

[도전] 리스트 값 채우기

- 먼저 0 으로 채워진 5칸으로 구성된 리스트를 만든다.
- 이후 for문을 이용하여 모든 칸을 5에서 하나씩 더한 값으로 채운다.
- 이후 for문으로 순회하여 모든 값을 하나씩 출력해본다.



전체 합계를 for문 순회로 구현하기

소스코드를 이해해보자.

리스트에 모든 숫자들을 모두 더하자.

sum 변수 하나 만든다.

for 문을 돌려 배열의 모든 값을 누적하여 더한다.

```
arr = [1, 5, 3, 4, 4, 2]

total_sum = 0
for num in arr:
    total_sum += num

print(total_sum)
```

파이썬 내장함수인

print(sum(arr)) 로 파이써닉하게 구현할 수 있지만,

프로그래밍 훈련을 위해 직접 구현을 해보자.

함수 다루기

민코딩

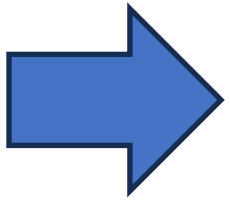
함수란?

함수는 소스코드를 묶어서 나만의 명령어를 만드는 것이다.

BBQ 명령어

```
print('#',end='')
print('#',end='')
print('#')
```

```
BBQ()
BBQ()
```



실행결과
###
###

만든 명령어를 2번 사용하기

함수의 실제 소스코드

함수(명령어)를 사용하기 위해서는, 먼저 함수를 만들어야 한다.

함수를 정의한다 라고 표현한다.

- 예시) 사랑이 무엇인지 내가 정의해볼게.

```
def BBQ():
    print('#',end='')
    print('#',end='')
    print('#')


BBQ()
BBQ()
```

BBQ함수를 두 번 수행한다.

BBQ함수를 두 번 "호출한다" 라고 표현한다.

[도전] 함수 소스코드 실행결과 예측

두 소스코드의 실행 결과를 맞춰보자

```
def fire():
    print('Fire')

def water():
    print('>>>')
    fire()
    print('<<<')

fire()
water()
```

```
def one():
    for i in range(3):
        two()

def two():
    print('HI')

one()
```

지역변수란?

◉ 함수 안에 만들어지는 변수들을 지역 변수라고 한다.

◉ abc 함수 안에 만든 변수 a와
bts 함수 안에 만든 변수 a는
이름만 같을 뿐, 완전히 다른 변수이다!

- 마치 미국사는 찰스랑, 한국에 찰스랑 다른 사람.. 으로 비유 가능

```
def abc():
    a = 10
    b = 20
    print(a + b)

def bts():
    a = 50
    print(a)

abc()
bts()
```

지역변수에서 에러가 발생하는 코드

◉ 함수 안에서 만든 변수 = 지역변수는
함수 밖에서 사용할 수 없다.!

```
def abc():
    a = 10
    b = 20

abc()
print(a + b)
```

```
NameError: name 'a' is not defined
```

전역변수

함수 밖에 만든 변수를 전역 변수라고 한다.

모든 함수들이 이 전역변수를 값을 읽을 수 있다.

- 출력결과 : 100 100

읽을 수만 있으며!

a 값을 함수 안에서 수정할 수 없다!

```
a = 100

def abc():
    print(a)

def bts():
    print(a)

abc()
bts()
```

전역변수

만약 전역변수 a가 존재하는데
지역변수 a를 또 만들었다면,

a 라는 이름의 변수가 2개가 각각 존재하는 것이다.
비추천!! 헷갈리는 코드 (명확성이 떨어짐)
따라서 파이써닉하지 않은 코드이다.

```
a = 100

def abc():
    a = 300
    print(a)

def bts():
    print(a)

abc()
bts()
```

```
300
100
```

전역변수

만약 전역변수를 수정하고 싶다면
global 키워드를 사용하면 된다.

global 키워드

- 함수 내에서, 전역변수를 수정하고 싶을 때 사용하는 키워드

```
a = 100

def abc():
    global a
    a = 300
    print(a)

def bts():
    print(a)

abc()
bts()
```

```
300
300
```

[도전] 함수 사용하기

◉ **input_num, output_num 함수를 호출 한다.**

- input_num 함수에는 a, b 숫자 2개를 입력받는 코드를 추가한다.
- output_num 함수에는 a ~ b 까지 for문을 이용하여 출력하는 코드를 작성

◉ **동작예시**

- 입력 : 3 8
- 출력 : 3 4 5 6 7 8

```
def input_num():
    pass

def output_num():
    pass


output_num()
```

Dictionary

민코딩

List 활용하기

1 ~ 6 까지 수가 각각 몇개 있는지 출력하는 문제

- 출력결과

  1 : 0개

  2 : 2개

  3 : 2개

  4 : 0개

  5 : 0개

  6 : 1개

3	3	2	6	2

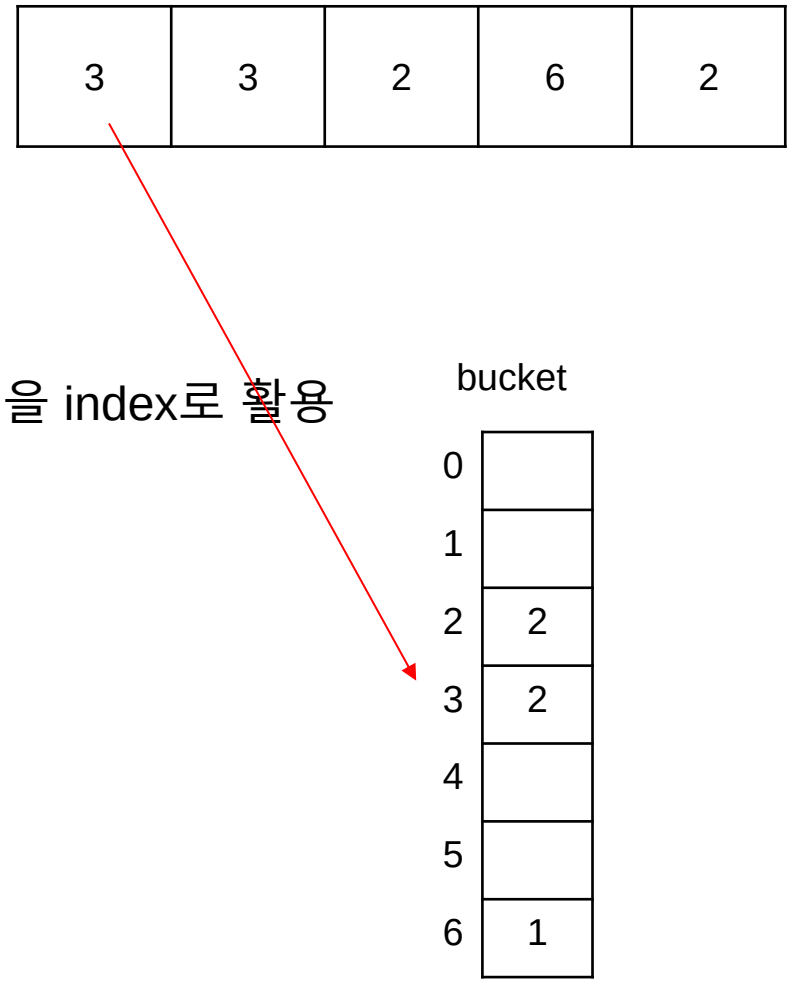
[도전] List 활용하기

1 ~ 6 까지 수가 각각 몇개 있는지 출력

- 추가 배열 1 개와, 1 중 for문으로 구현할 수 있다.
- 빈 배열 만들기 : bucket = [0] * 7

- 출력결과

  1 : 0개
  2 : 2개
  3 : 2개
  4 : 0개
  5 : 0개
  6 : 1개



정답코드

배열을 활용한 해결 방법

```
a = [3, 3, 2, 6, 2]

bucket = [0] * 7

a[3] += 1
a[3] += 1
a[2] += 1
a[6] += 1
a[2] += 1
```

main.py

```
1  a = [3, 3, 2, 6, 2]
2  bucket = [0] * 7
3
4  for i in a:
5      bucket[i] += 1
6
7  for i in range(7):
8      print(str(i) + " : " + str(bucket[i]) + "개")
```

```
0 : 0개
1 : 0개
2 : 2개
3 : 2개
4 : 0개
5 : 0개
6 : 1개
```

리스트로 이 문제를 짤 수 있을까?

ABC / MC / BTS의 개수를 구하는 문제

- 원하는 출력결과

  ABC : 0개

  MC : 2개

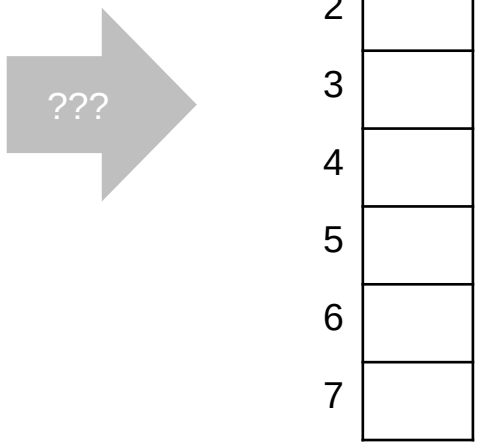
  BTS : 3개

MC	BTS	BTS	MC	BTS

문자열을 index로 쓸 수 없다.

따라서 이 문제는 리스트로 짤 수 없다.

- 문자열 / 음수 일때는 값을 index로 활용할 수 없다.



Dictionary를 사용하는 이유

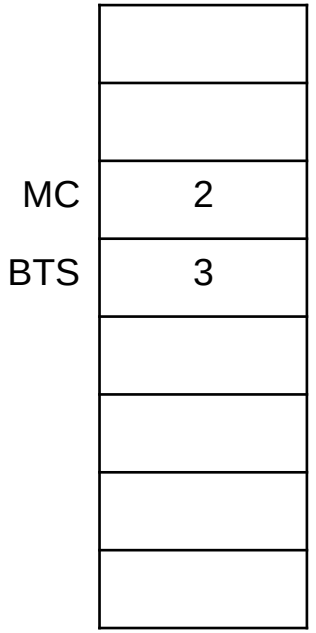
그래서 Dictionary를 사용한다.

- 문자열 / 음수를 index로 사용할 수 없다.

MC	BTS	BTS	MC	BTS

Dictionary 에서 index 역할 하는 값을

Key 값이라고 부른다.



초기화 방법

Dictionary 생성 두 가지 방법

a = dict()
a['HI'] = 55
a['BBQ'] = 'KFC'

또는

a = {'HI' : 55, 'BBQ' : 'KFC'}

main.py

```
1   a = dict()
2   a['HAHA'] = 'KKK'
3   a['OH'] = 345
4   a[-4456] = 15367
5
6   b = {'HI' : 55, 'BBQ' : 'KFC'}
7
8   print(a[-4456])
9   print(b["BBQ"])
10
```

Dictionary 사용하기

배열 index로 "문자열"을 사용 가능

- **다음과 같이 구현 가능 (예시)**

```
dt["MC"] += 1
dt["BTS"] += 1
dt["BTS"] += 1
dt["MC"] += 1
dt["BTS"] += 1
```

MC	BTS	BTS	MC	BTS

1. 리스트 하드코딩으로 초기화
2. 가장 많이 등장하는 단어 출력하기

왼쪽 코드를 for문으로 구현 가능

```
lst = ['MC', 'BTS', 'BTS', 'MC', 'BTS']

di = dict()
for i in lst:
    di[i] = 0

for i in lst:
    di[i] += 1
```

Key를 문자열로, Value를 숫자값으로 둔 Dictionary

하드코딩 : 수 및 데이터를 소스코드에 타이핑으로 적어두는 것 (입력 안받고)

[도전] Dictionary를 이용하여 최빈수 문자열 출력

다시 한번 이 문제를 풀어보자.

- 기존 소스코드를 모두 삭제하고,
  이 문제를 직접 풀어보자.

MC	BTS	BTS	MC	BTS

리스트 하드코딩으로 초기화를 한 후,
가장 많이 등장하는 단어 출력하기

- 정답 : BTS

참고 : 최대값 코드

```
for cnt in di.values():
    if cnt > max_count:
        max_count = cnt
```

하드코딩 : 수 및 데이터를 소스코드에 타이핑으로 적어두는 것 (입력 안받고)

Dictionary 자료구조

Dictionary 는 자주 쓰이나요?

네, 정말 많이 사용됩니다. C를 제외한 다른 언어는 Default 문법

- C언어에는 없다. (직접 구현 필요)
- C++ : Unordered_map 이라는 이름으로 존재
- Java : HashMap 이라는 이름으로 존재
- C# : Dictionary
- Python : Dictionary

많은 프로그래밍 언어에서
기본적으로 들어가 있는 필수 문법이다.
꼭 숙지 / 연습해두자!

Dictionary 사용시 유의사항

다음 소스코드를 이해해보자.

왜 에러가 발생할까?

bbq.py

```
1  dt = {'ABC':15, 'CCC':22}
2
3  dt['ABC'] += 1
4  dt['QQQQ'] += 1
5
```

에러가 나는 코드

Dictionary 사용시 유의사항 - 해결책

Key 공간을 만들어 두어야만, 이후로 값을 대입할 수 있다.

- 아직 공간이 만들어져 있지 않은 곳에, 값을 대입할 수 없다.
  - **값 대입을 하면**, 간단히 Key에 해당하는 공간을 생성할 수 있다.

bbq.py

```
1  dt = {'ABC':15, 'CCC':22}
2
3  dt['ABC'] += 1
4  dt['QQQQ'] += 1
5
```

에러 발생 코드

main.py

```
1  dt = {'ABC':15, 'CCC':22}
2
3  dt['ABC'] += 1
4
5  dt['QQQQ'] = 0
6  dt['QQQQ'] += 1
7
```

정상 코드

[도전] 직접 작성해보자.

아래 배열을 하드코딩으로 초기화를 한다.

이후 한 문자열을 입력 받은 후,

배열 안에 입력 받은 문자열이 몇 개 존재하는지 출력한다.

- 리스트에 없는 문자열은 입력되지 않는다.
- Dictionary 를 이용하여 구현한다. (훈련을 위해 Count 함수 사용 금지)

ABE	53	-99	-99	124

해결방법

str(i)를 Key 값으로 지정한다.

```
list = ['ABE', 53, -99, -99, 124]
d = dict()

for i in list: d[str(i)] = 0
for i in list:
    d[str(i)] += 1

a = input()
print(d[a])
```

사용자 입력은
모두 문자열로 입력을 받게 된다.

따라서 모든 문자열을
str로 변경하여 저장하면 된다.

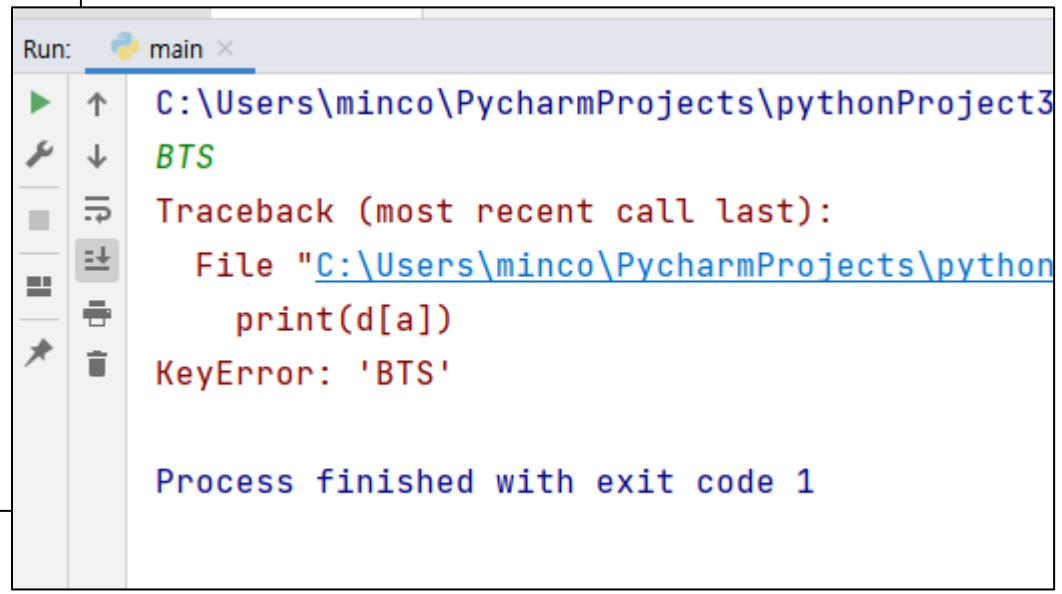
없는 문자열을 입력해보자.

해당 소스코드를

입력값으로 “BTS”를 입력해보자.

- dictionary에 생성되지 않은 Key값을 읽으려는 시도로 인해, Key Error 가 발생했다.

```
list = ['ABE', 53, -99, -99, 124]
d = dict()

for i in list: d[str(i)] = 0
for i in list:
    d[str(i)] += 1

a = input()
print(d[a])
```



Key 값 존재여부 확인하기

Key 값이 존재하는지 확인하는 방법! in을 사용한다. (맴버십 연산자)

- 값을 수정하기 전, Key값이 있는지 검사를 해야 한다.
- 아래 코드에서 ( ) 괄호는 생략할 수 있다.

if 'abc' in dt :

- abc값이 dt 안에 있는지 검사

if 'abc' not in dt :

- abc값이 dt 안에 없는지 검사

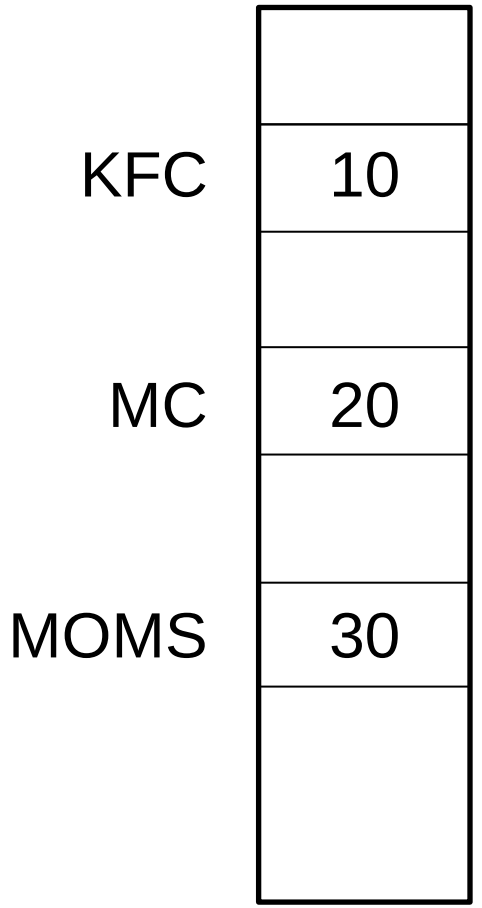
```
di = dict()
for i in lst:
    di[i] = 0

for i in lst:
    di[i] += 1
```

```
di = dict()
for i in lst:
    if (i not in di): di[i] = 0
    di[i] += 1
```

[도전] Dictionary 연습하기 1

문자열 하나를 입력받고, 초기화된 수를 출력하는 프로그램 구현

- Dictionary를 이용하여 문제를 풀 것.
- KFC 입력하면 → 10 출력
  MC 입력하면 → 20 출력
  MOMS 입력하면 → 30 출력
  그 외의 값을 입력 받는 경우 → 1000 출력



모든 Key값을 화면에 출력하기

키 값을 모두 출력한다.

- keys() 메서드 이용하기

keys()와 for 문을 이용하여 모든 키값을 순회 할 수 있다.

```
d = {'KFC':10, 'MC':20, 'MOMS':30}

print(d.keys())
```

```
main
C:\Users\minco\PycharmProjects\pythor
dict_keys(['KFC', 'MC', 'MOMS'])
```

모든 키 값을 출력하는 코드

```
dt = dict()
dt[15] = 1
dt[20] = 2
dt['ABC'] = 5

for i in dt.keys():
    print(i, dt[i])
```

모든 요소를 순회하는 코드

[도전] Dictionary 연습하기

존재하는 단어, 각각 Counting

- 배열에 들어있는 각 단어들 마다
  몇 개가 존재하는지 출력하는 프로그램 작성
  - Key 값 순회하기 방법을 사용해야함.

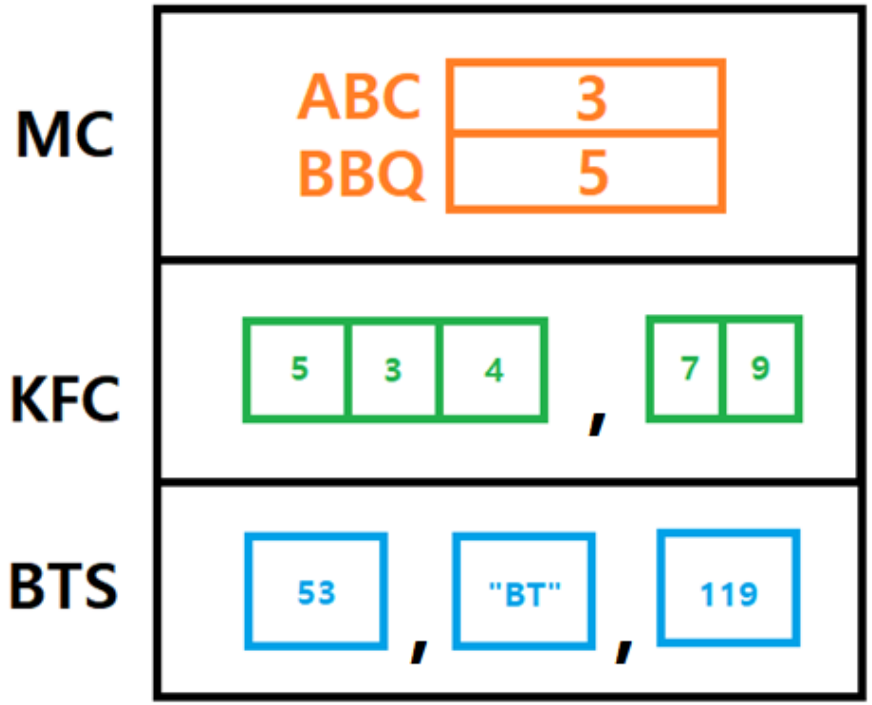
- **출력결과**

  MC : 2 개
  BTS : 3 개

MC	BTS	BTS	MC	BTS

[도전] 코드 이해하기

Dictionary 와 List를 사용하여
왼쪽 그림의 데이터들을 초기화 하는 방법은 다음과 같다.
아래 그림과 소스코드를 이해해보자.



main.py

```
value = {
    "MC" : {
        "ABC":3,
        "BBQ":5
    },
    "KFC" : [
        [5,3,4],
        [7, 9]
    ],
    "BTS" : [53, "BT", 119]
}
```

Tuple

민코딩

Tuple은 값 들의 묶음

내부 값을 변경할 수 없는 "단 하나의 값 덩어리" 이다.

- a = ("HI", 150)
- b = (3, 4, 6)

("HI", 150) 이라는 **값 한덩어리!**
(3, 4, 6) 이라는 **값 한 덩어리!**

튜플을 언제 사용할까?

일반적인 함수에서 값 리턴하기

- 단 하나의 값만 리턴할 수 있다.

여러 값을 리턴 하고 싶을 때,

튜플로 만들면 하나의 값을 리턴하면 된다.

```
def bts():
    return 10

a = bts()
print(a)
```

하나의 값을 리턴하는 예시
(튜플 사용안함)

```
def bts():
    return (3, 4, 5)

a = bts()
print(a)
```

여러개의 값을 리턴하고 싶을때,
튜플을 사용하여 하나의 값 리턴

튜플 문법 살펴보기 1

괄호를 사용 안해도, 튜플로 인식한다.

- 변수 a에는 하나의 값만 들어갈 수 있기 때문
- type(a) 도 출력해보자.

따라서 아래 두 코드는 같다.

- a = (3, 4, 5)
- a = 3, 4, 5

main.py

```
1  a = 3, 4, 5
2
3  print(a)
4
```

(3, 4, 5) 은 값 한 덩어리이다.

main.py

```
1  def abc():
2      return 1, 2, 3, 4, 5, 6
3
4
5  t = abc()
6  print(t)
```

abc 함수는 하나의 값(튜플)을 리턴한다.

튜플 문법 살펴보기 2

튜플 내부를 for로 순회가 가능

- iterator 방식 for문
- indexing으로 for문 순회

```
a = 3, 4, 5

for i in a:
    print(i, end = '')

print()

for i in range(3):
    print(a[i], end = '')
```

튜플 문법 살펴보기 3

튜플 안에 있는 내용들을, 각 변수에 대입 가능

main.py

```
1  a, b, c = "ABC", 14, 555
2
3  print(a)
4  print(b)
5  print(c)
```

("ABC", 14, 555) 튜플에 있는 원소들을
객체 a, b, c 에 각각 대입

main.py

```
t = (1, 2, 3)
a, b, c = t

print(a)
```

(1, 2, 3) 튜플에 있는 원소들을
객체 a, b, c 에 각각 대입

Tuple과 list의 차이점

Tuple 을 구성하고 있는 각각의 값을 Element(=요소) 라고 부른다.
(List도 똑같다.)

Tuple은 하나의 값 덩어리이다.

따라서 Tuple 내부 값(Element) 을 수정 할 수 없다.

bbq.py

```
1  a = (4, 2, 5, 6)
2
3  for i in a:
4      print(i)
5
6  a[0] = 10
```

에러발생!
튜플 element 수정 불가!

리스트 객체

리스트는 여러개의 Element로 구성된 하나의 객체이다.

- Element 값 수정 가능!!
- 비유하자면, 나이키 박스 하나, 여기에 신발들을 담을 수 있음

튜플도 여러개의 Element로 구성된 하나의 객체이다.

- Element 값 수정 불가
- 비유하자면, 나이키 박스 + 신발이 박제된 모형

그럼 튜플 안쓰고

리스트만 써도 되는가??

main.py

```
1  def abc():
2      return [1, 2, 3]
3
4
5  a = abc()
6
7  print(a)
8
```

튜플 대신 리스트를 이용해서 여러개의 값을 리턴할 수 있는 효과를 낼 수 있다.

튜플을 쓰는 이유

두 가지 이유

1. 성능 : 리스트 보다 성능이 빠르다.
2. 안전성 : 다른 개발자가 임의로 내부 값을 수정할 수 없어, 안전한 프로그래밍 가능

본인이 튜플 대신 리스트를 자주 사용하더라도,
파이썬 코드들을 살펴보면 위와 같은 이유로 튜플을 많이 사용된다.
따라서 튜플은 반드시 알고있어야 하는 자료구조이다.

[도전] 소스코드 분석 및 실행

다음 소스코드를 작성하고,
각자 동작을 분석해보자.

bbq.py

```
1  def run(a):
2      for i in a:
3          print(i)
4
5  run((1, 2, 3, 4, 5))
6
```

여는 괄호가 2개 ((
닫기 괄호가 2개))

```
def abc():
    return 1, 2, 3, 4


print(abc())
```

bbq.py

```
1  def run():
2      return 1, 2, 3
3
4  a, b, c = run()
5
6  print(a, b, c)
```

[도전] 단계별로 이해해보자. 1

어떤 값이 출력될까?

main.py

```
1  def abc( value ):
2      a = value[0]
3      print(a)
4
5  abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
```

[도전] 단계별로 이해해보자. 2

어떤 값이 출력될까?

main.py

```
1  def abc( value ):
2      a = value[2]
3      print(a)
4
5  abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
7
```

[도전] 단계별로 이해해보자. 3

어떤 값이 출력될까?

main.py

```
1  def abc( value ):
2      a = value[2][0]
3      print(a)
4
5  abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
```

[도전] 치킨 꺼내기

치킨 브랜드의 이름을 숨겨두었다.

- for문을 돌려, 치킨 브랜드 이름만 출력하는 프로그램을 작성하자.

main.py

```
1    def abc( value ):
2        ???
3
4
5    abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
```

출력결과

KFC
MOMS
BHC

치킨 꺼내기 정답 소스코드

정답코드

main.py

```
1  def abc( value ):
2      a = value[3][2][0]
3      print(a)
4
5  abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
```

for문을 사용한 출력

main.py

```
1  def abc( value ):
2      a = value[3][2][0]
3      for i in a:
4          print(i)
5
6  abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
7
```

[도전] 복잡한 Dictionary 구조

KFC1, 2, 3 형제를 구출하자.

- d["HI"], d["OH"], d[-153] 에서 각각 KFC 값을 꺼내서, print명령어로 출력한다.
- "KFC1 / KFC2 / KFC3 가 출력되는 프로그램 작성하기

KFC1을 꺼내자!
KFC2을 꺼내자!
KFC3을 꺼내자!

main.py

```
1   d = dict()
2
3   d["HI"] = [1, 2, 3, "KFC1"]
4   d["OH"] = [1, 5, {"HO":14, "MY": 119, "QQ" : "KFC2"}]
5   d[-153] = [(1, 2, (5, 6, "KFC3"))]
6
```