

- 파이썬에서 가장 많이 사용하는 자료구조
- 확장된 개념은 DataFrame(데이터프레임)이다.
- 키(key)와 값(value)를 쌍으로 저장하고 관리한다. ☆ index없음
- 키는 중복을 허용하지 않고, 값은 중복이 허용된다. ☆
- 저장된 데이터의 순서는 의미가 없다.
- 중괄호 {} 로 표현
- 콤마(,)를 기준으로 아이템을 구분한다.
- 아이터은 키와 값을 묶어서 부른다.
- 중요함수 : 사전.keys(), 사전.values(), 사전.items()

```
dic = {}
dic['파이썬'] = 'www.python.org'           # 파이썬이라는 키값을 만듦
dic['애플'] = 'www.apple.com'               # 키값을 만들고 해당 딕셔너리를 생성할 수 있다..
dic['마이크로소프트사'] = 'www.microsoft.com'
```

```
'애플' in dic.keys()           'www.apple.com' in dic.values()
```

```
person = {'name' : 'kim', 'phone' : '010-1111-2222', 'birth' : '1118'}
person['name']           → 'kim'
# get() 키는 통해 값을 얻어오는 함수      person.get('phone')
'name' in person         → True
```

- 인덱스가 없고 값만 저장되는 자료구조
- 순서는 의미가 없다.
- 중괄호 {} 로 표현
- 쉼표(,) 를 기준으로 아이템을 구분
- 중복을 허용하지 않는다. ☆

```
s1 = set() # 빈 집합
s2 = set("Hello") # 중복을 허용하지 않음, 순서는 의미가 없음
print(s2) → {'H', 'e', 'l', 'o'}
s1 = set([1, 2, 3]) l1 = list(s1) # s1이라는 집합을 리스트로 만들
t1 = tuple(s1) # 집합을 튜플로 만들
```

```
# 교집합, 합집합, 차집합
s1 = set([1, 2, 3, 4, 5, 6])    s2 = set([4, 5, 6, 7, 8, 9])
# 교집합, &(ampersand), intersection()      s1 & s2      s1.intersection(s2)
# 합집합, |(bar, pipe), union()           s1 | s2      s1.union(s2)
# 차집합, -(minus), difference()          s1-s2        s1.difference(s2)

# add 1개의 값을 추가하는 함수              s1.add(4)
# update 여러 개의 값을 추가하는 함수       s1.update([4, 5, 6])
# remove 특정 값을 제거하는 함수           s1.remove(2)
```

```
# 깊은 복사와 얇은 복사
# 얇은복사 (하나의 주소를 같이 사용)
a = [1, 2, 3, 4, 5]
b = a      # 복사 (얇은복사) - b는 a의 주소를 가저옴
b[0] = 10   # b를 수정하면 a와 같은 값을 사용하므로 a도 수정됨
print(a)    → [10, 2, 3, 4, 5]
print(b)    → [10, 2, 3, 4, 5]

a = [1, 2, 3, 4, 5]
b = a[:]    # 깊은복사 - 주소가 아닌 데이터만 복사함(새로운 공간 만들어짐)
b[0] = 10   # b의 값만 바꿈, a는 그대로
print(a)    → [1, 2, 3, 4, 5]
print(b)    → [10, 2, 3, 4, 5]
```

```
# 날짜 자료형 처리
# 외부의 패키지나 모듈 또는 함수를 불러오는 작업
# import 모듈 : 해당 모듈에 모든 객체를 불러온다.
# from 모듈 import 메서드나 함수 또는 변수 : 해당 모듈의 객체 일부분 불러옴
```

```
from datetime import date, time, datetime, timedelta
```

```
# today() 오늘 날짜를 반환하는 함수
today = date.today()          today.year      today.month
```

```
# 날짜와 시간을 한번에 추출
current_datetime = datetime.today()      current_datetime
```

```
# 형식화된 날짜 출력      today = datetime.today()      # 날짜와 시간을 반환
```

```
# 날짜를 문자열로 변환하여 서식을 이용해서 출력
# %y, %m, %M, %H, %S, %d (year, month, Minute, Hour, Second, day)
d = today.strftime("%y-%m-%d %H:%M:%S")      # strftime = string format time
days = timedelta(days=-1) # 날짜를 이용해서 특정날짜 알아냄      print(today+days) # 어제
days = timedelta(days=-7)      print(today+days) # 1주일 전
days = timedelta(days=30)      print(today+days) # 30일 후
hours = timedelta(hours=-8) # 8시간 전 print(today+hours)      print(hours.days, hours.seconds)
```

04. 콘솔 입출력

표준 입출력함수 : input(), print()

```
a = input("입력하세요 : ")      print("입력값 : ", a)
```

두 개의 정수를 입력받아 그 합을 계산한 후 결과를 출력하는 프로그램

```
a = eval(input("숫자1 = "))      # eval 문자열을 숫자로 변환
```

```
b = eval(input("숫자2 = "))
```

```
c = a+b      print("결과값 : ", c)
```

구구단을 출력하는 프로그램

```
dan = int(input("단을 입력하세요 : "))      # 단을 입력받아 해당 단을 계산하여 출력시킨다.
```

```
# int(문자열) : 문자열을 정수로 변환 // 캐스팅      # eval(문자열) : 문자열을 숫자로 변환
```

반복문 : for 개별변수 in 집합변수

```
for i in range(1, 10) :      # range : 순차적으로 데이터를 읽어옴
```

```
    print("{0} x {1} = {2}".format(dan, i, dan*i))
```

```
for i in range(10) :
```

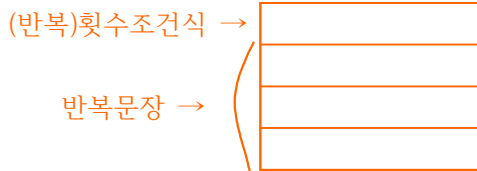
0~9 반복, 기본적으로 반복하면서 줄 바꿈

```
    print(i)
```

```
for i in range(1, 10) :
```

end : 라인 끝에 ~를 해라 (기본값 '\n' 줄바꿈)

```
    print(i, end='\t')
```



[C언어]

for(초기식; 조건식; 증강식)

{ 반복할 문장 }

[파이썬]

중괄호 안쓰고 ':' 사용

05. 제어문 - 조건문(if)과 반복문(for, while)

0. 들여쓰기와 제어문

- 파이썬은 들여쓰기를 강제하여 코드의 가독성을 높인다.
- 블록({}) 내부에 있는 문장들은 반드시 들여쓰기가 일치해야 한다.
- 블록의 시작은 콜론(:)이고, 블록의 끝은 들여쓰기가 끝나는 부분이다.

1. 조건문 or 선택문 or 비교판단문

- 조건식을 가지고 있는 문장
- 주어진 조건식이 참일 때만 해당 문장을 수행한다.
- if문 : 만약 ~라면
- if 논리조건식:

양수와 음수를 판정하는 프로그램

```
num = int(input("숫자 = "))
if num > 0:
    print("양수")
elif num < 0:          # elif = else if
    print("음수")
else:
    print("제로")
```

하나의 숫자를 입력받아 짝수인지 홀수인지 판단

```
num = int(input("숫자 = "))
if num % 2 == 0:
    print("짝수")
else:
    print("홀수")
```

2. 반복문 (for, while)

- 동일하거나 유사한 성격을 가지고 있는 문장
- 조건이 참일 때 반복을 수행
- 반복횟수를 제어할 때 사용
- 형식
- for 개별변수 in 집합변수:
- while 논리조건식:

```
for i in range(5):
    print(1)

for i in range(1,6):
    print(i)
```

