03. 리스트, 튜플, 딕셔너리, 집합, 날짜자료형

```
# 0. 리스트(List)
```

- 임의의 객체를 순차적으로 저장하는 자료형
- sequency DataType, 시퀀스 자료형
- 변경 가능 ☆
- **대괄호** [] 로 표현
- 중요함수 : range() 연속된 데이터를 생성 range(k) : 0부터 k-1까지의 숫자를 리스트로 생성

names = ['홍길동', '장보고', '이순신', '강감찬']

인덱싱 [], 인덱스의 시작은 0부터 print(a[1]) # 슬라이싱 : 리스트에서 여러개의 데이터추출 # [start : end-1]

- # '+' 연산자를 이용해서 리스트 결합이 가능하다.
- # 리스트는 변경이 가능하다. # 리스트의 특정 위치의 값을 참조하고 변경
- # 리스트의 길이를 반환하는 함수 len() len(a)
- # append() : 리스트 끝에 추가하는 함수 a.append(4)
- # '+' 리스트간의 결합을 의미 # extend() a.extend([4, 5, 6])
- # insert(index, data) 정해진 위치에 추가(위치, 데이터) a.insert(2,3)
- # remove() 제거하는 함수 a.remove('VOLKSWAGEN') a.remove(a[0])
- # pop() 마지막 위치의 요소를 제거하는 함수 a.pop()
- # count() 입력된 매개변수와 일치하는 데이터 개수 a.count(100)
- # sort() 정렬하는 함수(기본 오름차순, reverse=True 내림차순) a.sort() a.sort(reverse=True)
- # reverse() 리스트 요소를 뒤집는 함수 a.reverse()

1. 튜플(Tuple)

- 리스트와 유사하지만 튜플 내의 값은 변경할 수 없다.
- 읽기 전용 데이터를 저장할 때 사용
- 변경 불가능 ☆
- 소괄호 () 로 표현

튜플 생성하기 a = (1, 2, 3)

a = 1, 2, 3, 4, 5 # python에 기본 제공하는 packing기능. tuple이 됨

인덱싱과 슬라이싱, [] 대괄호 조건식- 특정 위치값 나타냄. 리스트의 의미아님

print(a[3]) print(a[:3]) # 인덱스 0,1,2 print(a[4:6]) # 인덱스 4,5

참조는 가능, 변경은 불가능 print(a[0]) $a[0] = 7 \rightarrow X$

print(t1+t2) print(a*3) # '*' : 반복

len(t1) # length 길이

패킹과 언패킹(packing, unpacking) - 여러개의 값을 하나로 묶음/나눔

t1 = 1, 2, 3 # 패킹, 괄호 없으면 튜플로 간주함 one, two, three = t1 # 언패킹

aa, bb = t1 # 에러발생; 요소의 개수가 같아야 한다. (변수 2개 ≠ t1 3개)

city, latitude, longitude = 'Seoul', 37.541, 126.986

리스트의 append() ; 뒤에 추가한다. t1.append(4) # 에러발생 ; 튜플은 수정(추가)불가능

튜플이 사용가능한 메서드는 index(), count() 뿐이다.

```
# 2. 딕셔너리(Dictionary)
- 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
- 컴마(,)를 기준으로 아이템을 구분한다.
- 아이템은 키와 값을 묶어서 부른다.
- 중요함수 : 사전.keys(), 사전.values(), 사전.items()
# 딕셔너리 생성
dic = \{\}
dic['파이썬'] = 'www.python.org' # 파이썬이라는 키값을 만듦
                            # 키값을 만들고 해당 딕셔너리를 생성할 수 있다..
dic['애플'] = 'www.apple.com'
dic['마이크로소프트사'] = 'www.microsoft.com'
dic.keys()
                              # 리스트형태로 나타남. 키는 정수(int)를 권장하지 않는다..
dic.values()
                dic.items()
# 'in' 연산자
'애플' in dic.keys()
                       'www.apple.com' in dic.values()
students = {'홍길동':'컴공과', '이순신':'해양과'}
                                               # 홍길동:키, 컴공과:값 - 합쳐서 아이템
# 'pop' 키와 값을 쌍으로 제거하는 함수 (마지막x, 키값 지정)
                                               dic.pop('애플')
# clear() 딕셔너리 요소를 모두 제거하는 함수
                                                dic.clear()
person = {'name' : 'kim', 'phone' : '010-1111-2222', 'birth' : '1118'}
            → 'kim'
person['name']
# get() 키는 통해 값을 얻어오는 함수 person.get('phone')
'name' in person → True
# 3. 집합(Set)
- 인덱스가 없고 값만 저장되는 자료구조
- 순서는 의미가 없다.
- 중괄호 {} 로 표현
- 컴마(,) 를 기준으로 아이템을 구분
- 중복을 허용하지 않는다. ☆
# 집합 생성
```

s1 = set() # 빈 집합 s1 = set([1, 2, 3]) # 리스트로 집합을 만듦

t1 = tuple(s1) # 집합을 튜플로 만듦

s1 = set([1, 2, 3]) l1 = list(s1) # s1이라는 집합을 리스트로 만듦

s2 = set("Hello")

print(s2) → {'H', 'e', 'l', 'o'}

중복을 허용하지 않음, 순서는 의미가 없음

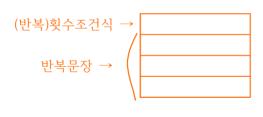
```
# 교집합, &(ampersand), intersection()
                                                   s1.intersection(s2)
                                      s1 & s2
# 합집합, |(bar, pipe), union()
                                      s1 ¦ s2
                                                   s1.union(s2)
# 차집합, -(minus), difference()
                                                   s1.difference(s2)
                                       s1-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 얕은 복사
# 얕은복사 (하나의 주소를 같이 사용)
a = [1, 2, 3, 4, 5]
b = a
            # 복사 (얕은복사) - b는 a의 주소를 가져옴
b[0] = 10
           # b를 수정하면 a와 같은 값을 사용하므로 a도 수정됨
print(a)
            \rightarrow [10, 2, 3, 4, 5]
            \rightarrow [10, 2, 3, 4, 5]
print(b)
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)
            \rightarrow [1, 2, 3, 4, 5]
print(b)
            \rightarrow [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)
```

교집합, 합집합, 차집합

s1 = set([1, 2, 3, 4, 5, 6]) s2 = set([4, 5, 6, 7, 8, 9])

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

