Python_05

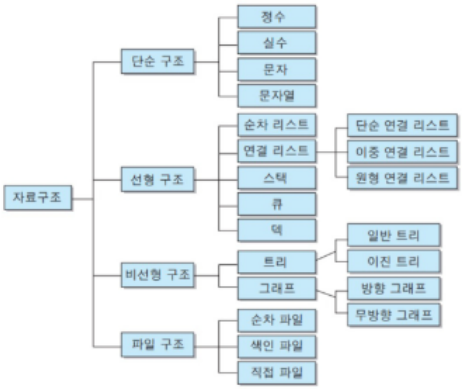
데이터 구조(Data Structure)

데이터 구조란? == 자료 구조

- 여러 데이터를 효과적으로 사용, 관리하기 위한 구조
- 파이썬에는 대표적으로 List(차례대로 저장/인덱스 연산), Tuple, Dict("키-값"의 쌍), Set 등의 데이터 구조가 있음

자료구조

- 컴퓨터공학에서는 '자료구조'라고 함
- 각 데이터의 효율적인 저장, 관리를 위한 구조를 나눠 놓은 것



데이터 구조 활용하기

- 데이터 구조를 활용하기 위해서는 매서드(method)를 사용
  - 메서드는 클래스 내부에 정의한 함수, 사실상 함수 동일
  - 쉽게 설명하지만 객체의 기능

    `데이터구조 . 메서드()`

> ✔️ **Check**
>
> **메서드**
>
> > 객체의 데이터를 처리하는 함수
>
> **클래스**
>
> > 객체를 만들어 내기 위한 틀
> > 연관되어 있는 변수와 메서드의 집합

> **객체(Object)(== 데이터(속성 attribute) + 메소드 == 캡슐화)**
>
> > 메모리(실제 저장공간)에 할당된 것으로 프로그램에서 사용되는 데이터
> > == 메모리 공간에 할당된 데이터
> > - 속성 : 멤버 변수, 특성, 필드, 상태
> > - 기능 : 메소드, 행위, 함수

파이썬 공식 문서의 표기법

- python 구문이 아니며, 문법을 표현하기 위한 것임

```
str.replace(old, new[,count])
```

- old, new는 필수 // [,count]는 선택적 인자를 의미

순서가 있는 데이터 구조

문자열(String)

- 문자들의 나열(sequence of characters)
  - 모든 문자는 str타입(변경 불가능한 immutable)
- 문자열은 작은 따옴표(')나 큰 따옴표(")를 활용하여 표기
  - 문자열을 묶을 때 동일한 문장부호를 활용
  - PEP8에서는 소스코드 내에서 하나의 문장부호를 선택하여 유지하도록 함

```
print('hello') # hello
print(type('hello')) # <class 'str'>

print('철수 "안녕"') # 철수 "안녕"
print("철수 '안녕'") # 철수 '안녕'
```

문자열 조회/ 탐색 및 검증 메서드

문법	설명
s.find(x)	x의 첫 번째 위치를 반환. 없으면, -1을 반환
s.index(x)	x의 첫 번째 위치를 반환. 없으면, 오류 발생
s.isalpha()	알파벳 문자 여부 * 단순 알파벳이 아닌 유니코드 상 Letter (한국어도 포함)
s.isupper()	대문자 여부
s.islower()	소문자 여부
s.istitle	타이틀 형식 여부

	12345	①②③④⑤	I, II, III, IV, V	abc, 12.3, -45
s.isdecimal()	True	False	False	False
s.isdigit()	True	True	False	False
s.isnumeric()	True	True	True	False

문법	설명
s.replace(old, new[,count])	바꿀 대상 글자를 새로운 글자로 바꿔서 반환
s.strip([chars])	공백이나 특정 문자를 제거

문법	설명
s.split(sep=None, maxsplit = -1)	공백이나 특정 문자를 기준으로 분리
‘separator’.join([iterable])	구분자로 iterable을 합침
s.capitalize()	가장 첫 번째 글자를 대문자로 변경
s.title()	문자열 내 띄어쓰기 기준으로 각 단어의 첫글자는 대문자로, 나머지는 소문자로 변환
s.upper()	모두 대문자로 변경
s.lower()	모두 소문자로 변경
s.swapcase()	대 ↔ 소문자 서로 변경

문자열은 immutable(불변형)인데, 문자열 변경이 되는 이유?

- 기존의 문자열을 변경하는 게 아니라, 변경된 문자열을 새롭게 만들어서 반환
  - replace, strip, title

리스트(List)

- 리스트는 여러 개의 값을 순서가 있는 구조로 저장하고 싶을 때 사용
- 리스트는 대괄호([]) 혹은 list()를 통해 생성
  - 파이썬에서는 어떠한 자료형도 저장할 수 있으며, 리스트 안에 리스트도 넣을 수 있음
  - 생성된 이후 내용 변경이 가능 → 가변 자료형
  - 이러한 유연성 때문에 파이썬에서 가장 흔히 사용
- 순서가 있는 시퀀스로 인덱스를 통해 접근 가능
  - 값에 대한 접근은 list[i]

문법	설명
L.append(x)	리스트 마지막에 항목 X를 추가
L.insert(i, x)	리스트 인덱스 i에 항목 X를 삽입
L.remove(x)	리스트 가장 왼쪽에 있는 항목(첫번째) x를 제거 항목이 존재하지 않을 경우, ValueError
L.pop()	리스트 가장 오른쪽에 있는 항목(마지막)을 반환 후 제거
L.pop(i)	리스트의 인덱스 i에 있는 항목을 반환 후 제거
L.extend(m)	순회형 m의 모든 항목들의 리스트 끝에 추가 (+=과 같은 기능)
L.index(x, start, end)	리스트에 있는 항목 중 가장 왼쪽에 있는 항목 x의 인덱스를 반환
L.reverse()	리스트를 거꾸로 정렬
L.sort()	리스트를 정렬(매개변수 이용가능)
L.count(x)	리스트에서 항목 x가 몇 개 존재하는지 갯수를 반환
L.clear()	리스트 초기화

튜플(tuple)

- 튜플은 여러 개의 값을 순서가 있는 구조로 저장하고 싶을 때 사용
  - 리스트와의 차이점은 생성 후, 담고 있는 값 변경이 불가(불변 자료형)
- 항상 소괄호 형태로 사용
- 튜플은 변경할 수 없기 때문에 값에 영향을 미치지 않는 메서드만 지원
- 리스트 메서드 중 항목을 변경하는 메서드들을 제외하고 대부분 동일

```
(2, ) # 값이 하나인 튜플
(2, 3, 4, 5) # 값이 여러개인 튜플
```

연산자(Operator)

맴버십 연산자(Membership Operator)

- 맴버십 연산자 in을 통해 특정 요소가 속해 있는지 여부를 확인
- 포함 여부 확인
  - in
  - not in
- 산술연산자(+)
  - 시퀀스 간의 concatenation(연결/연쇄)
- 반복연산자(*)
  - 시퀀스를 반복

```
list(range(1, 11)) * 3
```

```
# 주어진 문자열에서 숫자, 문자 , 기호가 각각 몇개인지를 판단하는 함수를 작성
# 문자 : 10개, 숫자 : 2개, 기호 : 7개
target_str = 'c1-j=a+!dg~b2@ef#ih'

def check(target_str):
    num_count = 0
    alphabet_count = 0
    etc_count = 0

    for i in target_str:
        if i.isdigit():
            num_count += 1
        elif i.isalpha():
            alphabet_count += 1
        else:
            etc_count += 1
    return(alphabet_count, num_count, etc_count)

a, b, c = check(target_str)
print(f'문자 : {a}개, 숫자 : {b}개, 기호 : {c}개')
```