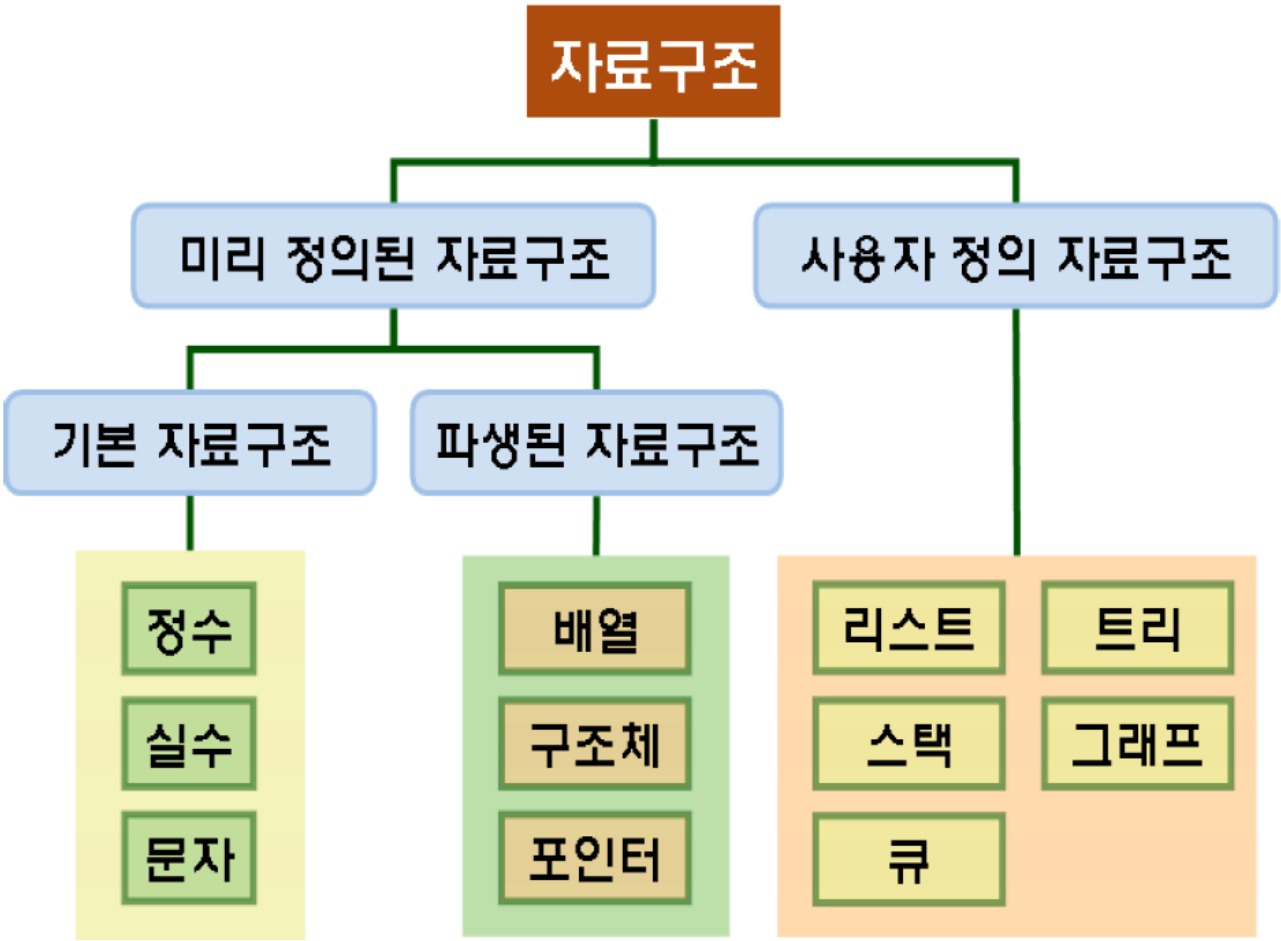
원본 링크 : https://velog.io/@suminwooo/2.-파이썬-프로그래밍-기초-자료형

1. 자료 구조란?



- 대량의 데이터를 효율적으로 관리할 수 있는 데이터의 구조를 의미한다.
- 코드상에서 효율적으로 데이터를 처리하기 위해, 데이터 특성에 따라, 체계적으로 데이터를 구조화를 해야 한다.
- 쉽게 설명하면, 프로그래밍을 할 때 쓰이는 숫자, 문자열 등 자료 형태로 사용하는 모든 것을 의미한다.프로그램의 기본이자 핵심 단위가 바로 자료형이다. 계산 프로그램을 만들려면 어떤 것을 계산할지부터 알아야 하고, 데이터베이스 프로그램을 만들려면 어떤 자료를 저장할지부터 알아야 하는 것처럼 기본 중의 기본이다.

2. 자료형

2-1. 숫자형 자료형

- 정수형 -> int
- 실수형 -> float

```
# 정수형
a = 1
```

```
print(a, type(a))
# 1 <class 'int'>

# 실수형
b = 1.0
print(b, type(b))
# 1.0 <class 'float'>
```

- 기본적인 사칙 연산 가능
- 나눗셈 연산자
    1. / : 나누기
    2. // : 몫 구하기
    3. & : 나머지 구하기

2-2. 문자형 자료형

- 문자형 -> str(), '', "", '''', """"""
- "파이썬은 0부터 숫자를 센다."
- 문자열 인덱싱, 슬라이싱

```
# 문자형
a = '1'
print(a, type(a))
# 1 <class 'str'>

# 문자형 자료형 다양한 함수 활용
example = 'Life is too short, You need Python'

# 인덱싱, 슬라이싱
print(example[0])
# 'e'

print(example[-4:])
# 'thon'

print('example[3] :' ,example[3],', example.split(' ')[3] :' , example.split(' ')
[3])
# example[3] : e , example.split()[3] : short,

# split 함수 활용
print("a b c a b c".split(" "))
# ['a', 'b', 'c', 'a', 'b', 'c']

print(example.split(' '))
# ['Life', 'is', 'too', 'short,', 'You', 'need', 'Python']


# replace 함수 활용
a = "Life is too short"
```

```
a.replace("Life", "Your leg")
# "Your leg is too short"

# strip 함수 활용
a = " hi "
print('왼쪽 공백 제거 : ', a.lstrip(),', 오른쪽 공백 제거 : ',a.rstrip(),', 양쪽 공백 제거 : ',a.strip())
# 왼쪽 공백 제거 :  hi  , 오른쪽 공백 제거 :   hi , 양쪽 공백 제거 :  hi

# join 함수 활용
print(",".join('abcd'))
# 'a,b,c,d'
```

2-3. 리스트 자료형

- 리스트명 = [요소1, 요소2, 요소3, ...]
- 파이썬은 다른 언어와 달리 리스트를 쉽게 생성 및 구성이 가능
- 인덱싱과 슬라이싱이 가능
- 추가, 수정, 삭제가 자유로움
- 예시
    1. a = [] / 빈 리스트
    2. b = [1, 2, 3] / 정수형으로 이루어진 리스트
    3. c = ['Life', 'is', 'too', 'short'] / 문자열로 이루어진 리스트
    4. d = [1, 2, 'Life', 'is'] / 정수 + 문자열로 이루어진 리스트
    5. e = [1, 2, ['Life', 'is']] / 정수 + 리스트로 이루어진 리스트

```
# 리스트 생성
test_lst=[1,2,3]
print(test_lst, type(test_lst))
# [1, 2, 3] <class 'list'>

# 인덱싱
test_lst[1]
# 2

indexing_lst = [1, 2, ['a', 'b', ['Life', 'is']]] # Life 만 꺼낸다면 ?

# 슬라이싱
test_lst[1:]
# [2, 3]

slicing_lst = [1, 2, 3, ['a', 'b', 'c'], 4, 5] # b,c만 꺼낸다면?

# 리스트 연산
a = [1, 2, 3]
b = [4, 5, 6]
print(a+b)
# [1, 2, 3, 4, 5, 6]

# 리스트 관련 함수 - append
```

```
a = [1, 2, 3]
a.append(4)
print(a)
# [1, 2, 3, 4]

# 리스트 관련 함수 - pop
a.pop()
# 4
a.pop(0)
# 1

print(a)
# [2, 3]

# 리스트 관련 함수 - sort, sorted

# sort
sort_lst1 = [1, 4, 3, 2]
sort_lst.sort()
print(sort_lst)
# [1, 2, 3, 4]

sort_lst2 = ['a','c','b']
sort_lst2.sort()
print(sort_lst2)
# ['a', 'c', 'b']

# sorted
sort_lst1 = [1, 4, 3, 2]
sorted(sort_lst1)
# [1, 2, 3, 4]
print(sort_lst1)
# [1, 4, 3, 2]
```

2-4. 튜플 자료형

- 리스트명 = (요소1, 요소2, 요소3, ...)
- 튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷하며 리스트와 다른 점은 다음과 같다.
- 리스트는 [ ]으로 둘러싸지만 튜플은 ( )으로 둘러싼다.
- 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.

```
a = (1,2,3)
print(a, type(a))
# (1, 2, 3) <class 'tuple'>

a.append(4)
# AttributeError: 'tuple' object has no attribute 'append'
```

2-5. 딕셔너리 자료형

- 사람은 누구든지 "이름" = "홍길동", "생일" = "몇 월 며칠" 등으로 구별할 수 있다.
- 파이썬은 영리하게도 이러한 대응 관계를 나타낼 수 있는 자료형을 가지고 있다. 요즘 사용하는 대부분의 언어도 이러한 대응 관계를 나타내는 자료형을 갖고 있는데, 이를 연관 배열(Associative array) 또는 해시(Hash)라고 한다.
- 파이썬에서는 이러한 자료형을 딕셔너리(Dictionary)라고 한다.
- 기본 형태 : {Key1:Value1, Key2:Value2, Key3:Value3, ...}

```
# 딕셔너리 구조
dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
# Key : 각각 'name', 'phone', 'birth'
# Value :  'pey', '0119993323', '1118'
print(dic, type(dict))
# {'name': 'pey', 'phone': '0119993323', 'birth': '1118'} <class 'type'>

# 딕셔너리 생성, 추가, 삭제, 조회
test_dict = {}
test_dict['a'] = 1
test_dict['b'] =2
test_dict['c'] = [1,2,3]
print(test_dict)
# {'a': 1, 'b': 2, 'c': [1, 2, 3]}

del test_dict['a']
print(test_dict)
# {'b': 2, 'c': [1, 2, 3]}

print(test_dict.keys())
# dict_keys(['b', 'c'])

for k in test_dict.keys():
    print(k)
# b
# c

print(test_dict.values())
# dict_values([2, [1, 2, 3]])

print(test_dict.items())
# dict_items([('b', 2), ('c', [1, 2, 3])])

for key,value in test_dict.items():
    print(key, value)
# b 2
# c [1, 2, 3]
```

2-6. 집합 자료형

- 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형
- set 자료형을 정말 유용하게 사용하는 경우는 교집합, 합집합, 차집합을 구할 때이다.

```
s1 = set([1,2,3,1])
print(s1, type(s1))
# {1, 2, 3} <class 'set'>

s2 = set("Hello")
print(s2)
# {'e', 'l', 'H', 'o'}

s1 = set([1, 2, 3, 4, 5, 6])
s2 = set([4, 5, 6, 7, 8, 9])
print(s1 & s2)
# {4, 5, 6}
print(s1.intersection(s2))
# {4, 5, 6}
print(s1 | s2)
# {1, 2, 3, 4, 5, 6, 7, 8, 9}
print(s1.union(s2))
# {1, 2, 3, 4, 5, 6, 7, 8, 9}
print(s1 - s2)
# {1, 2, 3}
print(s2 - s1)
# {8, 9, 7}
print(s1.difference(s2))
# {1, 2, 3}
print(s2.difference(s1))
# {8, 9, 7}
```

2-7. 불 자료형

- 불(bool) 자료형이란 참(True)과 거짓(False)을 나타내는 자료형이다. 불 자료형은 다음 2가지 값만을 가질 수 있다.
    1. True(1) - 참
    2. False(0) - 거짓
- bool 내장 함수를 사용하면 자료형의 참과 거짓을 식별할 수 있다.

```
print(1 == 1)
# True

print(2 < 1)
# False

print(bool('python'))
print(bool(''))
# True
# False

print(bool([1,2,3]))
print(bool([]))
# True
```

```
# False

print(bool(0))
print(bool(3))
# False
# True
```

- 참고 자료
    - https://wikidocs.net/book/1