1일차(파이썬 문법 - 자료형)

💡 자료형

- 정수형, 실수형, 복소수형, 문자열, 리스트, 튜플, 사전 등이 있다.

💡 정수형

- 정수를 다루는 자료형
- 양의 정수, 음의 정수, 0

```
# 양의 정수
a = 1000
print(a) # 1000

# 음의 정수
a = -7
print(a) # -7

# 0
a = 0
print(a) # 0
```

💡 실수형

- 소수점 아래의 데이터를 포함하는 수 자료형
- 소수부가 0이거나, 정수부가 0인 소수는 0을 생략하고 작성 가능

```
# 양의 실수
a = 157.93

# 음의 실수
a = -1837.2

# 소수부가 0일 때 0 생략
a = 5.

# 정수부가 0일 때 0 생략
a = -.8
```

- 파이썬에서는 e나 E를 이용한 지수 표현 방식 이용 가능
  - 1e9 = 10의 9제곱
  - 즉, 유효숫자e^지수 = 유효숫자 x 10^지수
  - 도달할 수 없는 노드에 대하여 최단 거리를 무한(INF)로 설정하곤 하는데 이때 가능한 최댓값이 10억 미만이라면 무한(INF)의 값으로 1e9 사용 가능

```
# 752.5
a = 75.25e1

# 3.954
a=3954e-3
```

- 컴퓨터 시스템은 실수 정보를 표현하는 정확도에 한계를 가진다
  - 10진수에서는 0.3 + 0.6 = 0.9인데
  - 2진수에서는 0.9를 정확히 표현할 방법이 없다.
  - 최대한 0.9와 가깝게 표현하지만, 미세한 오차가 발생한다.
  - 그렇기 때문에 round 함수(반올림)를 사용한다.
  - 예를 들어, 123.456을 소수 둘째자리 까지만 나타내려면 round(123.456, 2) 라고 작성하고, 결과는 123.46이 나온다.

```
a = 0.3 + 0.6 # 0.89999999999

if round(a, 4) == 0.9:
  print(True)

# True
```

- 파이썬에서 나누기 연산자(/)는 나눠진 결과가 실수형이기 때문에
- 몫 연산자(//), 나머지 연산자(%), 거듭제곱 연산자(**) 등을 사용한다.

리스트 자료형

```
a = [1, 2, 3, 4, 5, 6]

print(a[4]) # 5

n = 5
a = [0] * n # [0, 0, 0, 0, 0]

arr = []
for i in range(5):
  arr.append(0)

print(arr) # [0, 0, 0, 0, 0]
```

```
a = [1, 2, 3, 4, 5]
print(a[-1]) # 5
print(a[-3]) # 3

a[3] = 7
print(a) # [1, 2, 3, 7, 5]

a = [1, 2, 3, 4, 5]
print(a[0:2]) # [1, 2]
```

```
arr = [i for i in range(10) if i % 2 == 1]
print(arr) [1, 3, 5, 7, 9]

arr = [i * i for i in range(5)]
print(arr) [1, 4, 9, 16, 25]
```

```
n = 3
m = 4

arr = [m * [0] for _ in range(n)]
print(arr) # [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

# 2차원 리스트를 초기화할 때는 반드시 리스트 컴프리헨션 사용

n = 3
m = 4
arr = [m * [0]] * n
arr[1][1] = 5
print(arr) # [[0, 5, 0, 0], [0, 5, 0, 0], [0, 5, 0, 0]]
```

- List 관련 기타 함수
  - append() : O(1)
  - sort() : O(NlogN)
  - sort(reversre = True) : O(NlogN)
  - reverse() : O(N)
  - insert(index, value) : O(N)
  - count(value) : 리스트에서 value의 개수 : O(N)
  - remove(value) : O(N)

  - insert, remove는 O(N)이 걸리므로 insert는 append를 사용하고, remove는 다음과 같이 사용하는게 더 좋다.

```
arr = [1, 2, 3, 4, 5]
remove_set = {1, 2}
```

```
result = [i for i in arr if i not in remove_set]
# [3, 4, 5]
```

💡 문자열

- “”, ‘’ 를 사용

💡 튜플

- 리스트와 거의 비슷하지만 소괄호 사용
- 한 번 선언된 값 변경 불가

```
a = (1, 2, 3, 4)
a[1] = 2 # Error
```

- 그래프 알고리즘 구연할 때 자주 사용

💡 사전

- key, value 를 한 쌍으로 가진다.
- 기본적으로 해시 테이블을 이용하므로 검색 및 수정에 O(1)

```
data = dict()
data['a'] = 'A'
data['b'] = 'B'
data['c'] = 'C'

{ 'a': 'A', 'b': 'B', 'c': 'C' }
```

- 관련 함수 : keys(), values()

```
data = dict()
data['a'] = 'A'
data['b'] = 'B'
data['c'] = 'C'

{ 'a': 'A', 'b': 'B', 'c': 'C' }

key_list = data.keys()
value_list = data.values()

print(key_list) # dict_keys(['a', 'b', 'c'])
print(value_list) # dict_values(['A', 'B', 'C'])
```

집합 자료형

- 중복 X
- 순서 없음
- 특정 데이터가 이미 등장한 적 있는지 여부를 체크할 때 효과적

```
data = set([1, 2, 3, 4, 5])
data = { 1, 2, 3, 4, 5 }

data = set([1, 2, 3])
```

```
data.add(4) # {1, 2, 3, 4}
data.update([5, 6]) # {1, 2, 3, 4, 5, 6}
data.remove(3) # {1, 2, 4, 5, 6}
```