파이썬의 자료구조

파이썬의 기본 자료 구조 : 리스트(List), 튜플(Tuple), 사전(Dictionary), 집합(Set)

튜플 : 순서쌍

1차원의 고정된 크기를 가지는 변경 불가능한 순차 자료형. 튜플을 생성하는 가장 쉬운 방법은 쉼표로 구분된 값을 대입하는 것
괄호 ( )로 묶어서 표현할 수 있다.

In [1]:
```
ractangle = 100, 200 # 너비 100, 높이 200인 직사각형
ractangle
```

Out[1]: (100, 200)

In [2]:
```
ract_pos = 100, 100, 200, 100 # 100, 100의 위치에서 너비 200, 높이 100인 직사각형
ract_pos
```

Out[2]: (100, 100, 200, 100)

In [3]:
```
circle = 200, 200, 100 # 중심이 200, 200에 위치하며 반지름의 길이가 100인 원
circle
```

Out[3]: (200, 200, 100)

튜플의 원소의 값을 변경할 수 없다.

In [18]:
```
ract_pos[1] = 50
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-8ffd6314c55b> in <module>
----> 1 ract_pos[1] = 50

TypeError: 'tuple' object does not support item assignment
```

중첩된 튜플

In [5]:
```
ract_pos = (100, 100), (200, 100) # 100, 100의 위치에서 너비 200, 높이 100인 직사각형
ract_pos
```

Out[5]: ((100, 100), (200, 100))

In [6]:
```
circle = (200, 200), 100 # 중심이 200, 200에 위치하며 반지름의 길이가 100인 원
circle
```

Out[6]: ((200, 200), 100)

튜플에서 값 분리하기

In [10]:
```
pos, size = ract_pos
print('pos =', pos, 'size =', size)
```

```
pos = (100, 100) size = (200, 100)
```

In [11]:
```
center, radius = circle
center
```

Out[11]: (200, 200)

두 변수의 값 교환

In [13]:
```
a, b = 10, 20
```

In [14]:
```
a
```

Out[14]: 10

In [15]:
```
b
```

Out[15]: 20

In [16]:
```
b, a = a, b
b
```

Out[16]: 10

In [17]:
```
a
```

Out[17]: 20

리스트

튜플과 달리 리스트는 원소의 내용이나 리스트의 크기를 변경할 수 있다. 대괄호 []로 표현

In [26]:
```
shopping_list = ['딸기', '바나나', '우유', '달걀', '양배추', '당근', '소고기']
shopping_list
```

Out[26]: ['딸기', '바나나', '우유', '달걀', '양배추', '당근', '소고기']

In [27]:
```
shopping_list[4] = '상추'
shopping_list
```

Out[27]: ['딸기', '바나나', '우유', '달걀', '상추', '당근', '소고기']

In [28]:
```
shopping_list.append('마늘')
shopping_list
```

Out[28]: ['딸기', '바나나', '우유', '달걀', '상추', '당근', '소고기', '마늘']

In [29]:
```
shopping_list.insert(2, '오렌지')
shopping_list
```

Out[29]: ['딸기', '바나나', '오렌지', '우유', '달걀', '상추', '당근', '소고기', '마늘']

In [30]:
```
shopping_list.pop(1) #특정 위치의 값을 꺼내어 반환하고 해당 값을 리스트에서 삭제
```

Out[30]: '바나나'

In [31]:
```
shopping_list
```

Out[31]: ['딸기', '오렌지', '우유', '달걀', '상추', '당근', '소고기', '마늘']

In [32]:
```
shopping_list.remove('달걀') # 특정 원소 삭제
```

In [33]:
```
shopping_list
```

Out[33]: ['딸기', '오렌지', '우유', '상추', '당근', '소고기', '마늘']

리스트 안에 어떤 원소가 있는지 확인

In [34]:
```
'당근' in shopping_list
```

Out[34]: True

In [35]:
```
'바나나' in shopping_list
```

Out[35]: False

리스트 이어붙이기

In [36]:
```
color1 = ['red', 'green', 'blue']
color2 = ['white', 'black', 'cyan']
```

In [37]:
```
color1 = color1 + color2
color1
```

Out[37]: ['red', 'green', 'blue', 'white', 'black', 'cyan']

In [38]:
```
cities = ['서울', '부산', '대구']
cities.extend(['대전', '광주', '울산', '인천'])
cities
```

Out[38]: ['서울', '부산', '대구', '대전', '광주', '울산', '인천']

리스트 정렬

In [39]:
```
num = [9, 3, 7, 5, 1, 2, 8]
num
```

Out[39]: [9, 3, 7, 5, 1, 2, 8]

In [40]:
```
num.sort()
num
```

Out[40]: [1, 2, 3, 5, 7, 8, 9]

리스트 슬라이싱

In [41]:
```
cities
```

Out[41]: ['서울', '부산', '대구', '대전', '광주', '울산', '인천']

In [42]:
```
tour_list = cities[1:5]
tour_list
```

Out[42]: ['부산', '대구', '대전', '광주']

In [43]:
```
visited_list = cities[:4]
visited_list
```

Out[43]: ['서울', '부산', '대구', '대전']

In [44]:
```
to_go = cities[4:]
to_go
```

Out[44]: ['광주', '울산', '인천']

In [45]:
```
to_go = cities[-3:]
to_go
```

Out[45]: ['광주', '울산', '인천']

In [46]:
```
score = [94, 86, 92, 63, 88, 75, 52, 72, 98]
```

In [48]:
```
score[3:-2]=[65, 89, 78, 54]
score
```

Out[48]: [94, 86, 92, 65, 89, 78, 54, 72, 98]

리스트 함축

리스트 함축은 수학자들이 집합을 정의하는 것과 유사하다.
0부터 9까지의 자연수의 제곱의 값을 원소로 갖는 리스트를 다음과 같이 정의할 수 있다.

In [55]:
```
s = [x**2 for x in range(10)]
s
```

Out[55]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

리스트 [3, 4, 5]의 모든 원소에 2를 곱한 값을 원소로 갖는 새로운 리스트를 다음과 같이 정의할 수 있다.

In [56]:
```
list1 = [3,4,5]
list2 = [2*x for x in list1]
list2
```

Out[56]: [6, 8, 10]

조건이 붙는 리스트 함축 :
1부터 20 사이의 짝수를 원소로 갖는 리스트 정의

In [59]:
```
evenNum = [x for x in range(21) if x % 2 == 0]
evenNum
```

Out[59]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

다양한 자료형에 대한 리스트 함축

In [60]:
```
word_list= ['Every', 'bad', 'thing', 'has', 'its', 'end.']
item = [words[0] for words in word_list]
item
```

Out[60]: ['E', 'b', 't', 'h', 'i', 'e']

In [61]:
```
word_list = 'Every bad thing has its end.'.split()
word_len = [len(w) for w in word_list]
word_len
```

Out[61]: [5, 3, 5, 3, 3, 4]

리스트 연산

In [63]:
```
# 최댓값 찾기
score = [36, 78, 55, 49, 68, 64, 80, 93, 88, 80]
maxScore = score[0]
for x in range(1, len(score)):
    if score[x] > maxScore:
        maxScore = score[x]

maxScore
```

Out[63]: 93

In [65]:
```
# 조건을 만족하는 항목 모두 찾기
score = [36, 78, 55, 49, 68, 64, 80, 93, 88, 84]
excellent = []
for value in score:
    if value >= 80:
        excellent.append(value)

excellent
```

Out[65]: [80, 93, 88, 84]

내장 순차 자료형 함수

sorted : 오름차순 정렬된 새로운 순차 자료형을 반환

In [49]:
```
asc = sorted(score)
asc
```

Out[49]: [54, 65, 72, 78, 86, 89, 92, 94, 98]

집합

중복되지 않은 원들의 모임, 중괄호{}로 표현
세트 안의 원소들은 순서가 없기 때문에 인덱스를 가지고 세트의 원소에 접근할 수 없다.

In [66]:
```
numbers = {5, 2, 3, 5, 6, 8}
numbers
```

Out[66]:
```
{2, 3, 5, 6, 8}
```

In [67]:
```
num = numbers[2]
num
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-67-ec6cc96fdd2f> in <module>
----> 1 num = numbers[2]
      2 num

TypeError: 'set' object is not subscriptable
```

In [68]:
```
# 집합의 원소 추가
numbers.add(7)
numbers
```

Out[68]:
```
{2, 3, 5, 6, 7, 8}
```

In [69]:
```
# 집합의 원소 삭제
numbers.discard(6)
numbers
```

Out[69]:
```
{2, 3, 5, 7, 8}
```

In [70]:
```
# 집합을 공집합으로 만들기
numbers.clear()
numbers
```

Out[70]:
```
set()
```

In [71]:
```
len(numbers)
```

Out[71]:
```
0
```

In [72]:
```
char = set('It is my pleasure.')
char
```

Out[72]:
```
{' ', '.', 'I', 'a', 'e', 'i', 'l', 'm', 'p', 'r', 's', 't', 'u', 'y'}
```

집합의 연산 : 합집합, 교집합, 차집합 등

In [73]:
```
a = {5, 7, 3, 8, 9, 1}
b = {8, 4, 2, 3, 5, 6}
a | b
```

Out[73]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [74]:
```
a.union(b)
```

Out[74]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [75]:
```
b.union(a)
```

Out[75]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [76]:
```
a & b
```

Out[76]:
```
{3, 5, 8}
```

In [77]:
```
a.intersection(b)
```

Out[77]:
```
{3, 5, 8}
```

In [78]:
```
a - b
```

Out[78]:
```
{1, 7, 9}
```