11장 모음 자료형 1부: 문자열, 리스트, 튜플

주요 내용

- 기본 모음 자료형
- 순차형 모음 자료형(시퀀스): 문자열, 리스트, 튜플
- 순차 자료형 해체
- 유용한 함수: `enumerate()`, `zip()`

기본 모음 자료형

- 모음 자료형collection data types: 여러 개의 값을 항목으로 포함하는 값들의 자료형
- **컨테이너**container라고도 부름

- 파이썬이 제공하는 다섯 가지 기본 모음 자료형

```
문자열, 리스트, 튜플, 집합, 사전
```

스칼라 자료형

- 정수, 부동소수점, 불리언 등은 하나의 값으로만 구성된 자료형

모음 자료형 구분 1: 순서 존재 여부

- 순차 자료형: 문자열, 리스트, 튜플
- 비순차 자료형: 집합, 사전

모음 자료형 구분 2: 변경 허용 여부

- 가변mutable 자료형: 리스트, 집합, 사전
    - **시퀀스**sequences라고도 함
- 불변immutable 자료형: 문자열, 튜플

불변 자료형: 튜플

- 튜플은 불변 자료형이라 값을 수정할 수 없다.

In [1]:

```
a_tuple = (3, 15, 9, 12, 7, 14, 10)
a_tuple[0] = 100
```

```
---------------------------------------------------------------
---------------
TypeError                                 Traceback (most recent
call last)
~₩AppData₩Local₩Temp₩ipykernel_4612₩347940049.py in <module>
      1 a_tuple = (3, 15, 9, 12, 7, 14, 10)
----> 2 a_tuple[0] = 100

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

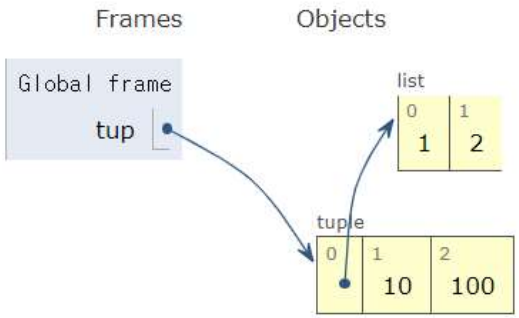
튜플 주의사항

- 아래 `tup` 이 가리키는 튜플의 첫번째 항목은 가변 자료형인 리스트 `[1, 2]` 임

In [2]:

```
tup = ([1, 2], 10, 100)
```

- 변수 `tup` 과 튜플 사이의 관계는 아래 그림이 참고



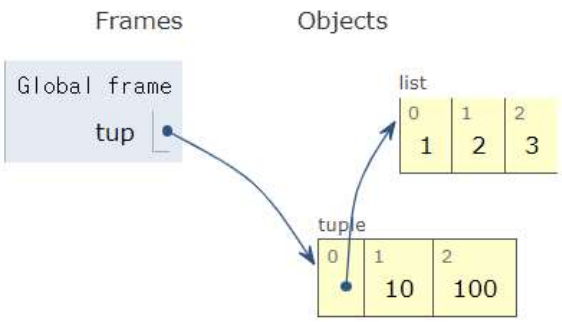
- 아래와 같이 첫번째 항목 자체는 변경이 가능

In [3]:

```
tup[0].append(3)
tup
```

Out[3]:

```
([1, 2, 3], 10, 100)
```



순차 자료형 해체

In [4]:
```
a, b = "12"
a
```

Out[4]:
```
'1'
```

In [5]:
```
c, d, e = [3, 4, 5]
e
```

Out[5]:
```
5
```

In [6]:
```
f, g = (6, 7)
g
```

Out[6]:
```
7
```

밑줄 기호 _ 활용

In [7]:

```
c, _, e = [3, 4, 5]
print(c + e)
```

```
8
```

주의사항

- 항목의 개수와 변수의 개수가 일치하지 않으면 오류가 발생

In [8]:

```
c, e = [3, 4, 5]
```

```
---------------------------------------------------------------
---------------
ValueError                                Traceback (most recent
call last)
~₩AppData₩Local₩Temp₩ipykernel_4612₩1486361304.py in <module>
----> 1 c, e = [3, 4, 5]

ValueError: too many values to unpack (expected 2)
```

별표 기호 `*` 활용

- 앞에 몇 개의 값에만 변수 할당에 사용하고 나머지는 하나의 리스트로 묶어서 퉁쳐 버릴 수 있음

In [9]:

```
values = (1, 2, 3, 4, 5)
a, b, *rest = values
rest
```

Out[9]:

```
[3, 4, 5]
```

- 나머지 항목들을 무시하고 싶다면, 별표와 밑줄을 함께 사용

In [10]:

```
a, b, *_ = values
a
```

Out[10]:

```
1
```

순차 자료형에 유용한 함수

`enumerate()` 함수

- 리스트의 항목과 인덱스를 쌍으로 갖는 모음 자료형의 객체 생성

In [11]:
```
some_list = ['foo', 'bar', 'baz', 'pyt', 'thon']
```

In [12]:
```
enumerate(some_list)
```

Out[12]:
```
<enumerate at 0x1d8b38c6c00>
```

In [13]:
```
list(enumerate(some_list))
```

Out[13]:
```
[(0, 'foo'), (1, 'bar'), (2, 'baz'), (3, 'pyt'), (4, 'thon')]
```

`enumerate()` 함수 활용 예제 1

- 짝수 인덱스의 항목만 추출

In [14]:

```
for i, v in enumerate(some_list):
    if i % 2 == 0:
        print(v)
```

```
foo
baz
thon
```

`enumerate()` 함수 활용 예제 2

- `(항목, 인덱스)` 형식의 튜플들의 리스트 생성

In [15]:

```
mapping = []

for i, v in enumerate(some_list):
    mapping.append((v, i))

mapping
```

Out[15]:

```
[('foo', 0), ('bar', 1), ('baz', 2), ('pyt', 3), ('thon', 4)]
```

`zip()` 함수

- 문자열, 리스트, 튜플 등 순차 자료형 여러 개를 묶어 하나의 값 생성
- 각 순차 자료형의 값에 사용된 순서를 그래도 반영

In [16]:
```
zip("abc", [1, 2, 3])
```

Out[16]:
```
<zip at 0x1d8b38c0180>
```

In [17]:
```
list(zip("abc", [1, 2, 3]))
```

Out[17]:
```
[('a', 1), ('b', 2), ('c', 3)]
```

- 여러 개의 모음 자료형을 짝짓는 것도 가능
- 길이가 다르면 가장 짧은 길이까지만 짝지음

In [18]:

```
list(zip("abcdefgh",(1, 2, 3, 4, 5), [5, 10, 15]))
```

Out[18]:

```
[('a', 1, 5), ('b', 2, 10), ('c', 3, 15)]
```

`zip()` 함수 활용 예제: 동시 반복

In [19]:

```
letters = ['a', 'b', 'c']
numbers = [0, 1, 2]
for l, n in zip(letters, numbers):
    print(f'문자: {l}')
    print(f'숫자: {n}')
```

```
문자: a
숫자: 0
문자: b
숫자: 1
문자: c
숫자: 2
```