

오늘의 강의 목표

- List에 대한 이해
- Tuple에 대한 이해
- Set에 대한 이해
- Dictionary에 대한 이해

List

- 아이 템(Item)들을 순차적으로 모아놓음
- '[]' 사이에 ','로 구분하여 아이 템 나열
- 각각의 아이 템은 서로 다른 data type 가능

```
>>> colors= ['red', 'green', 'blue']
>>> type(colors)
<class 'list'>
>>> print(colors)
['red', 'green', 'blue']
>>> l = ['가위', 1, '바위', 2, '보', 3]
>>> print(l)
['가위', 1, '바위', 2, '보', 3]
```

List Indexing

`list[i]`


list name

index number

```
>>> colors = ['red', 'green', 'blue']
>>> colors[0]
'red'
>>> colors[1]
'green'
>>> colors[2]
'blue'
>>> colors[2] = 'yellow'    # indexing & item update
>>> colors
['red', 'green', 'yellow']
```

List Slicing

`list[i:j]`



The diagram illustrates the list slicing syntax `list[i:j]`. Two red arrows point from the labels 'start' and 'end' to the characters 'i' and 'j' in the slice notation, respectively.

```
>>> elements = ['Al', 'B', 'C', 'Ca', 'Cu', 'Fe']
>>> elements[2:6]
['C', 'Ca', 'Cu', 'Fe']
>>> elements[:5]
['Al', 'B', 'C', 'Ca', 'Cu']
>>> elements[3:]
['Ca', 'Cu', 'Fe']
>>> elements[2:5] = ['Co', 'Cr', 'Cn'] # slice & update
>>> elements
['Al', 'B', 'Co', 'Cr', 'Cn', 'Fe']
```

Item Deletion

```
del list[i]
```

```
>>> fruits = ['apple', 'pineapple', 'strawberry', 'raspberry']
>>> del fruits[0]
>>> fruits
['pineapple', 'strawberry', 'raspberry']
>>> del fruits[2]
>>> fruits
['pineapple', 'strawberry']
```

len(), min() and max()

len(list)

```
>>> fruits = ['apple', 'pineapple', 'strawberry', 'raspberry']
>>> len(fruits)
4
>>> del fruits[0]
>>> len(fruits)
3
```

min(list) and max(list)

```
>>> scores = [10, 88, 37, 98, 54]
>>> min(scores)
10
>>> max(scores)
98
```

List Operators

`list1 + list2`

```
>>> list1 = ['a', 'b', 'c']
>>> list2 = ['d', 'e', 'f']
>>> list3 = list1 + list2
>>> list3
['a', 'b', 'c', 'd', 'e', 'f']
```

Concatenation

`list * n`

```
>>> list1 = ['a', 'b', 'c']
>>> list2 = list1 * 3
>>> list2
['a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c']
```

Repetition

Nested List

- List 역시 다른 list의 아이템이 될 수 있음

```
>>> lst = [1, 2, [3, 4]]
>>> lst[0]
1
>>> lst[1]
2
>>> lst[2]
[3, 4]
```

a list inside a list

List and For Loop

- List를 sequence로 한 for loop

```
>>> lst = ['a', 'b', 'c', 'd']
>>> for i in lst:
    print(i)
```

- List index를 이용한 for loop

```
>>> lst = ['a', 'b', 'c', 'd']
>>> for i in range(len(lst)):
    print(lst[i])
```

- Nested list의 for loop

```
>>> lst = [[1, 2], [3, 4], [5, 6]]
>>> for i, j in lst:
    print(i, j)
```

List Methods

- `list.append()`
- `list.extend()`
- `list.count()`
- `list.index()`
- `list.insert()`
- `list.clear()`
- `list.remove()`
- `list.reverse()`
- `list.sort()`
- `list.pop()`
- `list.copy()`

More at:

<https://docs.python.org/3.4/tutorial/datastructures.html>

append() and extend()

- append() : 맨 뒤에 아이템 추가
- extend() : 맨 뒤에 다른 list 붙이기

```
>>> lst = [1, 2, 3, 4]
>>> lst.append(5)
>>> lst
[1, 2, 3, 4, 5]
>>> lst.extend([6, 7])      # list를 extend
>>> lst
[1, 2, 3, 4, 5, 6, 7]
>>> lst.append([8, 9])      # list를 append
>>> lst
[1, 2, 3, 4, 5, 6, 7, [8, 9]]
```

의미 차이

count() and index()

- `count()` : 특정 값 아이템의 등장 회수 리턴
- `index()` : 특정 값 아이템의 첫 index 리턴

```
>>> lst = [1, 2, 3, 2, 2, 4, 5, 6, 7]
>>> lst.count(2)
3
>>> lst.count(7)
1
>>> lst.index(2)
1
>>> lst.index(7)
8
```

insert()

- insert() : 중간에 아이템 추가

```
>>> lst = ['one', 'two', 'four', 'five']
>>> lst.insert(0, 'zero')
>>> lst
['zero', 'one', 'two', 'four', 'five']
>>> lst.insert(2, 'three')
>>> lst
['zero', 'one', 'three', 'two', 'four', 'five']
```

reverse()

- reverse() : 순서 뒤집음

```
>>> lst = [1, 2, 3, 4, 5]
>>> lst.reverse()
>>> lst
[5, 4, 3, 2, 1]
```

clear() and remove()

- clear() : 모든 아이템 삭제
- remove() : 특정 값 첫 아이템 삭제

```
>>> lst = [1, 2, 3, 4, 5]
>>> lst.clear()
>>> lst
[]
```

```
>>> lst = [1, 2, 3, 2, 5, 7, 2]
>>> lst.remove(2)
>>> lst
[1, 3, 2, 5, 7, 2]
```

pop()

- pop() : 맨 뒤 아이템 꺼내고 삭제

```
>>> lst = [1, 2, 3, 4, 5]
```

```
>>> lst.pop()
```

```
5
```

```
>>> lst
```

```
[1, 2, 3, 4]
```

```
>>> lst.pop(2)
```

```
3
```

```
>>> lst
```

```
[1, 2, 4]
```

맨 뒤 아이템 꺼내고 삭제

2번 index의 아이템 꺼내고 삭제

sort()

- `sort()` : 오름차순 정렬

```
>>> lst = [3, 5, 1, 9, 6]
>>> lst.sort()
>>> lst
[1, 3, 5, 6, 9]
```

- 내림차순 정렬은?

```
>>> lst = [3, 5, 1, 9, 6]
>>> lst.sort(reverse = True)
>>> lst
[9, 6, 5, 3, 1]
>>>
```

참고: sorted() function

```
>>> lst = [3, 5, 1, 9, 6]
>>> sorted(lst)
[1, 3, 5, 6, 9]
>>> lst          # 원본 lst는 불변
[3, 5, 1, 9, 6]
>>> sorted(lst, reverse = True) # 내림차순
[9, 6, 5, 3, 1]
>>> lst
[3, 5, 1, 9, 6]      # 원본 lst는 역시 불변
```

List method 아님. Python built-in function임.

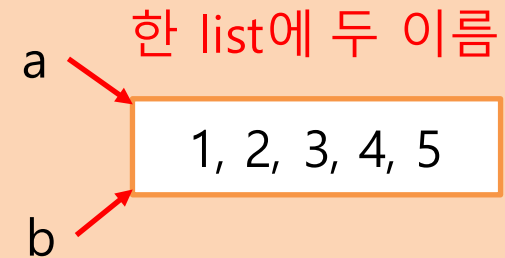
More at:

<https://docs.python.org/3.4/howto/sorting.html>

copy()

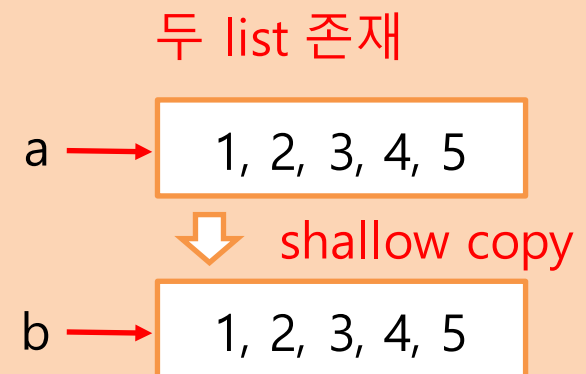
- List deep copy

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a      # deep copy
>>> a.append(6)
>>> a
[1, 2, 3, 4, 5, 6]
>>> b
[1, 2, 3, 4, 5, 6]
```



- copy() : list shallow copy

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a.copy()  # shallow copy
>>> a.append(6)
>>> a
[1, 2, 3, 4, 5, 6]
>>> b
[1, 2, 3, 4, 5]
```



Tuple

- 변경 불가능한 list
- '()' 사이에 ','로 구분하여 아이템 나열
- List 대비 장점
 - Faster
 - Safer (Write-protected data)

```
>>> rgb = ('red', 'green', 'blue')
>>> type(rgb)
<class 'tuple'>
>>> rgb.append('yellow')
Traceback (most recent call last):
  File "<pyshell#281>", line 1, in <module>
    rgb.append('yellow')
AttributeError: 'tuple' object has no attribute 'append'
```

Type Conversion between List and Tuple

- `list()`

```
>>> tpl = (1, 2, 3, 4)
>>> type(tpl)
<class 'tuple'>
>>> lst = list(tpl)
>>> type(lst)
<class 'list'>
>>> lst
[1, 2, 3, 4]
```

- `tuple()`

```
>>> lst = [1, 2, 3, 4]
>>> type(lst)
<class 'list'>
>>> tpl = tuple(lst)
>>> type(tpl)
<class 'tuple'>
>>> tpl
(1, 2, 3, 4)
```

Set

- 집합
- '{}' 사이에 ','로 구분하여 아이템 나열
- 아이템간 순서 의미 없음

```
>>> hbo = {'Band of Brothers', 'Rome', 'Game of Thrones'}
```

- 같은 아이템이 여러 개 있는 경우?

```
>>> s = {1, 1, 2, 2, 3, 4}
>>> s
{1, 2, 3, 4} # 중복 아이템들은 하나로
```

Set

- 아이템 포함 여부 확인 : in and not in

```
>>> hbo = {'Band of Brothers', 'Rome', 'Game of Thrones'}
>>> fox = {'24', 'House', 'Prison Break'}
>>> '24' in hbo
False
>>> '24' not in hbo
True
>>> 'Rome' in hbo
True
```

- For loop

```
>>> hbo = {'Band of Brothers', 'Rome', 'Game of Thrones'}
>>> for i in hbo:
    print(i)
```

Set Operators

- $s1 \mid s2$: 합집합
- $s1 \& s2$: 교집합
- $s1 - s2$: 차집합
- $s1 \wedge s2$: 대칭 차집합

```
>>> s1 = {1, 2, 3, 4, 5}
>>> s2 = {4, 5, 6, 7, 8}
>>> s1 | s2
{1, 2, 3, 4, 5, 6, 7, 8}
>>> s1 & s2
{4, 5}
>>> s1 - s2
{1, 2, 3}
>>> s1 ^ s2
{1, 2, 3, 6, 7, 8}
```


Dictionary

- Key, value pair의 set
- Key와 value는 ':'로 연결
- Key를 index로 하여 접근

```
>>> dic = {'apple': '사과', 'pear': '배', 'watermelon': '수박'}
>>> dic
{'watermelon': '수박', 'pear': '배', 'apple': '사과'}
>>> dic['watermelon']          # key를 index로 접근
'수박'
>>> dic['apple'] = '풋사과'    # 특정 key의 value 변경
>>> dic
{'watermelon': '수박', 'pear': '배', 'apple': '풋사과'}
>>> dic['strawberry'] = '딸기'  # 새로운 pair 추가
>>> dic
{'watermelon': '수박', 'pear': '배', 'apple': '풋사과',
'strawberry': '딸기'}
```

Dictionary

- Tuple은 key로 사용 가능하나 list는 불가능

```
>>> dic = {(1, 1): 'a', (2, 2): 'b', (3, 3): 'c'}
>>> dic = {[1, 1]: 'a', [2, 2]: 'b', [3, 3]: 'c'}
Traceback (most recent call last):
  File "<pyshell#220>", line 1, in <module>
    dic = {[1, 1]: 'a', [2, 2]: 'b', [3, 3]: 'c'}
TypeError: unhashable type: 'list'
```

Dictionary Operators

- `len(d)` : key, value pair의 개수
- `del(d[k])` : d에서 key k 삭제
- `k in d` : Key k가 있는가?
- `k not in d` : Key k가 없는가?

Dictionary and For Loop

- items() method : (key, value) tuple의 list 리턴

```
>>> dic = {'a': 'A', 'b': 'B', 'c': 'C'}
>>> dic.items()
dict_items([('b', 'B'), ('c', 'C'), ('a', 'A')])
>>> for key, value in dic.items():
    print(key, value)
```

```
b B
c C
a A
```

} 일종의 set이기 때문에 순서가 무의미

```
>>> for key, value in sorted(dic.items()):
    print(key, value)
```

Key 기준으로 sorting

```
a A
b B
c C
```

Dictionary and For Loop

- `keys()` method : key들로 이루어진 sequence 리턴

```
>>> dic = {'a': 'A', 'b': 'B', 'c': 'C'}
>>> dic.keys()
dict_keys(['b', 'c', 'a'])
>>> for key in dic.keys():
    print(key)
```

- `values()` method : value들로 이루어진 sequence 리턴

```
>>> dic = {'a': 'A', 'b': 'B', 'c': 'C'}
>>> dic.values()
dict_values(['B', 'C', 'A'])
>>> for value in dic.values():
    print(value)
```

get() method

- `get()` : [] 사용과 유사하나 key가 없을 경우 에러 대신 `None` 혹은 `default value` 리턴

```
>>> dic = {'a': 'A', 'b': 'B', 'c': 'C'}
>>> dic['c']
'C'
>>> dic['d']
Traceback (most recent call last):
  File "<pyshell#274>", line 1, in <module>
    dic['d']
KeyError: 'd'
>>> print(dic.get('d'))
None
>>> print(dic.get('d', '!'))
!
```

default value

Advanced Topics

- 2차원, 3차원 자료구조

Questions

