List and Tuples

Video 1

List and Tuples are compound data types

Tuples

ordered sequence
written as comma-seperated elements within parantheses.

Ratings = (10, 9, 6, 5, 10, 8, 9, 6, 2)

tuples1 = ('disco', 10, 1.2)
type (tuples1) : tuple

tuples1 [0] = 'disco'
[1] = 10
[2] = 1.2
[-1] = 1.2
[-2] = 10
[-3] = 'disco'

tuples2 = tuples1 + ('hard rock', 10)
gives : ('disko', 10, 1.2, 'hard rock', 10)

tuples2 [0:3] : ('disco', 10, 1.2)
tuples2 [3:5] : ('hard rock', 10)
len (tuples2) : 5

Tuples : Immutable

Ratings = (10, 5, 6)
Ratings[1] = 4 ✗
Ratings = (2, 3, 4) ✓

⊛ manipulating tuples

Ratings = (10, 5, 6)
RatingSorted = sorted(Ratings) : (5, 10, 6)

Ratings.index(5) : 1

Tuples : Nesting.

NT = (1, 2, ('pop', 'rock', (3, 4)))

NT[2] : ('pop', 'rock', (3, 4))
NT[2][0] : 'pop'
NT[2][0][2] : p

List [ ]

⊛ ordered sequences

L = ['Michael Jackson', 10.1, 1982]

List are mutable

tuple

L = ['Michael', 10.1, [1,2], ('A', 1)]

L[0] : Michael
L[-1] : ('A', 1)
L[1:3] : (10.1, [1,2])

⊛ Index conversions are identical for lists and tuples

Combine

L1 = L + ['pop', 10]

gives [Micheal, 10.1, [1,2], ('A', 1), ('Pop', 10)]

List are mutable

L.extend(['pop', 10])

gives [Micheal, 10.1, (1,2), ('A', 1), 'Pop', 10]

L.append(['pop', 10])

gives [Micheal, 10.1, (1,2), ('A', 1), ['Pop', 10]]

A = ['disco', 10.1, 2]

A[0] = 'hard rock' ✓

new A : ['hard rock', 10.1, 2]

del (A[0]) → A = [10.1, 2]

'hard rock'.split() : ['hard', 'rock']

Seperate at each space

'A, B, C, D'.split(',') : ['A', 'B', 'C', 'D']

List : Aliasing

A = ['hard rock', 10, 1.2]

B = A

List

A = ['hard rock', 10, 1.2]

B //

Same list for both A and B

↓

This is Aliasing

if we set A[0] = "banana"

we get side effect B[0] : 'banana'

List: Clone

A = ['hard rock', 10, 1.2]

B = A[:]

A = ['hard rock', 10, 1.2]

B = ['hard rock', 10, 1.2]

A = ['hard rock', 10, 1.2]

help(A)

Dictionaries

Video 1

index by label

similar to element

List

index	Element
0	Element 1
1	Element 2
2	⋮
3	⋮
⋮	⋮

Dictionary

key	Value
key 1	Value 1
key 2	Value 2
key 3	value 3

Dictionaries are denoted by curly brackets {}
The keys have to immutable and unique
The values can be immutable, mutable and duplicates

ex Dic = {'key1': 1, 'key2': '2', 'key3': [3,3,3],
'key4': (4,4), ('key5'): 5}

Dic['key2'] : '2'

Dic['key6'] = 'Six' ads new value to Dic
del (Dic['key2'] delete value in Dic
'key4' in Dic : True
'key10' in Dic : False

not on the lecture

Dic.keys() : dict_keys(['key1', 'key2', 'key3', ..., 'key6'])

Dic.values() : dict_values([1, '2', [3,3,3], (4,4), 5 ...])

key can also be something like (0,1)

nata tuples

Sets

Sets are type of collections
Unlike lists and Tuples they are unordered.
do not record element position
Sets only have unique elements

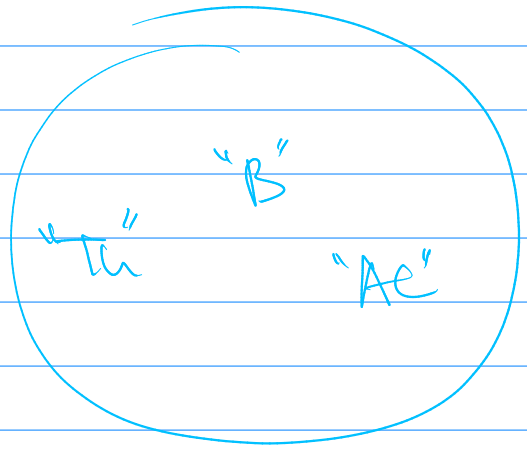
Set1 = {"pop", "rock", "soul", "rock"}

Set1 : {"rock", "pop", "soul"}

List ⟶ Set

```
album_list = ["MJ", "Th", "Th", 1982]
album_set = set(album_list)
album_set: {"MJ", "Th", 1982}
```

Set Operations



A = {"Th", "B", "Ac"}

A.add("NSY")

A: {"Th", "B", "Ac", "NSY"}

A.remove("NSY")

A: {"Th", "B", "Ac"}

"AC" in A : True
"Who" in A : False

album_set_1 ={"AC/DC"," Back in Black ","Thriller }

album_set_2={"AC/DC", "Back in Black","The Dark Side of the Moon"}

album_set_3=album_set_1 & album_set_2

= album_set_1 . intersection(album_set_2)

intersection ∩

Union ∪ ? { } + { } ?

{ } | { }

union

union = album_set_1 . union(album_set_2)

not necessary

set(A) . issubset(B) : True

set(A) . issuperset(B) : False.

A.issubset(B) ✓