

Structured Types 2

By 윤명근 / 박수현

수업목표

- Tuple
- Dictionary
- Set

Tuple

- **Immutable** data type
 - 값을 한번 정하면 추가, 수정, 삭제 불가
- **Sequential(순차적)** data type : index로 접근 (**read-only**)

4.2example1.py

```
File Edit Format Run Options Window Help
1 score = (80, 70, 90, 60)
2
3 print("\n1) Tuple score =", score, "\n")
4
5 i = 0
6 while i < len(score):
7     print(">> score[{idx}] = {sc}" .format(idx = i, sc = score[i]))
8     i = i + 1
9
10 #error
11 print("\n3) Tuple의 요소값 변경불가")
12 score[1] = 100
13
```

1) Tuple score = (80, 70, 90, 60)

```
>> score[0] = 80
>> score[1] = 70
>> score[2] = 90
>> score[3] = 60
```

3) Tuple의 요소값 변경불가

Traceback (most recent call last):

File "C:\W소사\강의예제\4.2example1.py", line 12, in <module>
score[1] = 100

TypeError: 'tuple' object does not support item assignment

Tuple

```
tuple_1.py
File Edit Format Run Options Window Help
1 student = (20250001, "김소프", 23)
2
3 print("\n1) Tuple student 값 :", student)
4 print("2) id(student) = ", id(student))
5
6 hakbun, name, age = student
7
8 print("\n3) hakbun =", hakbun)
9 print("    name =", name)
10 print("    age =", age)
11
12 # tuple 값 변경
13 hakbun = 20600002
14 name = "김کم공"
15 age = 20
16
17 student = (hakbun, name, age)
18 print("\n4) 변경된 Tuple student 값 :", student)
19 print("5) id(student) = ", id(student))
20 print("    새로운 Tuple student가 생성된 것임")
21
22 print("\n6) 변경 후 Tuple student = ", student, "\n")
23
24 # 갯수가 불일치
25 hakbun, name, age, id_2 = student
26
```

• Tuple 대입 연산

- tuple 에서 여러 개의 변수로 한번에 값을 대입하는 기능

```
1) Tuple student 값 : (20250001, '김소프', 23)
2) id(student) = 2122962028672

3) hakbun = 20250001
   name = 김소프
   age = 23

4) 변경된 Tuple student 값 : (20600002, '김کم공', 20)
5) id(student) = 2122962330816
   새로운 Tuple student가 생성된 것임

6) 변경 후 Tuple student = (20600002, '김کم공', 20)
```

```
Traceback (most recent call last):
  File "C:\소사\강의예제\tuple_1.py", line 25, in <module>
    hakbun, name, age, id_2 = student
ValueError: not enough values to unpack (expected 4, got 3)
```

```
tuple_2.py
File Edit Format Run Options Window Help
1 # Tuple 생성
2 '''
3 student = tuple()
4 print("Wn1) Tuple student 값 :", student)
5 print("2) id(student) :", id(student))
6 '''
7 student = (20250001, "김소프", 23)
8
9 print("Wn1) Tuple student 값 :", student)
10 print("2) id(student) :", id(student))
11
12 # tuple 값 변경
13 list_student = list(student)
14
15 list_student[0] = 20600002
16 list_student[1] = "김컴공"
17 list_student[2] = 20
18 print("Wn3) list_student :", list_student)
19
20 student = tuple(list_student)
21 print("Wn4) 변경된 Tuple student 값 :", student)
22 print("5) id(student) :", id(student))
23 print(" 새로운 Tuple student가 생성된 것임")
24
25 print("Wn6) 변경 후 Tuple student :", student, "Wn")
26
27 # slicing tuple
28 stu_name = ("park", "lee", "kim", "yeom", "hwang")
29 print("Wn7) stu_name[2] :", stu_name[2])
30 print("7-1) type(stu_name[2]) :", type(stu_name[2]))
31 print("Wn8) stu_name[2:4] :", stu_name[2:4])
32 print("8-1) type(stu_name[2:4]) :", type(stu_name[2:4]))
33 print("Wn9) stu_name[3:] :", stu_name[3:])
34
35 # merging tuple
36 merged_tuple = student + stu_name
37 print("Wn10) merged_tuple :", merged_tuple)
38
```

• 기타 Tuple 연산

- tuple 생성
- list로 type casting 후 tuple 값 변환
- tuple slicing
- 서로 다른 tuple들 merge : +

1) Tuple student 값 : (20250001, '김소프', 23)

2) id(student) : 2729087974912

3) list_student : [20600002, '김컴공', 20]

4) 변경된 Tuple student 값 : (20600002, '김컴공', 20)

5) id(student) : 2729093522304
새로운 Tuple student가 생성된 것임

6) 변경 후 Tuple student : (20600002, '김컴공', 20)

7) stu_name[2] : kim
7-1) type(stu_name[2]) : <class 'str'>

8) stu_name[2:4] : ('kim', 'yeom')

8-1) type(stu_name[2:4]) : <class 'tuple'>

9) stu_name[3:] : ('yeom', 'hwang')

10) merged_tuple : (20600002, '김컴공', 20, 'park', 'lee', 'kim', 'yeom', 'hwang')

Dictionary

- 사전
 - 키(key)와 값(value)으로 구성된 **집합** (non-sequential data type)
 - 유일한 key로 검색
 - **비순차형**이며 **Index 접근 불가** (ex: friends[2] → 불가)
 - Programming language 별로 dictionary, map, table 등 다양한 이름 존재



Dictionary

- 사전 입력 방법
 - 사전이름 = { 키1:값1, 키2:값2 }
 - 사전이름[키] = 값
 - 사전이름.update({키3:값3, 키4:값4})

```
*4.2example2.py
File Edit Format Run Options Window Help
1 # dictionary 생성
2 dic = {1: "Alice", 2: "Bob", "KMU": "Kookmin"}
3 print("1) dic =", dic)
4
5 # dictionary item 추가 1
6 dic[3] = "Carole"
7 print("\n2) 3:W'CaroleW' 추가")
8 print("    dic =", dic)
9
10 # dictionary item 추가 2
11 dic.update({4.0 : 200, 1.2: 3.14})
12 print("\n3) 4.0 : 200, 1.2 : 3.14 추가")
13 print("    dic =", dic)
14
15 print("\n4) dic[W'KMUW'] 값 :", dic["KMU"])
16 print("5) dic[2] 값 :", dic[2])
17 print("6) dic[4.0] 값 :", dic[4.0])
18 print("7) dic[1.2] 값 :", dic[1.2])
19
20 # dictionary value 변경
21 print("\n8) dic[1] 값을 W'ParkW'으로 변경")
22 dic[1] = "Park"
23 print("    dic =", dic)
24
25 print("\n9) dic[1.2] 값을 W'PhiW'로 변경")
26 dic[1.2] = "Phi"
27 print("    dic =", dic)
28
29 # 존재하지 않는 index의 값
30 print(dic["Alice"])
31
32
```

Dictionary 입력, 변경 및 조회

```
1) dic = {1: 'Alice', 2: 'Bob', 'KMU': 'Kookmin'}
2) 3: 'Carole' 추가
   dic = {1: 'Alice', 2: 'Bob', 'KMU': 'Kookmin', 3: 'Carole'}
3) 4.0 : 200, 1.2 : 3.14 추가
   dic = {1: 'Alice', 2: 'Bob', 'KMU': 'Kookmin', 3: 'Carole', 4.0: 200, 1.2: 3.14}
4) dic["KMU"] 값 : Kookmin
5) dic[2] 값 : Bob
6) dic[4.0] 값 : 200
7) dic[1.2] 값 : 3.14
8) dic[1] 값을 "Park"으로 변경
   dic = {1: 'Park', 2: 'Bob', 'KMU': 'Kookmin', 3: 'Carole', 4.0: 200, 1.2: 3.14}
9) dic[1.2] 값을 "Phi"로 변경
   dic = {1: 'Park', 2: 'Bob', 'KMU': 'Kookmin', 3: 'Carole', 4.0: 200, 1.2: 'Phi'}
Traceback (most recent call last):
  File "C:\소사\강의예제\4.2example2.py", line 30, in <module>
    print(dic["Alice"])
  KeyError: 'Alice'
```


Dictionary

- 사전 전체 검색 방법
 - 없는 키 접근 → 에러 발생

```
dic = {2022123: "Alice", 2022200: "Bob", "KMU": "JeongReung"}  
for key in dic:  
    print(key, "'s value is ", dic[key])  
  
print(dic[2022100])
```

2022123 's value is Alice

2022200 's value is Bob

KMU 's value is JeongReung

Traceback (most recent call last):

File "C:/Users/mkyoon/PycharmProjects/hello_world/hello.py", line 8, in <module>

print(dic[2022100])

KeyError: 2022100

Dictionary

- 사전 삭제
 - `del <사전>[<키>]` : <키>를 <사전>에서 찾은 다음 삭제 한다.
 - `<사전>.pop(<키>)` : <키>를 <사전>에서 찾은 항목을 반환한 다음 삭제

```

1 student = { "name" : "김국민", "id" : "2050111", "성적" : [91, 85, 99], "기타" : "NULL" }
2
3 print("1) student = ", student)
4
5 print("\n2) 학과 추가")
6 student["학과"] = "소프트웨어"
7
8 print("3) student :", student)
9
10 print("\n4) 장학금 추가")
11 student["장학금"] = 2000000
12 print("5) student :", student)
13
14 # Dictionary 요소 삭제. "기타" 삭제
15 print("\n6) del 이용 W\"기타\" 삭제")
16 del student["기타"]
17 print("\n7) student :", student)
18
19 # pop()을 이용하여 Dictionary에서 "장학금" 요소 삭제
20 print("\n8) pop() 이용 W\"장학금\" 삭제")
21 scholarship = student.pop("장학금")
22 print("9) scholarship =", scholarship)
23 print("10) student :", student)
24
25 # student.keys() : student의 Key만을 모아서 dict_keys 객체를 return
26 print("\n11) student.keys() :", student.keys())
27
28 # student.values() : student의 Values만을 모아서 dict_values 객체를 return
29 print("\n12) student.values() :", student.values())
30
31 # student.items() : student의 Key와 Value의 쌍을 tuple로 묶은 값을
32 # dict_items 객체로 return
33 print("\n13) student.items() :", student.items())
34
35 # Key: Value 쌍 모두 지우기(clear)
36 student.clear()
37 print("\n14) student :", student)
38

```

Dictionary 삭제

```

1) student = {'name': '김국민', 'id': '2050111', '성적': [91, 85, 99], '기타': 'NULL'}
2) 학과 추가
3) student : {'name': '김국민', 'id': '2050111', '성적': [91, 85, 99], '기타': 'NULL', '학과': '소프트웨어'}
4) 장학금 추가
5) student : {'name': '김국민', 'id': '2050111', '성적': [91, 85, 99], '기타': 'NULL', '학과': '소프트웨어', '장학금': 2000000}
6) del 이용 "기타" 삭제
7) student : {'name': '김국민', 'id': '2050111', '성적': [91, 85, 99], '학과': '소프트웨어', '장학금': 2000000}
8) pop() 이용 "장학금" 삭제
9) scholarship = 2000000
10) student : {'name': '김국민', 'id': '2050111', '성적': [91, 85, 99], '학과': '소프트웨어'}
11) student.keys() : dict_keys(['name', 'id', '성적', '학과'])
12) student.values() : dict_values(['김국민', '2050111', [91, 85, 99], '소프트웨어'])
13) student.items() : dict_items([('name', '김국민'), ('id', '2050111'), ('성적', [91, 85, 99]), ('학과', '소프트웨어')])
14) student : {}

```

```

1 dic = {2050123: "김국민", 2050200: "이국민", "주소": "성북구 정릉로"}
2
3 print("1) dic = ", dic)
4
5 print("\n2) for문을 이용한 key, value 출력")
6
7 for key in dic:
8     print(">> key :", key, ", value :", dic[key])
9
10 print("\n3)", dic.items())
11
12 print("\n4) for문을 이용한 dic.items()에서의 key, value 출력")
13
14 for key, value in dic.items():
15     print(">> key :", key, ", value :", value)
16
17 dic_list = list(dic.items())
18
19 print("\n5) dic_list =", dic_list)
20
21 print("\n6) dir(dic)")
22 print(dir(dic))
23

```

Dictionary - 사전 전체 검색 방법

1) dic = {2050123: '김국민', 2050200: '이국민', '주소': '성북구 정릉로'}

2) for문을 이용한 key, value 출력
 >> key : 2050123 , value : 김국민
 >> key : 2050200 , value : 이국민
 >> key : 주소 , value : 성북구 정릉로

3) dict_items([(2050123, '김국민'), (2050200, '이국민'), ('주소', '성북구 정릉로')])

4) for문을 이용한 dic.items()에서의 key, value 출력
 >> key : 2050123 , value : 김국민
 >> key : 2050200 , value : 이국민
 >> key : 주소 , value : 성북구 정릉로

5) dic_list = [(2050123, '김국민'), (2050200, '이국민'), ('주소', '성북구 정릉로')]

6) dir(dic)
 ['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']

Dictionary

- 키 존재 확인 - 2가지 방법 (1/2)
 - 방법 1 : `<키> in <사전>`
`<키>` 가 `<사전>`에 있으면 True, 없으면 False

```
checkup_key_existance1.py
File Edit Format Run Options Window Help
1 dic = {2050123: "Alice", 2050200: "Bob", "KMU": "JeongReung"}
2
3 for key in dic:
4     print(">> key =", key, ", value =", dic[key])
5
6 print("\n>> Find key existence.")
7
8 if 2025100 in dic :
9     print(">> 존재하는 key 입니다.")
10    print(dic[2017100])
11 else:
12    print(">> 2025100은 존재하지 않는 Key 입니다.")
```

```
>> key = 2050123 , value = Alice
>> key = 2050200 , value = Bob
>> key = KMU , value = JeongReung

>> Find key existence.
>> 2025100은 존재하지 않는 Key 입니다.
...
```

Dictionary

- 키 존재 확인 - 2가지 방법 (2/2)
 - 방법 2 : <사전>.get(<키>)
<키> 가 <사전>에 있으면 <키>에 해당하는 <값>을 리턴,
없으면 None 을 리턴

```
File Edit Format Run Options Window Help
1 dic = {2050123: "Alice", 2050200: "Bob", "KMU": "JeongReung"}
2
3 for key in dic:
4     print(">> key =", key, ", value =", dic[key])
5
6 id = dic.get(2030100)
7 print("\n>> ID:", id)
8
9 # get(dic, 'default 값') : Dictionary 내에서 찾으려는 Key 값이 없을 경우,
10 # 미리 정해 둔 default값을 대신 가져오고자 할 때 사용
11 id = dic.get(2030100, "김국민")
12 print("\n>> ID:", id)
13
```

```
>> key = 2050123 , value = Alice
>> key = 2050200 , value = Bob
>> key = KMU , value = JeongReung

>> ID: None

>> ID: 김국민
```

Set

- 집합
 - 유일한 값(unique value)들로 구성된 비순차적(non-sequential) 자료구조
 - 수학에서의 집합 구현
 - 집합이름 = {값1, 값2,...}
 - 집합연산 지원
 - 합집합 연산: |
 - 교집합 연산: &
 - 차집합 연산: -
 - 대칭 차집합 연산: ^

Set

- 집합



4.2example4.py

File Edit Format Run Options Window Help

```
1 A = {'a','b','c',1,2}
2 B = {'a','c','d',3,4}
3
4 print(">> A =",A)
5 print(">> B =",B)
6
7 print("\n>> A union B =", A|B)
8 print(">> A intersect B =", A&B)
9 print(">> A minus B =", A-B)
10
```

```
>> A = {1, 2, 'b', 'a', 'c'}
>> B = {3, 4, 'c', 'a', 'd'}
```

```
>> A union B = {1, 2, 3, 4, 'b', 'a', 'd', 'c'}
>> A intersect B = {'c', 'a'}
>> A minus B = {1, 2, 'b'}
```

```
>>>
>>>
```

```
===== RESTART: C:\W과소,
=====
```

```
>> A = {'c', 1, 2, 'b', 'a'}
>> B = {'c', 'd', 3, 4, 'a'}
```

```
>> A union B = {'c', 1, 2, 'b', 'd', 3, 4, 'a'}
>> A intersect B = {'c', 'a'}
>> A minus B = {1, 2, 'b'}
```

```
>>>
>>>
```

```
===== RESTART: C:\W과소,
=====
```

```
>> A = {1, 2, 'c', 'a', 'b'}
>> B = {'c', 3, 4, 'd', 'a'}
```

```
>> A union B = {1, 2, 3, 4, 'd', 'b', 'c', 'a'}
>> A intersect B = {'c', 'a'}
>> A minus B = {1, 2, 'b'}
```


Set

- 집합
- 원소 추가
 - <집합>.add(<원소>) : 집합에 <원소>를 넣는다.
 - <집합>.update([<원소>, <원소>, <원소>, ...]) : 집합에 리스트에 있는 <원소>들을 넣는다.

set_add_update.py

File Edit Format Run Options Window Help

```
1 prime_numbers = { 2, 3, 5, 7 }
2
3 print("1) 추가 하기 전 :", prime_numbers)
4
5 # 원소 한개를 추가 할때는 add를 사용
6 prime_numbers.add(11)
7 print("2) 원소 W'11W' 추가후 :", prime_numbers)
8
9 # 여러 개의 원소를 추가 할 때는 update를 사용. list 활용
10 prime_numbers.update([13, 17, 19])
11 print("3) 여러 개 원소 [13, 17, 19] 추가후 :", prime_numbers)
12
13 # 기존에 존재하는 원소 다시 추가
14 prime_numbers.add(3)
15 print("4) 이미 존재하는 원소 W'3W' 추가후 :", prime_numbers)
16
17 prime_numbers.update([13, 17, 19])
18 print("5) 이미 존재하는 원소들 [13, 17, 19] 추가후 :", prime_numbers)
19
```

1) 추가 하기 전 : {2, 3, 5, 7}

2) 원소 "11" 추가후 : {2, 3, 5, 7, 11}

3) 여러 개 원소 [13, 17, 19] 추가후 : {2, 3, 5, 7, 11, 13, 17, 19}


4) 이미 존재하는 원소 "3" 추가후 : {2, 3, 5, 7, 11, 13, 17, 19}

5) 이미 존재하는 원소들 [13, 17, 19] 추가후 : {2, 3, 5, 7, 11, 13, 17, 19}

^^^

Set

- 집합
- 원소 제거
 - <집합>.discard(<원소>) : <원소>가 집합에 **있으면 삭제**
 - <집합>.remove(<원소>) : <원소>가 집합에 **있으면 삭제하고, 없으면 에러를 발생**

 set_delete.py

File Edit Format Run Options Window Help

```
1 prime_numbers = { 1, 2, 3, 4 }
2 print("1) 원소 삭제 전 :", prime_numbers)
3
4 prime_numbers.discard(3)
5 print("2) W'3W" 삭제(discard 사용): ")
6 print("3) 삭제 후 :", prime_numbers)
7
8 print("Wn4) 현존하지 않는 원소 W'5W" 삭제시도(discard 사용)")
9 prime_numbers.discard(5)
10 print("5) W'5W" 삭제 시도 후(discard 사용) 내용 변화 없음 :", prime_numbers)
11
12 print("Wn6) 현존하지 않는 원소 W'5W" 삭제시도(remove사용). error 발생")
13 prime_numbers.remove(5)
14
```

```
1) 원소 삭제 전 : {1, 2, 3, 4}
2) "3" 삭제(discard 사용):
3) 삭제 후 : {1, 2, 4}

4) 현존하지 않는 원소 "5" 삭제시도(discard 사용)
5) "5" 삭제 시도 후(discard 사용) 내용 변화 없음 : {1, 2, 4}

6) 현존하지 않는 원소 "5" 삭제시도(remove사용). error 발생
Traceback (most recent call last):
  File "C:\W소사W강의예제Wset_delete.py", line 13, in <module>
    prime_numbers.remove(5)
KeyError: 5
```

숙제

- 그리스 문자를 “키=영문이름”, “값=한글이름” 으로 사전을 구현하시오.
예) key = “Alpha”, value = “알파”
- Input 명령어로 키를 입력받아 키에 해당하는 값을 출력하시오.

이름	그리스 문자			이름	그리스 문자	
	소문자	대문자			소문자	대문자
알파 (Alpha)	α	A		뉴 (Nu)	ν	N
베타 (Beta)	β	B		크사이 (Xi)	ξ	Ξ
감마 (Gamma)	γ	Γ		오미크론 (Omicron)	\omicron	Ο
델타 (Delta)	δ	Δ		파이 (Pi)	π	Π
엡실론 (Epdilon)	ϵ	Ε		로 (Rho)	ρ	Ρ
제타 (Zeta)	ζ	Z		시그마 (Sigma)	σ	Σ
에타 (Eta)	η	H		타우 (Tau)	τ	T
세타 (Theta)	θ	Θ		입실론(Upsilon)	υ	Υ
요타 (Iota)	ι	I		피 (Phi)	ϕ	Φ
카파 (Kappa)	κ	K		카이 (Chi)	χ	Χ
람다 (Lambda)	λ	Λ		프사이 (Psi)	ψ	Ψ
뮤 (Mu)	μ	M		오메가 (Omega)	ω	Ω

Homework

- 제출방법

파일명 :

Greek -이름-학번.py

예) Greek -김국민-20211234.py

- 파일이 여러 개일 경우 zip으로 묶어서 ecampus 숙제제출 link에 upload
- 제출마감
 - 2023.10.31(화) 13:00
 - 제출 마감 일시까지만 제출 가능. 마감일시 이후 ecampus 숙제제출 링크 자동 close