Structured Types 2

By 윤명근 / 박수현

수업목표

- Tuples
- Dictionaries
- Sets

Tuples

- 변경 불가능한 리스트
  - 값을 한번 정하면 추가, 수정, 삭제 불가

4.2example1.py

File Edit Format Run Options Window Help

```
score = (80, 70, 90, 60)

print ("₩n1) Tuple score =", score, "₩n")

i = 0
while i < len(score) :
    print (">> score[{idx}] = {sc}" .format(idx = i, sc = score[i]))
    i = i + 1

#error
print ("₩n3) Tuple의 요소값 변경불가")
score[1] = 100
```

```
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:₩과소사₩과소사-강의예제₩4.2example1.py", line 12, in <module>
    score[1] = 100
TypeError: 'tuple' object does not support item assignment
>>>
```

Tuples

- Tuple 대입 연산
  - Tuple 에서 여러 개의 변수로 한번에 값을 대입하는 기능

tuple_1.py

File Edit Format Run Options Window Help

```
student = (20250001, "김소프", 23)

print("₩n1) Tuple student 값 :", student)
print("2) id(student) = ", id(student))

hakbun, name, age = student

print("₩n3) hakbun =₩t", hakbun)
print("    name =₩t", name)
print("    age =₩t", age)

# tuple 값 변경
hakbun = 20600002
name = "김컴공"
age = 20

student = (hakbun, name, age)
print("₩n4) 변경된 Tuple student 값 :", student)
print("5) id(student) = ", id(student))
print("    새로운 Tuple student가 생성된 것임")

print("₩n6) 변경 후 Tuple student = ", student, "₩n")

# 갯수가 불일치
hakbun, name, age, id_2 = student
```

```
1) Tuple student 값 : (20250001, '김소프', 23)
2) id(student) =  62364616

3) hakbun =       20250001
   name =         김소프
   age =          23

4) 변경된 Tuple student 값 : (20600002, '김컴공', 20)
5) id(student) =  62364488
   새로운 Tuple student가 생성된 것임

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

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

Dictionaries

- 사전
  - 키(key)와 값(value)으로 구성된 집합 (비순차 데이터 타입)
  - 유일한 키로 검색
    - 비순차형이며 Index 접근 불가 (ex: friends[2] → 불가)
  - 프로그래밍 언어별로 사전, 맵(map), 테이블(table) 등 다양한 이름 존재

Dictionaries

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

Dictionaries 입력, 변경 및 조회

4.2example2.py

File Edit Format Run Options Window Help

```
dic = {1: "Alice", 2: "Bob", "KMU": "Kookmin"}
print("1) dic = ", dic)

dic[3] = "Carole"
print("2) 3:₩'Carole₩' 추가")
print(" ", dic)

dic.update({4.0 : 200, 1.2: 3.14})
print("3) 4.0 : 200, 1.2 : 3.14 추가")
print(" ", dic)

print("4) dic[₩"KMU₩"] 값 ")
print("  ", dic["KMU"])

print("5) dic[2] 값")
print("  ", dic[2])

print("6) dic[4.0] 값")
print("  ", dic[4.0])

print("7) dic[1.2] 값")
print("  ", dic[1.2])

print("8) dic[1] 값을 ₩"Park₩"으로 변경")
dic[1] = "Park"
print(" ", dic)

print("9) dic[1.2] 값을 ₩"Phi₩"로 변경")
dic[1.2] = "Phi"
print(" ", dic)


# 존재하지 않는 index의 값
print (dic["Alice"])
```

```
1) dic =  {1: 'Alice', 2: 'Bob', 'KMU': 'Kookmin'}
2) 3:'Carole' 추가
  {1: 'Alice', 2: 'Bob', 'KMU': 'Kookmin', 3: 'Carole'}
3) 4.0 : 200, 1.2 : 3.14 추가
  {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
  {1: 'Park', 2: 'Bob', 'KMU': 'Kookmin', 3: 'Carole', 4.0: 200, 1.2: 3.14}
9) dic[1.2] 값을 "Phi"로 변경
  {1: 'Park', 2: 'Bob', 'KMU': 'Kookmin', 3: 'Carole', 4.0: 200, 1.2: 'Phi'}
Traceback (most recent call last):
  File "C:₩과소사₩과소사-강의예제₩4.2example2.py", line 34, in <module>
    print (dic["Alice"])
KeyError: 'Alice'
```

Dictionaries

- 사전 전체 검색 방법
  - 없는 키 접근 → 에러 발생

```
dic = {2015123: "Alice", 2015200: "Bob", "KMU": "JeongReung"}
for key in dic:
    print(key, "'s value is ", dic[key])

print(dic[2017100])
```

```
2015123 's value is  Alice
2015200 '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[2017100])
KeyError: 2017100
Process finished with exit code 1
```

Dictionaries

- 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 한다.

Dictionaries 삭제

4.2example2-1.py

File Edit Format Run Options Window Help

```
student = { "name" : "김국민", "id" : "2050111", "성적" : [91, 85, 99],  "기타" : "NULL" }

print("1) student = ", student)

print("₩n2) 학과 추가")
student["학과"] = "소프트웨어"

print("3) student :", student)

print("₩n4) 장학금 추가")
student["장학금"] = 2000000

print("5) student :", student)

# Dictionary 요소 삭제. "기타" 삭제
del student["기타"]

print("6) student :", student)

# student.keys() : student의 Key만을 모아서 dict_keys 객체를 return
print("₩n7) student.keys() :", student.keys())

# student.values() : student의 Values만을 모아서 dict_values 객체를 return
print("₩n8) student.values() :", student.values())

# student.items() :  studentdml Key와 Value의 쌍을 tuple로 묶은 값을
#                    dict_items 객체로 return
print("₩n9) student.items() :", student.items())

# Key: Value 쌍 모두 지우기(clear)
student.clear()
print("10) student :", student)
```

```
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) student : {'name': '김국민', 'id': '2050111', '성적': [91, 85, 99], '학과': '소프트웨어', '장학금': 2000000}

7) student.keys() : dict_keys(['name', 'id', '성적', '학과', '장학금'])

8) student.values() : dict_values(['김국민', '2050111', [91, 85, 99], '소프트웨어', 2000000])

9) student.items() : dict_items([('name', '김국민'), ('id', '2050111'), ('성적', [91, 85, 99]), ('학과', '소프트웨어'), ('장학금', 2000000)])
10) student : {}
```

Dictionaries - 사전 전체 검색 방법

4.2example3.py

File Edit Format Run Options Window Help

```
dic ={2050123: "김국민", 2050200: "이국민", "주소": "성북구 정릉로"}

print("1) dic = ", dic)

print("₩n2) for문을 이용한 key, value출력")
for key in dic:
    print (">> key :", key, ", value :", dic[key])

print("₩n3) dic.items()")
print(dic.items())

print("₩n4) for문을 이용한 dic.items()에서의 key, value 출력")

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

dic_list = list(dic.items())

print("₩n5) dic_list =", dic_list)

print("₩n6) dir(dic)")
print(dir(dic))
```

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

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

3) dic.items()
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__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
>>>
```

국민

Dictionaries

- 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

```
dic = {2050123: "Alice", 2050200: "Bob", "KMU": "JeongReung"}

for key in dic:
    print(">> key =", key, ", value =", dic[key])

print("₩n>> Find key existance.")

if 2025100 in dic :
    print(">> 존재하는 key 입니다.")
    print(dic[2017100])
else:
    print(">> 2025100은 존재하지 않는 Key 입니다.")
```

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

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

Dictionaries

- 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

checkup_key_existance2.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 id = dic.get(2030100)
7 print("Wn>> 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("Wn>> ID:", id)
13
```

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

>> ID: None

>> ID: 김국민
```

Sets

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

Sets

- 집합

4.2example4.py

File Edit Format Run Options Window Help

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

print ("A =",A)
print ("B =",B)

print ("A union B =", A|B)
print ("A intersect B =", A&B)
print ("A minus B =", A-B)
```

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


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


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


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

Sets

- 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 ₩"11₩" 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 ₩"3₩" 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}
```

Sets

- 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) ₩"3₩" 삭제(dicard 사용): ")
6 print("3) 삭제 후 : ", prime_numbers)
7
8 print("₩n4) 현존하지 않는 원소 ₩"5₩" 삭제시도(dicard 사용)")
9 prime_numbers.discard(5)
10 print("5) ₩"5₩" 삭제 시도 후(discard 사용) 내용 변화 없음 : ", prime_numbers)
11
12 print("₩n6) 현존하지 않는 원소 ₩"5₩" 삭제시도(remove사용). error 발생")
13 prime_numbers.remove(5)
14
```

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

4) 현존하지 않는 원소 "5" 삭제시도(dicard 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:₩과소사₩과소사-강의예제₩set_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.

이 름	그리스 문자	
	소문자	대문자
알파 (Alpha)	α	A
베타 (Beta)	β	B
감마 (Gamma)	γ	Γ
델타 (Delta)	δ	Δ
엡실론 (Epdilon)	ε	E
제타 (Zeta)	ζ	Z
에타 (Eta)	η	H
세타 (Theta)	θ	Θ
요타 (Iota)	ι	I
카파 (Kappa)	κ	K
람다 (Lambda)	λ	Λ
뮤 (Mu)	μ	M

이 름	그리스 문자	
	소문자	대문자
뉴 (Nu)	ν	N
크사이 (Xi)	ξ	Ξ
오미크론 (Omicron)	ο	O
파이 (Pi)	π	Π
로 (Rho)	ρ	P
시그마 (Sigma)	σ	Σ
타우 (Tau)	τ	T
입실론(Upsilon)	υ	Y
피 (Phi)	φ	Φ
카이 (Chi)	χ	X
프사이 (Psi)	ψ	Ψ
오메가 (Omega)	ω	Ω

Homework

- 다음 수업 시작 전까지 ecampus로 제출
- 제출방법

화일명 :

Greek -이름-학번-일시.py

예) Greek -김국민-20201234-20200424.py

- 파일이 여러 개일 경우 zip으로 묶어서 제출