데이터 처리를 위한 Python 프로그래밍 입문

5-2강. 데이터 구조 : tuple, set, dictionary

ERICA 2018-2

강의 내용

- Tuple
- Set
- Dictionary

Tuple(1/9)

- Tuple
  - List형을 사용하다보면, 원소를 변경할 수 없게 만들어야 하는 경우
  - 원소의 순서가 바뀌어서는 안 되는 경우
  - Tuple형 기호 : 변수 생성시 소괄호(( ))로 사용, 원소들은 쉼표(,)로 구분

```
>>> days = ('Mon','Tue','Wed','Thu','Fir','Sat','Sun')
>>> days
('Mon', 'Tue', 'Wed', 'Thu', 'Fir', 'Sat', 'Sun')
>>>
```

Tuple(2/9)

- Tuple

```
>>> string_march = 'March'
>>> string_march
'March'
>>> tuple_march =tuple(string_march)
>>> tuple_march
('M', 'a', 'r', 'c', 'h')
>>>
>>> list_spring =['March','April','May']
>>> list_spring
['March', 'April', 'May']
>>> tuple_spring = tuple(list_spring)
>>> tuple_spring
('March', 'April', 'May')
>>>
>>>
>>> type(string_march)
<class 'str'>
>>> type(tuple_march)
<class 'tuple'>
>>> type(list_spring)
<class 'list'>
>>> type(tuple_spring)
<class 'tuple'>
>>>
```

* String형으로부터 tuple형을 생성

* list형으로부터 tuple형을 생성

* type() 함수를 활용하여 데이터 타입, 구조를 확인합니다.

Tuple(3/9)

- Tuple index

```
>>> tuple_t = (1,2,3,4)
>>> tuple_t[3]
4
>>>
```

```
>>> t = ('Valentine',2,14,'gift')
>>> t[0]
'Valentine'
>>> t[3]
'gift'
>>>
>>> t[4] ='chocolate'
Traceback (most recent call last):
  File "<pyshell#39>", line 1, in <module>
    t[4] ='chocolate'
TypeError: 'tuple' object does not support item assignment
>>>
>>>
>>> t[3] = 'chocolate'
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    t[3] = 'chocolate'
TypeError: 'tuple' object does not support item assignment
>>>
>>> t
('Valentine', 2, 14, 'gift')
```

* Tuple형의 원소들의 대소문자는 당연히 구분됩니다.

- Tuple형의 존재하지 않는 index를 수정하려고하면 Error메세지 확인합니다.
- Tuple형의 원소는 변경 불가능한 것을 확인합니다. Error메세지 확인합니다.

Tuple(4/9)

- Tuple 연산
  - List형과 같이 원소로 구성되어있고, index구조이기 때문에 slice기능, 병합(+), 반복(*)연산이 가능

```
>>> t
('Valentine', 2, 14, 'gift')
>>> t[1:3]
(2, 14)
>>> tt = ('love',)
>>> tt[0]
'love'
>>> tt[:]
('love',)
>>>
>>> t + tt
('Valentine', 2, 14, 'gift', 'love')
>>>
>>> print(tt * 6)
('love', 'love', 'love', 'love', 'love', 'love')
>>> t*2
('Valentine', 2, 14, 'gift', 'Valentine', 2, 14, 'gift')
```

* Index1부터 index2-1까지 원소를 반환합니다.

* 두 개의 Tuple을 병합합니다.

* tt변수 Tuple을 6번 반복합니다.

Tuple(5/9)

- Tuple 길이 계산

```
>>> t1 = (3,6,9,11)
>>> t2 = ('love','Power','평화')
>>>
>>> t1
(3, 6, 9, 11)
>>> t2
('love', 'Power', '평화')
>>> print(t1, t2)
(3, 6, 9, 11) ('love', 'Power', '평화')
>>> len(t1, t2)
Traceback (most recent call last):
  File "<pyshell#64>", line 1, in <module>
    len(t1, t2)
TypeError: len() takes exactly one argument (2 given)
>>> len(t1)
4
>>> len(t2)
3
>>>
```

- 변수명의 원소들을 확인합니다.
  소괄호 ( )로 묶인 것으로 tuple임을 알 수 있습니다.
- Len()함수를 통해 tuple 길이를 확인합니다.
  여러 변수 확인은 불가능합니다.

Tuple(6/9)

- Tuple 활용 : 코드간소화
  - 상황 : 일반 변수선언과 조건문을 활용한 코드를 확인한다.
    튜플을 적용하여 동일한 결과 화면이 나오게 코드를 간소화

Tuple(7/9)

- Tuple 활용 : 코드간소화
  - 일반 코드

```
import random

# 답변을 입력해봅시다.
ans1="자! 해보세요!"
ans2="됐네요, 이 사람아"
ans3="뭐라고? 다시 생각해보세요."
ans4="모르니 두려운 것입니다."
ans5="칠푼인가요?? 제 정신이 아니군요!"
ans6="당신이라면 할 수 있어요!"
ans7="해도 그만, 안 해도 그만, 아니겠어요?"
ans8="맞아요, 당신은 올바른 선택을 했어요."

print("MyMagic8Ball에 오신 것을 환영합니다.")

# 사용자의 질문을 입력 받습니다.
question = input("조언을 구하고 싶으면 질문을 입력하고 엔터 키를 누르세요.₩n")

print("고민 중 ...₩n" * 4)

# 질문에 알맞은 답변을 하는 일에 randint() 함수를 활용합니다.
choice=random.randint(1, 8)
if choice==1:
    answer=ans1
elif choice==2:
    answer=ans2
elif choice==3:
    answer=ans3
elif choice==4:
    answer=ans4
elif choice==5:
    answer=ans5
elif choice==6:
    answer=ans6
elif choice==7:
    answer=ans7
else:
    answer=ans8

# 화면에 답변을 출력합니다.
print(answer)

input("₩n₩n마치려면 엔터 키를 누르세요.")
```

Tuple(8/9)

- Tuple 활용 : 코드간소화
  - 간소화 된 코드

```
import random


# 답을 튜플에 넣습니다.
answers = (
    "자! 해보세요!",
    "됐네요, 이 사람아",
    "뭐라고? 다시 생각해보세요.",
    "모르니 두려운 것입니다.",
    "칠푼인가요?? 제 정신이 아니군요!",
    "당신이라면 할 수 있어요!",
    "해도 그만, 안 해도 그만, 아니겠어요?",
    "맞아요, 당신은 올바른 선택을 했어요."
    )

print("MyMagic8Ball에 오신 것을 환영합니다.")

# 사용자의 질문을 입력 받습니다.
question = input("조언을 구하고 싶으면 질문을 입력하고 엔터 키를 누르세요.₩n")

print("고민 중 ...₩n" * 4)

# 질문에 알맞은 답변을 하는 일에 randint() 함수를 활용합니다.
choice=random.randint(0, 7)

# 화면에 답변을 출력합니다.
print(answers[choice])

# 이제 종료합니다.
input("₩n₩n마치려면 엔터 키를 누르세요.")
```

Tuple(9/9)

- Tuple과 list의 비교

구 분	List형	Tuple형
공통점	원소로 구성(쉼표로 원소 구분)	
	시퀀스 형 (Slice, indexing 가능)	
차이점	대괄호로 표기 : [ ]	소괄호로 표기 : ( )
	원소 삽입, 삭제, 교체 가능	원소 삽입, 삭제, 교체 불가능

Set(1/5)

- Set
  - 집합의 표현
  - set형 기호 : 변수 생성시 중괄호({ })로 사용, 원소들은 쉼표(,)로 구분
  - List나 tuple에서 사용하였던 indexing, slice 연산은 사용 할 수 없음
    - Set은 index구조가 아니며, 원소들의 순서가 지정되어 있지 않음
    - 만일 set에서 indexing으로 접근하려면 list형이나 tuple형으로 형변환 후 가능함.
  - Set형은 중복을 허용하지 않음
  - Set형은 순서가 없음

Set(2/5)

- set

```
>>> odd_num = {2,4,6,8}
>>> fruit = {'apple','banana','peach'}
>>>
>>> odd_num
{8, 2, 4, 6}
>>> fruit
{'apple', 'peach', 'banana'}
>>>
```

- 변수명의 원소들을 확인합니다.
  중괄호 { } 로 묶인 것으로 set임을 알 수 있습니다.

```
>>> set_hello = set('hello')
>>> type(set_hello)
<class 'set'>
>>>
>>> set_hello
{'e', 'h', 'l', 'o'}
>>>
```

- String ‘hello’ 를 set형으로 변환하고
  type() 함수로 set임을 확인합니다.
- 중복된 단어는 어떻게 표현되었는지 확인합니다.
- 순서는 어떻게 표현되었는지 확인합니다.

Set(3/5)

- Set 길이

```
>>> set_number = {1,2,3,4}
>>> set_word ={'one','two','three'}
>>>
>>> len(set_number)
4
>>> len(set_word)
3
>>>
```

```
>>> s1 = {0,1,3,5,7,9}
>>> s2 = {0,2,4,6,8,9}
>>> s1 & s2
{0, 9}
>>>
>>> s1 | s2
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> s1 - s2
{1, 3, 5, 7}
>>>
```

```
>>> s3 = {'red','blue','yellow'}
>>> 'red' in s3
True
>>> 'black' in s3
False
>>>
```

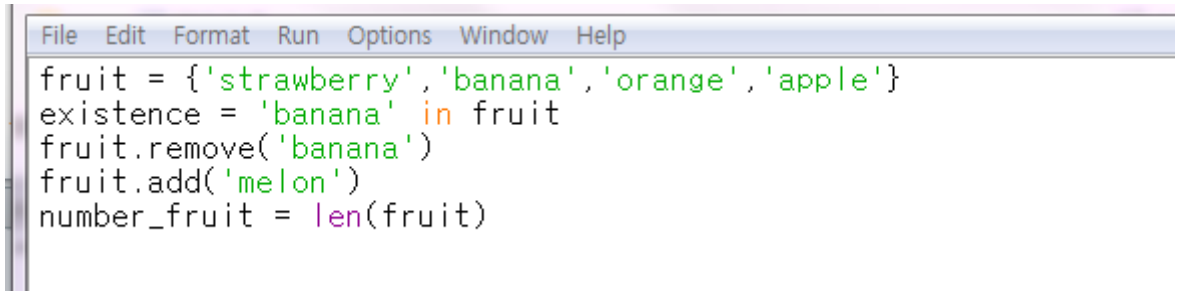
Set(4/5)

- Set 활용 : 냉장고 과일
  - 상황 : 냉장고에 딸기, 바나나, 오렌지, 사과가 있다. Set형을 생성한다.
    냉장고에 바나나가 있는지 확인한다.
    과일들 중 바나나를 먹고, 멜론을 사와서 냉장고에 넣었을 때
    남아있는 과일은 무엇인가?
    냉장고에 남아 있는 과일종류 개수는 몇 개인가?
  - 결과 화면

```
>>> existence
True
>>> fruit
{'orange', 'strawberry', 'melon', 'apple'}
>>> print('냉장고에 남아있는 과일 ', number_fruit,'개')
냉장고에 남아있는 과일  4 개
>>>
>>>
```

Set(5/5)

- Set 활용 : 냉장고 과일
  - 해결 코드



Dictionary(1/4)

- Dictionary
  - 사전과 비슷한 형식의 자료형
  - Key, value쌍으로 구성
  - 변수생성시 중괄호({ })로 생성, 원소구분은 쉼표(,)
  - 원소는 key와 value로 구성되어 콜론(:)으로 구분
  - Key :변경할 수 없는 자료형이 사용됨 : tuple 사용가능 , List 사용불가
  - Value : 모든 자료형이 사용 가능

Dictionary(2/4)

- Dictionary

```
>>> price = {('떡볶이','김밥'):'3000원',('라면','만두'):'4000원'}
>>> price
{('떡볶이', '김밥'): '3000원', ('라면', '만두'): '4000원'}
>>>
>>> price = {['떡볶이','김밥']:'3000원',['라면','만두']:'4000원'}
Traceback (most recent call last):
  File "<pyshell#116>", line 1, in <module>
    price = {['떡볶이','김밥']:'3000원',['라면','만두']:'4000원'}
TypeError: unhashable type: 'list'
>>>
```

- **Dictionary 자료형으로 소괄호 ()를 사용한 tuple형이 가능합니다.**
  대괄호 [ ] 를 사용한 list형은 에러가 발생함을 확인합니다.

Dictionary(3/4)

- Dictionary

```
>>> price = {'떡볶이':'3000원','만두':'4000원'}
>>> price['만두']
'4000원'
>>>
>>> price['만두']='3000원'
>>> price['만두']
'3000원'
>>>
```

- **Key값으로 index하고 value값을 변경할 수 있음을 알 수 있습니다.**

Dictionary(4/4)

- Dictionary

```
>>> food={'한식':['보리밥','비빔밥'],'중식':['자장면','짬뽕']}
>>> food
{'중식': ['자장면', '짬뽕'], '한식': ['보리밥', '비빔밥']}
>>> food['일식']=['초밥','돈부리']
>>> food
{'중식': ['자장면', '짬뽕'], '일식': ['초밥', '돈부리'], '한식': ['보리밥', '비빔밥']}
>>>
>>> del food['중식']
>>> food
{'일식': ['초밥', '돈부리'], '한식': ['보리밥', '비빔밥']}
>>>
```

- **Key: value를 추가하고, 삭제할 수 있음을 알 수 있습니다.**

```
>>> print('일식' in food.keys())
True
>>> print('중식' in food.keys())
False
>>>
```

List, Tuple, set, dictionary

- Tuple형은 list구조로 사용이 거의 동일하다는 공통점이 있으나, 한 번 생성되면 원소의 변경이 불가능하다는 차이점이 있음
- Set형은 집합과 관련된 자료형으로 중복된 원소를 허용하지 않음
- Dictionary형은 key와 value가 한 쌍을 이루는 원소로 이루어 지며, key값은 고유하여야 함으로 tuple형을 사용하여야 함.

구 분	표현 기호
List	대괄호로 표기 : [ ]
Tuple	소괄호로 표기 : ( )
Set	중괄호로 표기 : { }
dictionary	중괄호로 표기 : {key:value }

Thank you