모든 계열을 위한 기초 교육

지능형로봇프로그램밍의 이해

7 : 튜플 집합 딕셔너리

목차

2

1. 튜플의 개념

- 튜플 정의
  - tuple은 본질적으로 list와 매우 유사하지만. 요소가 변경 불가능한 선형 자료구조이다. tuple이 생성되면 요소로 구성되며, 순서가 존재한다. tuple의 요소는 어떤 타입도 가능하다. 단 새로운 요소를 추가, 삭제, 교체, 및 수정 하는 것은 제한된다. 또한 list 와 구분하기위해 소괄호로 표기한다.

variable_name = (element 1, element 2, ... element n)	( ) : tuple 기호

  - 튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷하다.

구분	선언 문법 구성	값 변경 가능 여부
리스트	'[', ']'	수정, 삭제 가능
튜플	'(', ')'	값 변경 불가

1. 튜플의 개념

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

- 튜플 기초
  - 요소가 하나인 tuple은 ‘,’ 구분 기호를 포함해서 나타낸다.
    - Ex) (10,)
  - 동일한 자료형을 요소로 갖는 tuple이다
    - Ex) (1, 2, 3, 4), ('hello', 'welcome', 'Hi')
  - 혼합 자료형을 요소로 갖는 tuple이다
    - Ex) (1,2,3, [4,5,6], (7,8),9, (‘apple’, ‘orange’))
  - Empty tuple : 요소를 하나도 갖지 않는 tuple이다.
    - Ex) ( )
  - list 및 문자열로부터 tuple을 생성할 수 있다. 문자일 안에 각 문자들은 tuple요소가 된다.
    - Ex) tuple([1,2,3]), tuple(‘python’)

1. 튜플의 개념

- **튜플 활용**
  - ➢ **변수는 사용하기 전에 반드시 할당 되어야한다**
    - **EX)**

```
>>> numbers = (1, 2, 3)                #숫자를 가지는 tuple
>>> strTuple = ('hanyang', 'AI', 'Computer')          #문자열을 가지는 tuple
>>> print(numbers, strTuple)
(1, 2, 3) ('hanyang', 'AI', 'Computer')
```

1. 튜플 선언 방법

- 다음 예를 통해 튜플을 선언해보자.

```
>>> tp1 = ()     # 비어있는 튜플
>>> tp2 = (10,) # 요소의 값을 하나만 선언할 때 반드시 콤마를 붙여야 함
>>> tp3 = (10,20,30)
>>> tp4 = 10,20,30 # 괄호를 생략해도 무방
```

- 튜플 요소 값 삭제/변경 불가능

```
>>> tp1 = (10,20,30)
>>> del tp1[0]  # 삭제 불가능: Error 발생
>>> tp1[0] = 10 # 변경 불가능: Error 발생
```

2. 튜플의 연산

- tuple은 list에서 사용한 병합 및 반복하는 일반 연산도 사용한다. len(), min(), max(), sum()함수를 tuple에서 사용할 수 있다. 또한 tuple안에 slice기능도 가능하다.
  - 다음 예를 코드를 작성하여 살펴보자

```
>>> tuple1 = ('Red', 'Green', 'Blue')
>>> tuple2=tuple([5,6,7,8])
>>> tuple3=tuple1+tuple2
>>> print(tuple3)
('Red', 'Green', 'Blue', 5, 6, 7, 8)
>>> tuple4=tuple2*3
>>> tuple4
(5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8)
>>> len(tuple1)
3
>>> max(tuple2)
8
>>> min(tuple2)
5
>>> sum(tuple2)
26
```

2. 튜플의 연산 - 인덱싱

- tuple인덱싱은 문자열, 리스트 인덱싱과 동일하다.
  - 다음 예를 코드를 작성하여 살펴보자

```
>>> tp1 = (10,20,'one')
>>> tp1[0]   # 인덱스 0번 위치(첫번째 위치) 요소 값
10
>>> tp1[2]   # 인덱스 2번 위치(세번째 위치) 요소 값
'one'
```

2. 튜플의 연산 - 슬라이스

- 시퀀스[start : end]
  - start에서 시작해서 end 이전 까지 추출
  - start는 생략이 가능하며 기본값이 '0'
  - end는 음수로 표현 가능
  - start나 end는 생략 가능하며 생략되면 처음 또는 끝을 의미
- 시퀀스[start : end : n] 표현도 가능
  - start에서 end이전 까지 매 n번째 항목 추출
- 시퀀스 추출에서의 콜론 : 사용
  - 단독으로 사용될 경우: 전체를 의미
  - 좌우로 사용될: 처음과 끝을 의미

2. 튜플의 연산 - 슬라이스

- 다음 예를 통해 살펴보자

```
>>> tu = (23, 'abc', 4.5, (2, 3), 'def')
>>> tu[1:4]
('abc', 4.5, (2, 3))

>>> tu[1:-1]
('abc', 4.5, (2, 3))

>>> tu[:2]
(23, 'abc')

>>> tu[2:]
(4.5, (2, 3), 'def')

>>> tu[0:5:2]
(23, 4,5, 'def')

>>> tu[:]
(23, 'abc', 4.5, (2, 3), 'def')
```

```
>>> tu[3]
(2, 3)

>>> tu[-2][1]     #튜플내 튜플
3

>>> tu[-1][2]     #튜플낼 문자열
f
```

3. 집합의 개념

- 집합 정의
  - 집합에 관련된 자료형이다. 데이터의 중복이 없다. 순서가 없는(unordered) 집합체 자료형으로 색인 사용이 안된다. 특정 위치의 항목을 접근하여 사용할 수가 없다.

variable_name = { element 1, element 2, ... element n }	{ } : set 기호

- 사용 사례
  - 어떠한 항목의 존재 여부를 확인하거나, 자료형의 중복을 제거하기 위한 필터 역할로 사용될 수 있다.
  - 교집합, 차집합, 합집합을 적용한다.

3. 집합 개념

- 집합 기초
  - 요소의 구분은 ‘,’ 구분 기호를 포함해서 나타낸다.
    - Ex) {10,20}
  - 동일한 자료형을 요소로 갖는 set이다
    - Ex) {1, 2, 3, 4}, {'hello', 'welcome', 'Hi'}
  - 혼합 자료형을 요소로 갖는 set이다
    - Ex) {1,2,3, [4,5,6], (7,8), 3.14, (‘apple’, ‘orange’)}
  - Empty set : 요소를 하나도 갖지 않는 set이다.
    - Ex) { }
  - list, tuple 및 문자열로부터 set함수를 사용하여 생성할 수 있다.
    - Ex) set{[1,2,3]}, set{‘python’}

3. 집합 개념

- 집합 사용
  - EX)

```
>>> numbers = {1, 2, 3}                #숫자를 가지는 set
>>> strset = {'hanyang', 'AI', 'Computer'}  #문자열을 가지는 set
>>> print(numbers, strset)
{1, 2, 3} {'Computer', 'hanyang', 'AI'}
```

4. 집합 함수

- 다음 예를 통해 집합 요소 추가 및 삭제해보자.

연산자	동 작
<set>add.(element)	집합에 요소 추가
<set>remove.(element)	집합에 요소 제거
<set1>.update(list)	입력받은 list의 항목 일괄 추가
<set1> = set2.copy()	집합1에 집합2의 요소들을 복사

```
>>> set_number = {1,2,3,4,5,6}
>>> set_number.add(7)
>>> set_number
{1, 2, 3, 4, 5, 6, 7}

>>> set_number.remove(1)
```

```
>>> set1 = {1,3,5,7}
>>> list1 = [2,4,6,8]
>>> set1.update(list1)
>>> set1
{1, 2, 3, 4, 5, 6, 7, 8}
>>> set2 = set1.copy()
```

4. 집합 함수

- **set()**함수를 사용하여 문자열, list, tuple등을 기반으로 세트 생성
  - 문자열이 입력 값으로 들어가면 문자별로 세트에 추가

```
#빈 세트
set_empty = { }

>>> set_hello = set('hello')
>>> set_hello
{'o','h','e','l'}          #중복된 값 ' l ' 삭제


>>> set_number = set([7,1,2,3,2,3,1,4,6,4,5])
>>> set_number
{1,2,3,4,5,6,7}          #중복된 값  삭제
```

4. 집합 함수

- set은 문자열, 리스트, 튜플에서 사용된 함수 사용 가능하다. len(), min(), max()함수를 set에서 사용할 수 있다.
  - 다음 예를 코드를 작성하여 살펴보자

```
>>> set1 = {'Red', 'Green', 'Blue'}
>>> set2=set([5,6,7,8])
>>> len(set1)
3
>>> len(set2)
4
>>> max(set1)
'Red‘
>>> min(set1)
'Blue‘
>>> max(set2)
8
```

5. 집합 활용

- set은 교집합. 합집한, 차집합을 사용할 수 있다.
- 교집합
  - 기호 “&”를 사용
  - intersection() 함수 를 사용
- 합집합
  - 기호 “ | ”를 사용
  - union() 함수 를 사용
- 차집합
  - 기호 “ - ”를 사용
  - difference() 함수 를 사용

```
s1 = set([1, 2, 3, 4, 5, 6])
s2 = set([4, 5, 6, 7, 8, 9])
```

```
>>> s1 & s2
{4, 5, 6}
>>> s1.intersection(s2)
{4, 5, 6}
```

```
>>> s1 | s2
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> s1.union(s2)
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
>>> s1 - s2
{1, 2, 3}
>>> s2.difference(s1)
{7, 8, 9}
```

6. 딕셔너리의 개념

- 딕셔너리 정의
  - 키-벨류 쌍(key-value pair)형태를 항목으로 둔 시퀀스 집합이다. 키-벨류의 자료형은 정수, 실수, 문자 사용이 가능하다. 시퀀스 집합에서 정수 이외의 색인을 사용할 수 있다. 임의의 키를 이용해 연관된 값을 검색할 수 있다. 즉 리스트나 튜플처럼 순차적으로 자료를 구하지 않고, key를 통하여 value를 얻는다.

variable_name = {<key_1>:<$value_1$>, <key_2>:<$value_2$>, ... }	{ } : 딕셔너리 기호

6. 딕셔너리 개념

- 딕셔너리 기초
  - 요소의 구분은 ‘,’ 구분 기호를 포함해서 나타낸다.
  - 딕셔너리 이름은 변수 명 생성과 동일하다.
  - 데이터 저장: 키 : 벨류 쌍(Key : Value Pair)으로 저장
    - Ex) dic = {‘id’:‘swcho', 'pw':12456, 'email':‘swcho@kocw.org'}

주의) key는 고유한 값이기 때문에 동일한 key를 선언하게 되면 앞에 값이 무시된다

6. 딕셔너리 개념

- 딕셔너리 사용
  - EX)

```
>>> d1 = {101:'smart', 102:'graphic'}
>>> d2 = {'one':[10, 20, 30]}
>>> print(d1,d2)
{101: 'smart', 102: 'graphic'} {'one': [10, 20, 30]}
```

6. 딕셔너리 개념

- 딕셔너리 추가
  - 딕셔너리[추가할 키] = 추가할 값
  - **EX)**

```
>>> d1 = {101:'smart', 102:'graphic'}
>>> d1[201] = ‘orange'
>>> d1
{101: 'smart', 102: 'graphic', 201: 'orange'}
>>> d1['one'] = [1, 2, 3]
>>> d1
{101: 'smart', 102: 'graphic', 201: 'orange', 'one': [1, 2, 3]}
```

6. 딕셔너리 개념

- 딕셔너리 삭제
  - del 딕셔너리[삭제할 키] 또는 del(딕셔너리[삭제할 키])
  - EX)

```
>>> d1 ={101: 'smart', 102: 'graphic', 201: 'orange', 'one': [1, 2, 3]}
>>> del d1['one']
{101: 'smart', 102: 'graphic', 201: 'orange'}
>>> del d1[201]
d1
{101: 'smart', 102: 'graphic'}
>>>
```

6. 딕셔너리 개념

- 딕셔너리 요소 값 추출
  - 딕셔너리명[key]를 이용해서 value를 추출
  - 딕셔너리명 [value]: 사용 불가
  - EX)

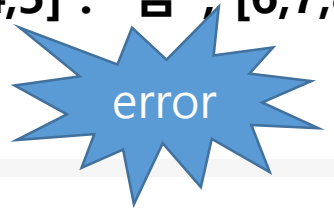
```
>>> dic = {"user":"Hong", "pwd":1234, "country":"Korea"}
>>> dic["user"]        # key "user" 입력
'Hong'
>>> dic["pwd"]         # key "pwd" 입력
1234
>>> dic["Korea"]       # value "Korea" 값 입력
```

error

6. 딕셔너리 개념

- 딕셔너리에서의 항목 변경
  - key는 값이 정해지면 변경 불가(immutable), 리스트 자료형은 키로 사용 불가
  - 튜플은 키로 사용 가능, value의 값은 변경 가능(mutable)
    - EX)

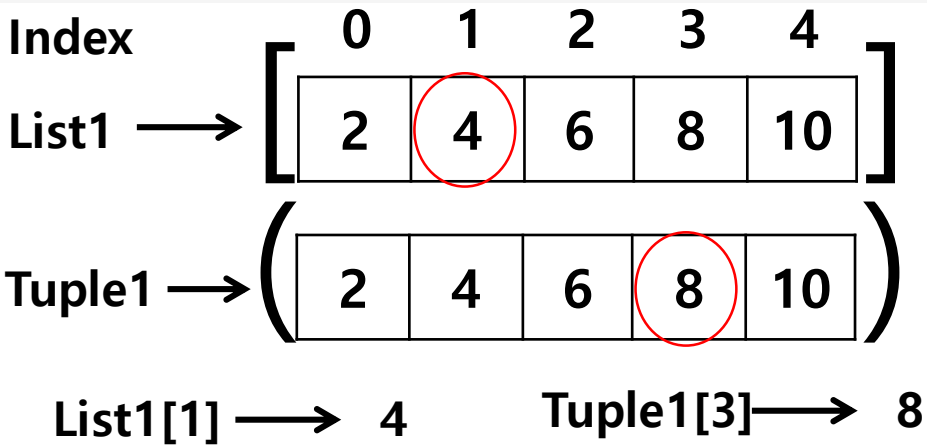
```
>>> dic = {"user":"Hong", "pwd":1234, "country":"Korea"}
>>> dic["user"] = "Kim"      # key "user" 의 value값 변경
>>> dic
{'user': 'Kim', 'pwd': 1234, 'country': 'Korea'}

>>> Dict1 = {  [3,4,5] : "봄", [6,7,8] : "여름", [9,10,11] : "가을", [12, 1, 2] : "겨울"  }
```



7. 딕셔너리 구조 비교

List1 = [2, 4, 6, 8, 10]

Tuple1 = (2, 4, 6, 8, 10)



pairs

Dict1 = {"Spring" : "봄" , "Summer" : "여름", "Fall" : "가을", "Winter" : "겨울"}

Dict1 →

key	value
Spring	봄
Summer	여름
Fall	가을
Winter	겨울

4 pairs

Dict1["Spring"] → "봄"

Dict1["Fall"] → "가을"

8. 딕셔너리의 활용 - 함수

- 딕셔너리 검색 및 반환

연산자	동 작
<key> in <dict>	딕셔너리가 <key>를 포함하면 **True**, 아니면 **False**
<dict>.keys()	딕셔너리의 key들을 리스트로 반환
<dict>.values()	딕셔너리의 value들을 리스트로 반환
<dict>.items()	키-값의 쌍을 튜플 (key, value) 시퀀스로 반환

```
dic2 = {"guido":"python", "jobs":"iphone", "bill":"windows"}
>>> "bill" in dic2
True
>>> "fred" in dic2
False
```

8. 딕셔너리의 활용 - 함수(cont'd)

- 딕셔너리 반환 함수 사용 예

```
dic2 = {"guido":"python", "jobs":"iphone", "bill":"windows"}

>>> dic2.keys()              # 해당 딕셔너리의 key만 가져옴
dict_keys(['guido', 'jobs', 'bill'])



>>> dic2.values()              # 해당 딕셔러니의 value만 가져옴
dict_values(['python', 'iphone', 'windows']



>>> dic2.items()       # 해당 딕셔러니의 모든 키:벨류 쌍을 가져옴
dict_items([('guido','python'), ('jobs','iphone'), ('bill' 'windows')])
```

8. 딕셔너리의 활용 - 함수(cont'd)

- 딕셔너리 삽입, 반환, 삭제 함수
  - `list()`함수를 사용하여 결과 값을 리스트로 변환

```
dic2 = {"guido":"python", "jobs":"iphone", "bill":"windows"}

>>> list(dic2.keys())        # 해당 딕셔너리의 key만 가져옴
['guido', 'jobs', 'bill']

>>> list(dic2.values())      # 해당 딕셔러니의 value만 가져옴
['python', 'iphone', 'windows']

>>> list(dic2.items())  # 해당 딕셔러니의 모든 키:벨류 쌍을 가져옴
[('guido', 'python'), ('jobs', 'iphone'), ('bill', 'windows')]
```

8. 딕셔너리의 활용 - 함수(cont'd)

- 키-값 삽입, 반환, 삭제

연산자	동 작
<dict>[<key>] = value	딕셔너리에 새로운 key, value를 추가한다.
del <dict>[<key>]	<key> 값에 해당하는 엔트리(키-값 쌍)를 제거
<dict>.get(<key>, <default>)	<key>값이 존재하면 를 해당 value를 반환 <key>값이 없으면 <default>를 반환
<dict>.clear()	딕셔너리의 모든 키:값 쌍을 삭제
for <key> in <dict>:	딕셔너리에 있는 키값이 <key>에 배정되어 반복문 수행

8. 딕셔너리의 활용- 삽입, 반환, 삭제 예제


```
>>> dic2 = {"guido":"python", "jobs":"iphone", "bill":"windows"}
>>> dic2["mark"] = "facebook "              #새로운 key:value 추가
>>> dic2
{'bill': 'windows', 'guido': 'python', 'jobs': 'iphone', 'mark': 'facebook'}

>>> dic2.get('bill','unknown ' )
'windows'
>>> dic2.get('jobs','unknown')
'iphone'
>>> del dic2["jobs"]          # key "jobs" 삭제, value "iphone"도 삭제
>>> dic2.get('jobs','unknown')
'unknown'

>>> dic2.clear()          # 딕셔너리의 모든 키-벨류 쌍 삭제
>>> dic2
{}
```

8. 딕셔너리의 활용

- 딕셔너리 for문에 의한 탐색

```
>>> dic2= {'bill': 'windows', 'guido': 'python', 'jobs': 'iphone', 'mark': 'facebook'}
>>> for key in dic2:
        print("key:{0}\t | value:{1}".format(key, dic2[key]))
```

실행결과

```
key:bill        | value:windows
key:guido       | value:python
key:jobs        | value:iphone
key:mark        | value:facebook
```

주의)딕셔너리는 만들어진 순서와 출력되는 순서가 다를 수 있음

8. 딕셔너리의 활용

- 가위·바위·보 프로그램:
  - 중첩 if문사용

```
import random
rps = ['가위','바위','보']
com = random.choice(rps)
player = input('가위, 바위, 보 중에서 하나를 입력하세요: ')
if player == '가위' :
    if com == '가위':
        print('비겼습니다')
    elif com == '바위':
        print('졌습니다')
    elif com == '보':
        print('이겼습니다')
```

```
if player == '바위' :
    if com == '가위':
        print('이겼습니다')
    elif com == '바위':
        print('비겼습니다')
    elif com == '보':
        print('졌습니다')
if player == '보' :
    if com == '가위':
        print('졌습니다')
    elif com == '바위':
        print('이겼습니다')
    elif com == '보':
        print('비겼습니다')
```

8. 딕셔너리의 활용

- 가위·바위·보 프로그램:
  - 딕셔너리사용

```
import random
rps = ['가위','바위','보']
result = {
    ('가위','보'):True,
    ('가위','바위'):False,
    ('바위','가위'):True,
    ('바위','보'):False,
    ('보','바위'):True,
    ('보','가위'):False,
    }
```

```
com = random.choice(rps)
player = input('가위, 바위, 보 중에서 하나를 입력하세요: ')
if  player == com :
    print('비겼습니다')
elif  result[(player,com)]  :
    print('이겼습니다!!')
else:
    print('졌습니다')
```

연습문제

1. 집합에서 교집합을 구하는 메서드를 고르시오.

① difference

② union

③ intersection

④ len

연습문제

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

2. 집합에서 사용되는 연산자가 아닌 것을 고르시오.

① &

② |

③ -

④ +

연습문제

3. dictionary형은 ( )와 ( )가 한 쌍을 이루는 원소로 구성되어 있다.

연습문제

4. 딕셔너리에서 키만 고르는 함수와 값만 고르는 함수를 차례대로 나열한 것을 고르시오.

① key( ), value( )

② value( ), key( )

③ keys( ), values( )

④ values( ), keys( )

연습문제

5. 어느 문구점에서 판매하는 연필은 200원, 펜은 800원, 지우개는 500원, 자는 300원이다. 이 목록을 dictionary형을 이용하여 작성해보고 가격만 list형으로 출력해보자.

감사합니다.