Programming in Python

08 – 순서쌍, 파일, 그 외

2016년 8월, 국민대학교 컴퓨터공학부

순서쌍 (Tuples)

Python 순서쌍의 특징들

- 임의의 객체를 모아 순서를 부여한 것
  - 리스트와는 무엇이 다른가?
- 오프셋 (인덱스) 을 이용하여 접근
  - 인덱싱, 슬라이싱 등이 모두 가능
- “immutable sequence”
  - 문자열과 마찬가지로, 생성된 순서쌍 개체는 변경될 수 없음
- 고정 길이이며 중첩 가능
  - 변경될 수 없으므로 길이가 고정되어 있으나, 문자열과 달리 리스트, 사전, 다른 순서쌍 등을 포함 가능
- 객체 참조의 배열
  - 내부적으로는 순서쌍을 구성하는 각 객체에 대한 참조 (포인터) 의 모음

기본적 순서쌍 연산

```
>>> (1, 2) + (3, 4)          # 병합 (concatenation)
(1, 2, 3, 4)

>>> (1, 2) * 4               # 반복
(1, 2, 1, 2, 1, 2, 1, 2)

>>> T = (1, 2, 3, 4)
>>> T[0]                     # 인덱싱
1
>>> T[1:3]                   # 슬라이싱
(2, 3)


>>> x = (40)                 # 정수형 상수에 괄호를 씌운 것
>>> x, type(x)
(40, <type 'int'>)

>>> y = (40,)                # 순서쌍
>>> y, type(y)
((40,), <type 'tuple'>)
```

리스트의 경우와 비교!

한 번 생성된 순서쌍은 변경할 수 없음

```
>>> T = ('cc', 'aa', 'dd', 'bb')
>>> tmp = list(T)
>>> tmp
['cc', 'aa', 'dd', 'bb']

>>> tmp.sort()                          # 리스트를 구성하여 정렬함
>>> tmp
['aa', 'bb', 'cc', 'dd']

>>> T = tuple(tmp)                      # 이로부터 다시 순서쌍을 생성
>>> T
('aa', 'bb', 'cc', 'dd')

>>> sorted(('cc', 'aa', 'dd', 'bb'))    # sorted built-in 함수는 순서쌍에도 적용됨
['aa', 'bb', 'cc', 'dd']                # 그러나, 리턴은 리스트
```

변경될 수 없다는 것이 리스트와의 거의 유일한 차이점!

순서쌍의 연산

```
# 내포 - 리스트 생성

>>> T = (1, 2, 3, 4, 5)
>>> L = [x + 20 for x in T]
>>> L
[21, 22, 23, 24, 25]
```

```
# 인덱스와 카운트

>>> T = (1, 2, 3, 2, 4, 2)
>>> T.index(2)
1
>>> T.index(2, 2)
3
>>> T.count(2)
3
```

```
>>> T = (1, [2, 3], 4)
>>> len(T)                  # 순서쌍의 길이 - 포함된 원소의 수
3

>>> T[1] = 'spam'           # 순서쌍은 생성 이후 변경할 수 없음
TypeError: 'tuple' object does not support item assignment

>>> T[1][0] = 'spam'        # 하지만, 순서쌍의 원소로 포함된 객체에 대해서는 그렇지 않음
>>> T
(1, ['spam', 3], 4)
```

파일 (Files)

```
>>> f = open('data.txt', 'w')          # 새로운 파일을 출력 모드로 생성
>>> f.write('Hello\n')                 # 문자열 (바이트의 나열) 을 파일에 기록
6
>>> f.write('world\n')                 # 또다른 문자열을 기록 (뒤에 이어서 쓰여짐)
6
>>> f.close()                          # 파일 닫기 시점에 버퍼 내용을 디스크에 기록
>>> f
<closed file 'data.txt', mode 'w' at 0x…>

>>> f = open('data.txt')               # 디폴트는 읽기 모드로 파일을 열게 됨 (‘r’ 을 지정한 것과 동일)
>>> text = f.read()                    # 파일 내용 전체를 문자열로 읽어들임
>>> text
'Hello\nworld\n'
>>> print(text)
Hello
world

>>> text.split()                       # 파일의 내용은 언제나 텍스트 문자열
['Hello', 'world']
```

파일 관련 메서드들

Operation	Interpretation
`aString = input.read( )`	Read entire file into a single string
`aString = input.read(N)`	Read next N bytes (one or more) into a string
`aString = input.readline( )`	Read next line (including end-of-line marker) into a string
`aList = input.readlines( )`	Read entire file into list of line strings
`output.write(aString)`	Write a string of bytes into file
`output.writelines(aList)`	Write all line strings in a list into file
`output.close( )`	Manual close (done for you when file is collected)
`outout.flush( )`	Flush output buffer to disk without closing
`anyFile.seek(N)`	Change file position to offset N for next operation

이진 파일 다루기

struct 모듈을 이용하여 pack 및 unpack 함으로써 이진 데이터를 다룸

```
>>> F = open('data.bin', 'wb')                    # 파일은 이진 쓰기 모드로
>>> import struct
>>> data = struct.pack('>i4sh', 7, 'spam', 8)     # Little-endian, 정수, 문자열(4), 2-byte 정수
>>> data
'\x00\x00\x00\x07spam\x00\x08'
>>> F.write(data)                                 # 파일에 쓰기
>>> F.close()

>>> F = open('data.bin', 'rb')                    # 읽기 모드로 이진 파일을 열어서
>>> data = F.read()                               # 전체를 읽어들이면
>>> data                                          # 이진 데이터가 입력되고
'\x00\x00\x00\x07spam\x00\x08'
>>> values = struct.unpack('>i4sh', data)         # 다시 동일한 포맷을 가정하여 unpack
>>> values                                        # 결과는 순서쌍
(7, 'spam', 8)
```

struct 참고

Character	Byte order	Size	Alignment
@	native	native	native
=	native	standard	none
<	little-endian	standard	none
>	big-endian	standard	none
!	network (= big-endian)	standard	none

Format	C Type	Python type	Standard size	Notes
x	pad byte	no value		
c	char	string of length 1	1	
b	signed char	integer	1	(3)
B	unsigned char	integer	1	(3)
?	_Bool	bool	1	(1)
h	short	integer	2	(3)
H	unsigned short	integer	2	(3)
i	int	integer	4	(3)
I	unsigned int	integer	4	(3)
l	long	integer	4	(3)
L	unsigned long	integer	4	(3)
q	long long	integer	8	(2), (3)
Q	unsigned long long	integer	8	(2), (3)
f	float	float	4	(4)
d	double	float	8	(4)
s	char[]	string		
p	char[]	string		
P	void *	integer		(5), (3)

Python 객체들을 파일에 저장 - pickle

```
>>> D = {'a': 1, 'b': 2}
>>> F = open('datafile.pkl', 'wb')          # 파일은 이진 쓰기 모드로
>>> import pickle
>>> pickle.dump(D, F)
>>> F.close()

>>> F = open('datafile.pkl', 'rb')
>>> E = pickle.load(F)
>>> E
{'a': 1, 'b': 2}
```

```
>>> open('datafile.pkl', 'rb').read()
"(dp0\nS'a'\np1\nI1\nsS'b'\np2\nI2\ns."
```

```
>>> a = open('datafile.pkl', 'rb').read()
>>> print a
(dp0
S'a'
p1
I1
sS'b'
p2
I2
s.
```

Python 객체 타입의 정리

- 같은 분류에 속하는 객체들은 동일한 (유사한) 연산의 대상이 됨
  - 예) 문자열, 리스트, 순서쌍은 병합, 길이 파악, 인덱싱, 슬라이싱 등의 연산을 지원
- “Mutable” 객체들에 한하여 생성 이후에 변경이 가능
  - 수치형, 문자열, 순서쌍은 한번 생성된 이후에는 변경이 불가능하다.

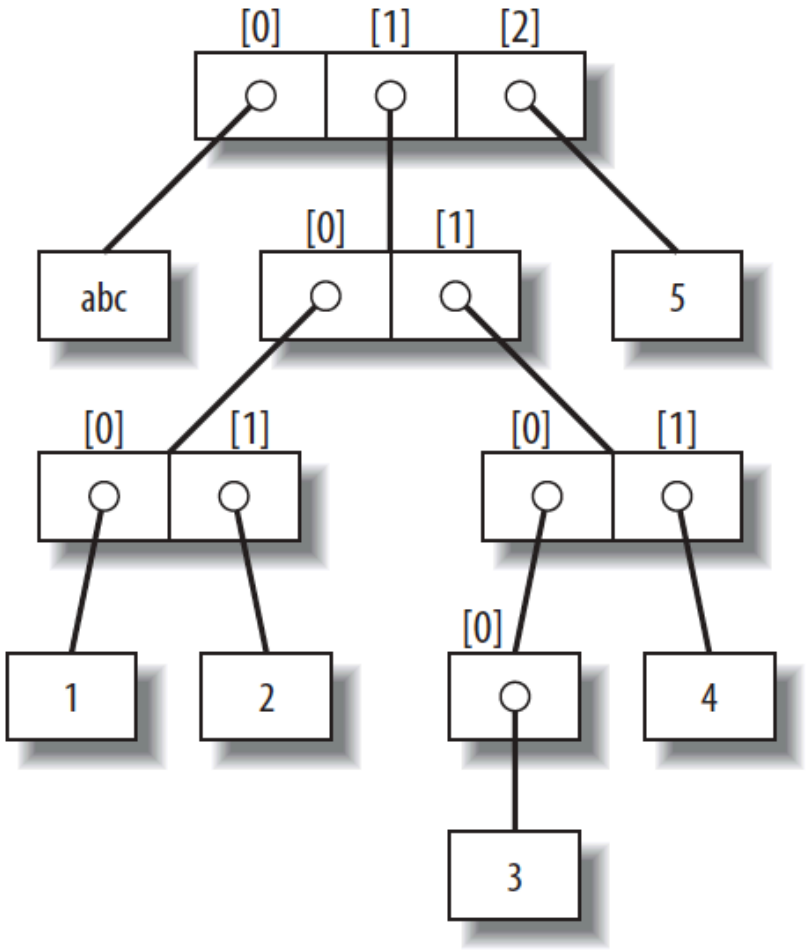
Object type	Category	Mutable?
Numbers	Numeric	No
Strings	Sequence	No
Lists	Sequence	Yes
Dictionaries	Mapping	Yes
Tuples	Sequence	No
Files	Extension	N/A

Python 객체의 구성

- 리스트, 사전, 순서쌍은 어떠한 종류의 객체도 포함 가능
- 리스트, 사전, 순서쌍은 임의의 수준까지 중첩 가능
- 리스트와 사전은 동적으로 자라나고 줄어들 수 있음

오른쪽 트리에서 각 노드의 객체 타입은?

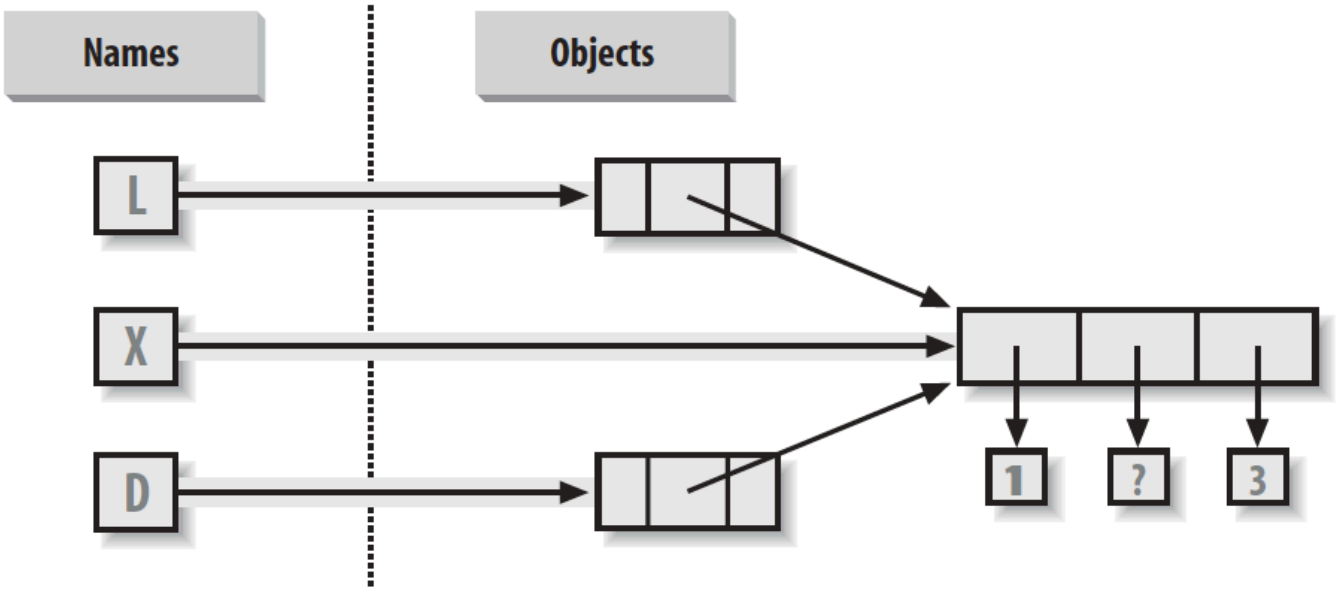
L = ['abc', [(1, 2), ([3], 4)], 5]



공유된 참조

```
>>> X = [1, 2, 3]
>>> L = ['a', X, 'b']
>>> D = {'x': X, 'y': 2}

>>> X[1] = 'surprise'
>>> L
['a', [1, 'surprise', 3], 'b']
>>> D
{'y': 2, 'x': [1, 'surprise', 3]}
```



동치 테스트 (Equality Tests)

- 연산자 == 는 값이 동일한지를 판단
  - 양변의 표현식이 계산됨
- 연산자 is 는 참조되는 객체가 동일한지를 판단
  - 양변이 참조하는 객체가 같은 객체인지

```
>>> L1 = [1, ('a', 3)]
>>> L2 = [1, ('a', 3)]
>>> L1 == L2, L1 is L2
(True, False)

>>> S1 = 'spam'
>>> S2 = 'spam'
>>> S1 == S2, S1 is S2
(True, True)

>>> S1 = 'a longer string'
>>> S2 = 'a longer string'
>>> S1 == S2, S1 is S2
(True, False)
```

비교 연산 (Comparisons)

- 수치형은 (타입에 관계 없이) 크기에 따라 비교
  - 복소수 (complex) 타입은 대소 관계가 정의되지 않으므로 예외
- 문자열은 사전적 (lexicographical) 순서에 따라 비교
- 리스트와 순서쌍은 왼쪽부터 우선으로 각 원소를 비교
- 사전은 (key, value) 쌍을 key 에 따라 정렬하여 동일한지를 비교
  - 대소 관계는 정의되지 않음
- 수치형이 아닌 타입이 섞여 있는 비교 (e.g. 1 < 'spam') 는 허용되지 않음

```
>> L1 = [1, ('a', 3)]
>>> L2 = [1, ('a', 2)]
>>> L1 < L2, L1 == L2, L1 > L2
(False, False, True)
```

객체 타입에 따른 참과 거짓 (True and False)

- 수치형 객체는 0 이 아니면 참
- 그 외의 객체는 비어 있지 않으면 (nonempty) 참

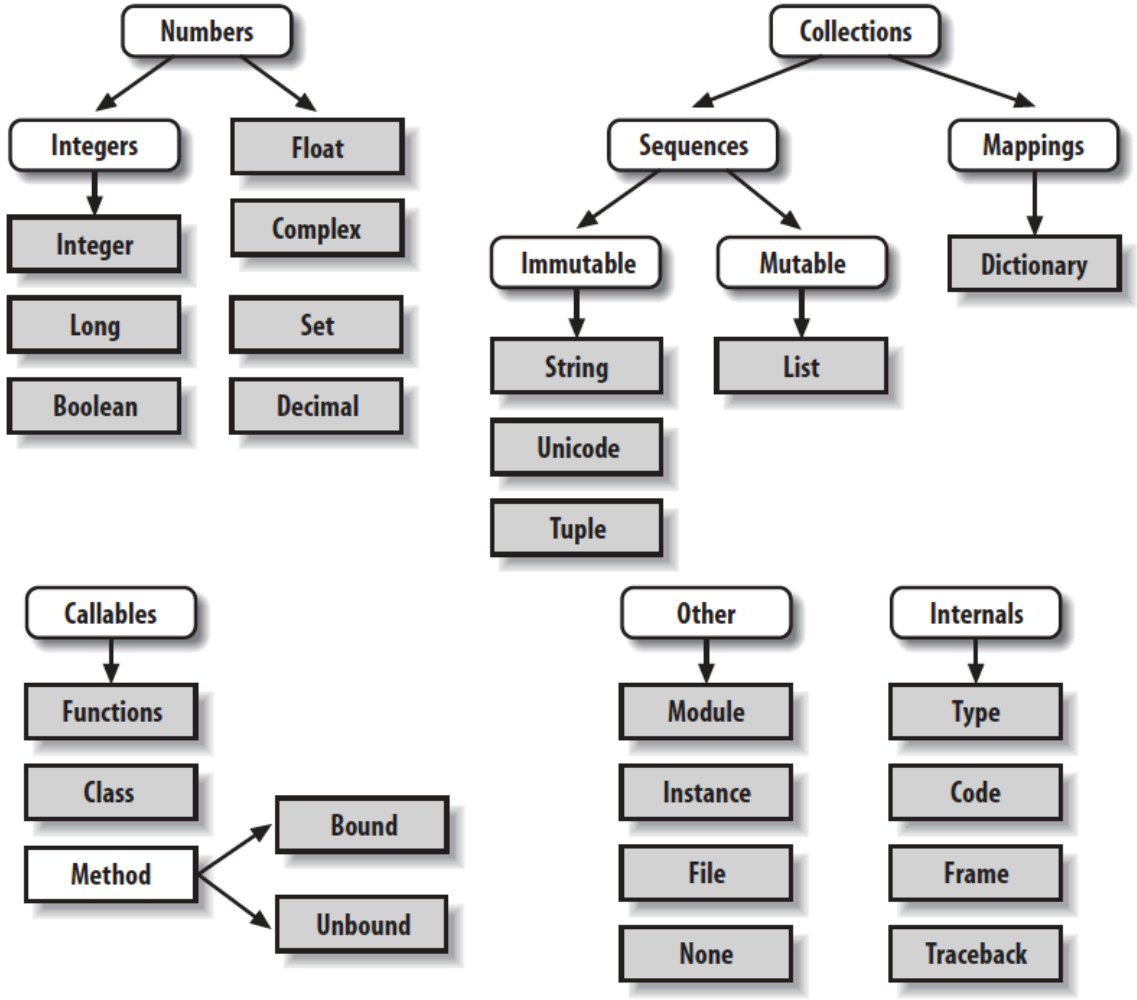
```
>>> bool(4)
True
>>> bool('spam')
True
>>> bool(0.0)
False
>>> bool(0.1)
True
>>> bool({})
False
>>> bool([])
False
```

Object	Value
"spam"	True
""	False
[]	False
{}	False
1	True
0.0	False
None	False

Type 객체

type(X) 는 객체 X 의 타입을 리턴
: 타입 자체도 객체!
The type of an object is an object of type **type**.

```
>>> type(1)
<type 'int'>
>>> type(1.0)
<type 'float'>
>>> type([1])
<type 'list'>
>>> type({})
<type 'dict'>
>>> type(())
<type 'tuple'>
>>> isinstance([1], list)
True
```



주의!

- 리스트 반복

```
>>> L = [4, 5, 6]
>>> X = L * 4
>>> Y = [L] * 4

>>> X
[4, 5, 6, 4, 5, 6, 4, 5, 6, 4, 5, 6]
>>> Y
[[4, 5, 6], [4, 5, 6], [4, 5, 6], [4, 5, 6]]

>>> L[1] = 0
>>> X
[4, 5, 6, 4, 5, 6, 4, 5, 6, 4, 5, 6]
>>> Y
[[4, 0, 6], [4, 0, 6], [4, 0, 6], [4, 0, 6]]
```

- 싸이클을 가지는 객체

```
>>> L = ['grail']
>>> L.append(L)
>>> L
['grail', [...]]
```

Quiz

- T = (4, 5, 6) 일 때, T == (1, 5, 6) 이 되게 하려면 어떤 Python 문장을 써야 하는가?
  - T = (1, 5, 6) 은 제외
  - 순서쌍은 생성 후 변경될 수 없음을 주의

- L = [1, 2, 3] 일 때, 다음의 결과는? 그 이유는?
  - X = L; X[1] = 0; print(L)
  - X = L[:]; X[1] = 0; print(L)

- 순서쌍과 리스트가 둘 다 존재하는 이유는?

Exercise

(1) 준비 - 앞 절에서 작성한 Score 클래스의 constructor 를 수정하여 인스턴스가 생성될 때 제목을 입력받을 수 있도록 해보자.

- s = Score('Summer Class')

(2) printScore() 메서드를 수정하여 title 을 맨 위에 출력하도록 해보자.

- 아래 예제처럼 출력
  - Title: Summer Class
  - John: 87
  - Mary: 92
  - Peter: 63

```
class Score:

    def __init__(self):
        self.scores = {}

    def putScore(self, name, score):
        self.scores[name] = score

    def printScore(self):
        names = list(self.scores.keys())
        for name in sorted(names):
            print('%s\t%s' % (name, self.scores[name]))

    def getScore(self, name):
        return self.scores.get(name, -1)
```

Answers

score.py

```
class Score:

    def __init__(self, title):
        self.scores = {}
        self.title = title

    def putScore(self, name, score):
        self.scores[name] = score

    def printScore(self):
        print('title:\t%s' % self.title)
        names = list(self.scores.keys())
        for name in sorted(names):
            print('%s\t%s' % (name, self.scores[name]))

    def getScore(self, name):
        return self.scores.get(name, -1)
```

test.py

```
from score import Score

s = Score('Summer Class')
s.putScore('Mary', 92)
s.putScore('Peter', 63)
s.putScore('John', 87)

s.printScore()
```

Exercise

(1) 앞의 Score 클래스에 메서드를 추가하여 보자.

- 메서드: saveScore(self, filename)
  - 현재까지 기록된 점수 데이터를 파일에 저장한다.
  - 저장할 파일 이름은 인자로 주어짐
- 메서드: loadScore(self, filename)
  - 파일에 저장된 점수 데이터를 읽어온다.
  - 읽어올 파일 이름은 인자로 주어짐

힌트: pickle 모듈을 이용할 것!

Answers

score.py 에 메서드 추가

```
def saveScore(self, filename):
        S = {'title': self.title,
             'scores': self.scores}
        F = open(filename, 'wb')
        pickle.dump(S, F)
        F.close()

def loadScore(self, filename):
        F = open(filename, 'rb')
        S = pickle.load(F)
        self.title = S['title']
        self.scores = S['scores']
```

test.py

```
from score import Score

s = Score('Summer Class')
s.putScore('Mary', 92)
s.putScore('Peter', 63)
s.putScore('John', 87)

s.printScore()

s.saveScore('scores.pkl')
```

Python interactive session

```
% python test.py
title:      Summer Class
John        87
Mary        92
Peter       63
% python
>>> from score import Score
>>> d = Score('')
>>> d.loadScore('scores.pkl')
>>> d. printScore()
title:      Summer Class
John        87
Mary        92
Peter       63
```

Q & A