

Python Programming



Python 기본자료구조

리스트
튜플
딕셔너리
연습문제

Department of Computer Engineering, Hanbat University
sykim@hanbat.ac.kr

내용

1. 리스트

- 생성 및 추가
- 인덱싱 (indexing)
- 슬라이싱 (slicing)
- 연접,복제
- 일부요소교체 (replacement),삭제 (delete),삽입 (insert),Clear

2. 튜플

- 정의 2가지 방법
- 한원소
- 인덱싱
- 슬라이싱

3. 사전형 (딕셔너리)

- 정의,추가,참조,
- 일괄대입,원소삭제
- 키-값 구하기

4. 연습문제



파이썬의 기본자료구조

- 파이썬의 기본 자료형 정수형, 실수형, 문자열을 배웠습니다. 이번 장에서는 정수형, 실수형, 문자열 등의 데이터가 여러 개 있을 때 이를 효과적으로 관리하는 데 사용되는 자료구조에 대해 배우겠습니다.
- 파이썬에서 가장 중요하게 이용되는 시퀀스(sequence) 자료형으로 리스트(list), 튜플(tuple), 사전(dict)이 제공됩니다.
 - 리스트(list)
 - 원소를 []안에 나열하는 자료구조의 데이터형이다.
 - 신청과목=['영어' , '수학']
 - 튜플(tuple)
 - 원소를 ()안에 나열하는 자료구조의 데이터형이다. 선언 후 변경불가
 - 신청과목=('영어' , '수학')
 - 사전형(dictionary)
 - (키,값)의 원소를 {}안에 나열하는 자료구조의 데이터형이다.
 - 과목별점수= {'영어':70, '수학':80}

1.리스트(list)

- 수강과목 중에서 관심있는 과목이 ‘데이터로 표현하는 세상’ , ‘영어’ , ‘수학’ 이다. 과목명을 문자열로 표현하고 변수에 저장해봅시다.

```
>>> 과목1 = '데이터로 표현하는 세상'
>>> 과목2 = '영어'
>>> 과목3 = '수학'
```

- 이와 같이 과목별로 변수명을 정하여 저장할 수 있습니다. 그러나 과목이 10개라면 이방법은 불편해 보입니다. 이러한 문제를 편리하게 처리하기 위하여 파이썬에서는 리스트(list)라는 기본자료구조를 제공합니다.
- 상기 3과목들은 세과목으로 리스트 상수를 만들고 리스트형 변수 “과목”에 대입(선언,정의)하고 정수인덱싱하여 사용한다.

```
>>> 과목 = ['데이터로 표현하는 세상' , '영어' , '수학' ] #리스트형변수(과목) 대입연산자(=) 과목리스트상수(['데이터로 표현하는 세상' , '영어' , '수학' ])
>>> 과목[0]          #첫(0)번째과목 참조
'데이터로 표현하는 세상'
>>> 과목[1]          #1번째 과목 참조
'영어'
>>>
```

1.리스트(list) (cont.)

- 리스트의 생성 및 요소추가

```
>>> s=['math',80,70,75.0] #리스트 생성(선언)
>>> s
['math',80,70,75.0]
>>> s=[] #공백리스트
>>> s
[]
>>> s.append('math' ) #요소추가
>>> s
['math']
>>> s.append(80) #요소추가
>>> s
['math',80]
```

인덱스	0	1	2	3
리스트 s=	'math'	80	70	75.0
역인덱스	-4	-3	-2	-1

- 리스트이 indexing(참조), slicing(쪼개기)

```
>>> s=['math',80,70,75.0] #리스트 생성(선언)
>>> s[0] #리스트인덱싱 0번째요소 참조
'math'
>>> s[1] #리스트인덱싱 1번째요소 참조
80
>>> s[-1] #리스트인덱싱 -1(뒤에서 1번째)요소 참조
75.0
>>> s[-2] #리스트인덱싱 -1(뒤에서 1번째)요소 참조
70
```

1.리스트(list) (cont.)

- 리스트의 슬라이싱 (slicing), 연결, 복제

```
>>> s=['math',80,70,75.0] #리스트 생성(선언)
>>> s[0:2] #0번째부터 2번째의 바로 전까지의 부분 리스트 slicing
['math',80]
>>> s[1:-1] #앞과 뒤의 한 요소만 제거한 부분리스트
[80,70]
>>> s[1:] #1번째부터 마지막까지의 부분리스트
[80,70,75.0]
>>> s+s #리스트 연결, 리스트 복제
['math',80,70,75.0, 'math',80,70,75.0]
>>> s+[1000,20] #리스트 연결
['math',80,70,75.0,1000,20]
>>> s[:2]+'english',90] #리스트 slicing, 연결
['math',80, 'english',90]
>>> 2*s[:2] #리스트 slicing, 복제
['math',80, 'math' ,80]
```

인덱스	0	1	2	3
리스트 s=	'math',	80,	70,	75.0]
역인덱스	-4	-3	-2	-1

1.리스트(list) (cont.)

- 부분리스트 교체,부분리스트제거,리스트의 삽입,리스트 clear,max,min

```
>>> s=['math',80,70,75.0]
>>> s[:2]=['eng',100] #Replace some items
>>> s
['eng', 100, 70, 75.0]
>>> s[0:2]=[] #Remove some items
>>> s
[70, 75.0]
>>> s[0:0]=['math',100] #Insert list
>>> s
['math', 100, 70, 75.0]
>>> s[:0]=s #Insert (a copy of) itself at the beginning
>>> s
['math', 100, 70, 75.0, 'math', 100, 70, 75.0]
>>> s[:]=[] #Clear the list: replace all items with empty list
>>> s
[]
```

인덱스	0	1	2	3
리스트 s=	'math'	80	70	75.0
역인덱스	-4	-3	-2	-1

```
>>> score=['math' ,80,70] #수학,중간,기말
>>> s
['math', 80, 70]
>>> s[1]=90 #중간시험 교체
>>> s
['math', 90, 70]
>>> s[1:1]=[10,20] #중간시험 출석 및 과제점수 [10,20]삽입
>>> s
['math', 10, 20, 90, 70]
>>>
```

```
>>> score=[70,80,90,85] # 4회 시험
>>> max(score) # 최고점수
90
>>> min(score) # 최저점수
70
>>> [max(score),min(score)] # 최고,최저]
[90, 70]
>>> max(score);min(score) # 한 줄에 여러명령
90
70
>>>
```

1.리스트(list) (cont.)

- 리스트의 생성, indexing, slicing,복제
- 일부교체,일부제거,삽입,자신삽입,Clear

```
>>> s=['math',80,70,75.0] #리스트 생성(선언)
>>> s[0] #리스트 인덱싱
'math'
>>> s[1] #리스트 인덱싱
80
>>> s[-2] #리스트 역인덱싱
70
>>> s[0:2] #slicing
['math', 80]
```

인덱스	0	1	2	3
리스트 s=['math'	80,	70,	75.0]
역인덱스	-4	-3	-2	-1

```
>>> s[1:-1]
[80, 70]
>>> s[1:] #slicing
[80, 70, 75.0]
>>> s[:2]+['english',90] #slicing
['math', 80, 'english', 90]
>>> 2*s[:2] #리스트 복제
['math', 80, 'math', 80]
```

```
>>> s=['math',80,70,75.0]
>>> s[:2]=['eng',100] #Replace some items
>>> s
['eng', 100, 70, 75.0]
>>> s[0:2]=[] #Remove some items
>>> s
[70, 75.0]
>>> s[0:0]=['math',100] #Insert list
>>> s
['math', 100, 70, 75.0]
>>> s[:0]=s #Insert (a copy of) itself at the beginning
>>> s
['math', 100, 70, 75.0, 'math', 100, 70, 75.0]
>>> s[:]=[] #Clear the list: replace all items with empty
list
>>> s
[]
```


2. 튜플(tuple)

- 튜플은 원소를 () 안에 나열하는 자료구조이다.
- 튜플은 일단 변수를 만들고 나면 새 원소를 추가하거나 기존의 원소를 제거하는 것은 불가능하지만, 그 외의 성질은 리스트와 같다.

```
>>> a=(1,3,4,-5) #Define a tuple
>>> a
(1, 3, 4, -5)
>>> a=1,3,4,-5 #앞줄과 같은 tuple 정의
>>> a
(1, 3, 4, -5)
>>> b=(11,) #한 원소를 갖는 튜플정의
>>> b
(11,)
>>> b=(11) #정수형
>>> b
11
>>>
```

```
>>> a=(1,3,4,-5) #Define a tuple
>>> a[0]
1
>>> a[0]=3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support
item assignment
>>> a[:2] #slice tuple
(1, 3)
>>> a(:2)
File "<stdin>", line 1
  a(:2)
    ^
SyntaxError: invalid syntax
>>>
```

3. 사전형(dict)

- 사전형 (딕셔너리)의 상수는 키(key)와 값(value)이라는 것을 쌍이다.
- 사전형 상수 삽입 • 정의, 타입체크, (key,value) 삽입, 참조

```
>>> price = {}           #공백 사전형변수 정의
>>> type(price)         #변수 price 의 타입 체크,
<class 'dict' >        # 'dict' 는 dictionary의 사전형의 이름이다.
>>> price['samsung']=42000 # (key,value)=(samsung,42000)을 추가
>>> price                #사전(형변수) price에 저장된 내용확인
{' Samsung ': 42000}
>>> price[ ' Daum KAKAO ']=80000 #( 'Daum KAKAO' ,80000) 추가
>>> price                #사전 price에 저장된 내용확인
{'samsung': 42000, 'Daum KAKAO': 80000}
>>> price['samsung']   #사전 price에서 key= 'samsung' 으로 값은 참조(검색)한다.
42000
>>> price['Daum KAKAO']
80000
```

3. 사전형(dict) (cont.)

- 딕셔너리 삭제
 - 일괄 입력, 원소삭제

```
>>> price[0]          #숫자로 참조 불허 반드시 키로 참조하여야한다.
Traceback (most recent call last):
  File "<stdin> ", line 1, in <module>
KeyError: 0
>>> #사전형변수에 키값을 일괄대입
>>> price = { ' Daum KAKAO ' : 80000, ' naver ' :800000, ' daeshin ' :30000}
>>> price           # 사전 price의 모든 요소(key,value)를 읽어 오기
{'Daum KAKAO': 80000, 'naver': 800000, 'daeshin': 30000}
>>> del price['naver' ] #사전 price에서 키 naver의 요소(naver,80000)을 삭제
>>> price           # 사전 price의 모든 요소(key,value)를 읽어 오기
{'Daum KAKAO': 80000, 'daeshin': 30000}
>>>
```

key	values
naver	80000
Daum KAKAO	800000
daeshin	30000

key	values
Daum KAKAO	800000
daeshin	30000

3. 사전형(dict) (cont.)

- 사전(형 변수)에서 키-값 읽어오기

```
>>> price = {'Daum KAKAO': 80000, 'naver':800000, 'daeshin':30000}
>>> price
{'Daum KAKAO': 80000, 'naver': 800000, 'daeshin': 30000}
>>> price.keys()      #사전 price의 모든 key 요소를 읽어오기
dict_keys(['Daum KAKAO', 'naver', 'daeshin'])
>>> list(price.keys()) #사전 price의 key 요소를 리스트 형으로 변환
['Daum KAKAO', 'naver', 'daeshin']
>>> list(price.values()) # 사전 price의 값 요소를 리스트 형으로 변환
[80000, 800000, 30000]
>>> 'naver' in price.keys() #naver가 사전 price의 키에 있으므로 참
True
>>> 'Naver' in price.keys() #Naver가 사전 price의 키가 키에 없으므로 거짓
False
>>> price['naver']
800000
```

key	values
naver	80000
Daum KAKAO	800000
daeshin	30000

```
>>> price['Naver']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Naver'
>>>
```

4. 연습문제

날짜	요일	종가
08/26	금	43,600
08/27	목	44,050
08/28	수	44,150
08/29	화	43,400
08/30	월	44,000

1. 2019년 8월 말의 삼성전자 종가에 대하여, 8/26일의 종가를 리스트의 첫 번째 항목으로 입력해서 price라는 이름의 리스트를 만들어보세요.
2. 리스트 price를 이용해 해당 주에 종가를 기준으로 가장 높았던 가격을 출력하세요. (힌트: 리스트에서 최댓값을 찾는 함수는 `max()`이고, 화면에 출력하는 함수는 `print()`입니다.)
3. 리스트 price를 이용해 해당 주에 종가를 기준으로 가장 낮았던 가격을 출력하세요. (힌트: 리스트에서 최솟값을 찾는 함수는 `min()`이고, 화면에 출력하는 함수는 `print()`입니다.)
4. 리스트 price를 이용해 해당 주에서 가장 종가가 높았던 요일과 가장 종가가 낮았던 요일의 가격 차를 화면에 출력하세요.
5. 리스트 price를 이용해 수요일의 종가를 화면에 출력하세요.
6. 날짜와 종가를 키와 값으로 구성되는 사전(dict) price2라는 만드세요.
7. 사전 price2 를 이용해 08/30일의 종가를 출력하세요.