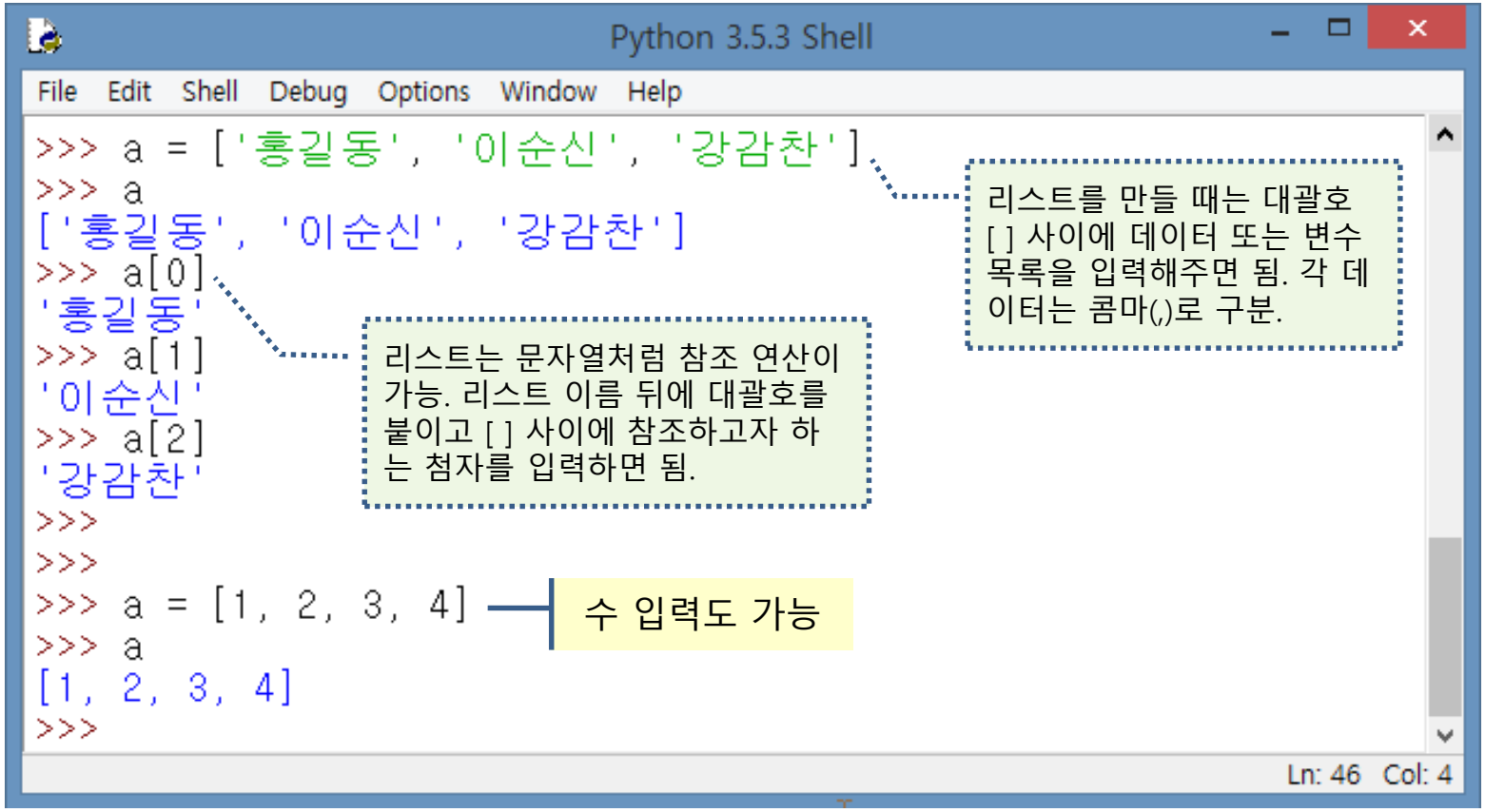
리스트 / 튜플 / 딕셔너리

리스트(List)

- 데이터의 목록을 다루는 자료형
- 리스트를 만들 때는 대괄호 [ ] 사용



리스트(List)

- 슬라이싱이 가능

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> a[0:5]
[1, 2, 3, 4, 5]
>>> a[5:]
[6, 7, 8, 9, 10]
>>> a[:3]
[1, 2, 3]
```

- +연산자를 통한 리스트간의 결합도 가능

```
>>> a = [1, 2, 3, 4]
>>> b = [5, 6, 7]
>>> a + b
[1, 2, 3, 4, 5, 6, 7]
```

- 리스트 내의 특정 위치에 있는 데이터를 변경하려면 참조 연산을 이용

```
>>> a = [1, 2, 3, 4, 5]
>>> a[2] = 30
>>> a
[1, 2, 30, 4, 5]
>>> a[3] = 40
>>> a
[1, 2, 30, 40, 5]
```

- 리스트의 길이

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

리스트 메서드(1)

```
>>> # append() : 리스트의 끝에 새 요소를 추가
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
>>>
>>> # extend() : 기존 리스트에 다른 리스트를 이어 붙임(+연산자와 같은 기능).
>>> a = [1, 2, 3]
>>> a.extend([4, 5, 6])
>>> a
[1, 2, 3, 4, 5, 6]
>>>
>>>
>>> # insert() : 첨자로 명시한 리스트 내의 위치에 새 요소를 삽입.
>>> #            insert(첨자, 데이터)의 형식으로 사용.
>>> a = [2, 4, 5]
>>> a.insert(0, 1) # 0 위치(첫번째)에 데이터 1 삽입.
>>> a
[1, 2, 4, 5]
>>> a.insert(2, 3) # 2 위치(세번째)에 데이터 3 삽입.
>>> a
[1, 2, 3, 4, 5]
>>>
>>>
>>> # remove() : 매개변수로 입력한 데이터를 리스트에서 찾아 발견한 첫번째 요소를 제거.
>>> a = ['BMW', 'BENZ', 'VOLKSWAGEN', 'AUDI']
>>> a.remove('BMW')
>>> a
['BENZ', 'VOLKSWAGEN', 'AUDI']
```

리스트 메서드(2)

```
>>> # pop() : 리스트의 마지막 요소를 뽑아 리스트에서 제거.
>>> a = [1, 2, 3, 4, 5]
>>> a.pop()
5
>>> a
[1, 2, 3, 4]
>>> a.pop()
4
>>> a
[1, 2, 3]
>>> #  한편, 마지막이 아닌 특정 요소를 제거하고 싶을 때에는 pop() 메서드에 제거하고자
>>> #  하는 요소의 인덱스를 입력.
>>> a = [1, 2, 3, 4, 5]
>>> a.pop(2)     # 3번째 요소 제거
3
>>> a
[1, 2, 4, 5]
>>>
>>>
>>>
>>> # index() : 리스트 내에서 매개변수로 입력한 데이터와 일치하는 첫 번째 요소의
>>> #           첨자를 알려줌.
>>> #           찾고자하는 데이터와 일치하는 요소가 없으면 오류 발생.
>>> a = ['abc', 'def', 'ghi']
>>> a.index('def')
1
>>> a.index('jkl')
Traceback (most recent call last):
  File "<pyshell#186>", line 1, in <module>
    a.index('jkl')
ValueError: 'jkl' is not in list
```

리스트 메서드(3)

```
>>> # count() : 매개변수로 입력한 데이터와 일치하는 요소의 개수 반환.
>>> a = [1, 100, 2, 100, 3, 100]
>>> a.count(100)
3
>>> a.count(200)
0
>>>
>>>
>>>
>>> # sort() : 리스트 내의 요소를 정렬.
>>> #          매개변수로 reverse=True를 입력하면 내림차순,
>>> #          아무것도 입력하지 않으면 오름차순으로 정렬.
>>> #  - 키워드 매개변수 : reverse=True와 같이 이름을 명시하여 사용하는 매개변수
>>> a = [3, 4, 5, 1, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4, 5]
>>> a.sort(reverse = True)
>>> a
[5, 4, 3, 2, 1]
>>>
>>>
>>>
>>> # reverse() : 리스트 내 요소의 순서를 반대로 뒤집는다.
>>> a = [3, 4, 5, 1, 2]
>>> a.reverse()
>>> a
[2, 1, 5, 4, 3]
>>> b = ['안', '녕', '하', '세', '요']
>>> b.reverse()
>>> b
['요', '세', '하', '녕', '안']
```

튜플(Tuple)

- 기본적인 용법 : ( ) 를 이용해서 생성.
- 새로운 요소를 추가하거나 삽입할 수 없고, 기존 요소를 삭제할 수 없다.
- 소프트웨어의 성능을 향상하는 데 도움.

```
>>> a = (1, 2, 3)
>>> a
(1, 2, 3)
>>> type(a)
<class 'tuple'>
>>>
>>> # 괄호 생략 가능.
>>> a = 1, 2, 3, 4
>>> a
(1, 2, 3, 4)
>>> type(a)
<class 'tuple'>
```

튜플(Tuple)

- 요소가 하나인 튜플 정의하기
  - 요소가 하나뿐인 튜플을 정의할 때는 요소 뒤에 콤마(,)를 반드시 넣어줘야 함. 그렇지 않으면 정수로 받아들임.

```
>>> a = (1)
>>> a
1
>>> type(a)
<class 'int'>
```

  - 올바른 정의

```
>>> a = (1,)
>>> a
(1,)
>>> type(a)
<class 'tuple'>
>>> b = 1,
>>> b
(1,)
>>> type(b)
<class 'tuple'>
```

튜플(Tuple)

- 슬라이싱이 가능

```
>>> a = (1, 2, 3, 4, 5, 6)
>>> a[:3]
(1, 2, 3)
>>> a[4:6]
(5, 6)
```

- +연산자를 통한 튜플간의 결합도 가능

```
>>> a = (1, 2, 3)
>>> b = (4, 5, 6)
>>> c = a + b
>>> a
(1, 2, 3)
>>> b
(4, 5, 6)
>>> c
(1, 2, 3, 4, 5, 6)
```

- 참조 연산은 가능하지만, 참조 연산을 이용한 요소 변경은 허용되지 않음.

```
>>> a = (1, 2, 3)
>>> a[0]
1
>>> a[0] = 7
Traceback (most recent call last):
  File "<pyshell#294>", line 1, in <module>
    a[0] = 7
TypeError: 'tuple' object does not support item assignment
```

- 리스트의 길이

```
>>> a = (1, 2, 3)
>>> len(a)
3
```

튜플 메서드

```
>>> # index() : 매개변수로 입력한 데이터와 일치하는 튜플 내 요소의 첨자를 알려줌.
>>> #           찾고자 하는 데이터와 일치하는 요소가 없으면 오류 발생.
>>> a = ('abc', 'def', 'ghi')
>>> a.index('def')
1
>>> a.index('jkl')
Traceback (most recent call last):
  File "<pyshell#312>", line 1, in <module>
    a.index('jkl')
ValueError: tuple.index(x): x not in tuple
>>>
>>>
>>>
>>> # count() : 매개변수로 입력한 데이터와 일치하는 요소 개수 반환.
>>> a = (1, 100, 2, 100, 3, 100)
>>> a.count(100)
3
>>> a.count(200)
0
```

딕셔너리(Dictionary)

- 기본적인 용법 : { } 를 이용해서 생성.
- 리스트처럼 첨자를 이용해서 요소에 접근하고, 변경할 수 있음.
- 요소에 접근할 때 0부터 시작하는 수 첨자 뿐 아니라, 문자열과 숫자를 비롯해서 변경이 불가능한 형식이면 어떤 자료형이든 사용 가능.
- 첨자는 키(key)라 하고, 이 키가 가리키는 슬롯에 저장되는 데이터를 일컬어 값(value)라 함.
- 딕셔너리는 키-값의 쌍으로 구성.
- 탐색 속도가 빠르고, 사용하기도 편함.
- 새로운 키-값을 입력하거나 딕셔너리 안에 있는 요소를 참조할 때는 리스트와 튜플에서 처럼 대괄호 [ ]를 이용.

딕셔너리(Dictionary)

- 사용 예

```
>>> dic = {}
>>> dic['파이썬'] = 'www.python.org'
>>> dic['마이크로소프트'] = 'www.microsoft.com'
>>> dic['애플'] = 'www.apple.com'
>>> dic['파이썬']
'www.python.org'
>>> dic['마이크로소프트']
'www.microsoft.com'
>>> dic['애플']
'www.apple.com'
>>> type(dic)
<class 'dict'>
```

- 딕셔너리 출력

```
>>> dic
{'파이썬': 'www.python.org', '마이크로소프트': 'www.microsoft.com', '애플': 'www.apple.com'}
>>>
```

딕셔너리 메서드

```
>>> # keys() : 키의 목록 출력
>>> dic.keys()
dict_keys(['파이썬', '마이크로소프트', '애플'])
>>>
>>>
>>> # values() : 값의 목록 출력
>>> dic.values()
dict_values(['www.python.org', 'www.microsoft.com', 'www.apple.com'])
>>>
>>>
>>>
>>>
>>> # items() : 키와 값의 쌍으로 이루어진 전체 목록 반환
>>> dic.items()
dict_items([('파이썬', 'www.python.org'), ('마이크로소프트', 'www.microsoft.com'), ('애플',
'www.apple.com')])
>>>
```