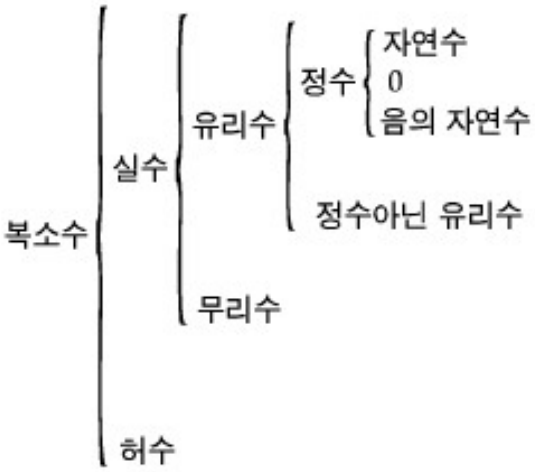
Python

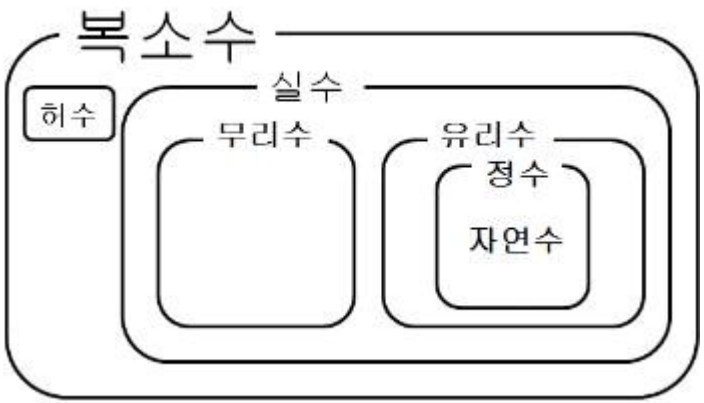
DataType

숫자형/문자열 자료형
리스트 자료형
튜플 자료형/딕셔너리 자료형
집합자료형과 그외

숫자형(Number)

항목	사용 예
정수	123, -345, 0
실수	123.45, 1234.5, 3.4e10
복소수	1+2j, -3j
8진수	0o34, 0o25
16진수	0x2A, 0xFF





정수형/실수형/8진수/16진수

In [6]:
```
# 정수형
a = 123
a = -123
a = 0
```

In [7]:
```
# 실수형
b = 1.2
b = -3.45
```

In [8]:
```
# 4.24E10 = 4.24*10^10승, 4.24e-10 = 4.24*10^-10승(e와 E 둘 중 어느 것을 사용해도 무방)
c = 4.24E10
c = 4.24e10
```

In [9]:
```
# 8진수(숫자 0 + 알파벳 소문자 o 또는 대문자 O)
d = 0o177
```

In [10]:
```
# 16진수(0x로 시작)
e = 0x8ff
f = 0xABC
```

복소수

In [11]:
```
# 복소수(파이썬에서는 i대신 j를 사용)
a = 1+2j
b = 3-4J
```

In [12]:
```
# 복소수 내장함수
# 복소수.real - 복소수의 실수부분 리턴
a = 1+2j
a.real
```

Out[12]: 1.0

In [15]:
```
# 복소수.imag - 복소수의 허수부분 리턴
a.imag
```

Out[15]: 2.0

In [16]:
```
# 복소수.conjugate() - 복소수의 켤레복소수 리턴
a.conjugate()
```

Out[16]: (1-2j)

In [17]:
```
# abs(복소수) - 복소수의 절대값(1+2j의 절대값은 √(1²+2²) 이다.)
abs(a)
```

Out[17]: 2.23606797749979

사칙연산

In [18]:
```
1 # 사칙연산
2 a = 3
3 b = 4
4 a + b
```

Out[18]: 7

In [19]:
```
1 a * b
```

Out[19]: 12

In [20]:
```
1 a / b
```

Out[20]: 0.75

In [21]:
```
1 a ** b  # 제곱
```

Out[21]: 81

사칙연산

In [22]:
```
7/4
```

Out[22]: 1.75

In [23]:
```
# 나눗셈 후 소수점 아랫자리를 버리는 연산자
7//4
```

Out[23]: 1

In [24]:
```
# 나머지값 반환
7%3
```

Out[24]: 1

문자열 자료형

In [3]:
```
# 큰 따옴표(")로 양쪽 둘러싸기
"Hello Workd"
```

Out[3]: 'Hello Workd'

In [4]:
```
# 작은 따옴표(')로 양쪽 둘러싸기
'Python is fun'
```

Out[4]: 'Python is fun'

In [5]:
```
# 큰 따옴표 3개를 연속(""")로 양쪽 둘러싸기
"""Life is to short, You need python"""
```

Out[5]: 'Life is to short, You need python'

In [6]:
```
# 작은 따옴표 3개를 연속(''')로 양쪽 둘러싸기
'''Life is to short, You need python'''
```

Out[6]: 'Life is to short, You need python'

문자열 자료형

In [20]:
```
# 문자열에 작은 따옴표(')포함시키기
food = "Python's favorite food is perl"
print(food)
```

```
Python's favorite food is perl
```

In [21]:
```
food = 'Python's favorite food is perl'
```

```
  File "<ipython-input-21-d60d52314e29>", line 1
    food = 'Python's favorite food is perl'
                   ^
SyntaxError: invalid syntax
```

In [22]:
```
# 문자열에 큰 따옴표(")포함시키기
say='"Python is very easy." he says.!'
print(say)
```

```
"Python is very easy." he says.!
```

In [23]:
```
say=''Python is very easy.' he says.!'
```

```
  File "<ipython-input-23-ad05afed2e19>", line 1
    say=''Python is very easy.' he says.!'
                ^
SyntaxError: invalid syntax
```

In [24]:
```
# ₩(백슬래시)를 이용해서 작은 따옴표(')를 문자열에 포함시키기
food = 'Python₩'s favorite food is perl'
print(food)
```

```
Python's favorite food is perl
```

In [25]:
```
# ₩(백슬래시)를 이용해서 큰따옴표(")를 문자열에 포함시키기
say = "₩"Python is very easy.₩" he says.!"
print(say)
```

```
"Python is very easy." he says.!
```

여러 줄인 문자열 변수에 대입

In [5]:
```
# 줄을 바꾸기 위한 이스케이프 코드 '₩n' 삽입하기
multiline = "Life is to short₩nYou need python"
print(multiline)
```

```
Life is to short
You need python
```

In [6]:
```
# 연속된 작은 따옴표 3개(''') 이용
multiline = '''
    Life is to short
    You nedd python
'''
print(multiline)
```

```
Life is to short
You nedd python
```

In [7]:
```
# 연속된 큰 따옴표 3개(""") 이용
multiline = """
    Life is to short
    You nedd python
"""
print(multiline)
```

```
Life is to short
You nedd python
```

문자열 연산하기

In [5]:
```
# 문자열 더해서 연결하기(concatenation)
head = "Python"
tail = " is fun!"
# content = head + tail
# print(content)
print(head + tail)
```

```
Python is fun!
```

In [4]:
```
# 문자열 곱하기
a = "Python"
print(a * 2)
```

```
PythonPython
```

In [6]:
```
# 문자열 곱하기
print("=" * 50)
print("My Program")
print("=" * 50)
```

```
==================================================
My Program
==================================================
```

문자열 인덱싱


```
In [8]: a = "Life is too short, You need Python"
        a[3]
Out[8]: 'e'

In [9]: a[0]
Out[9]: 'L'

In [10]: a[12]
Out[10]: 's'

In [11]: a[-1]  # 뒤에서 첫번째가 되는 문자
Out[11]: 'n'

In [12]: a[-0]  # 0과 -0은 같은 것
Out[12]: 'L'

In [13]: a[-2]
Out[13]: 'o'

In [14]: a[-5]
Out[14]: 'y'
```

문자열 슬라이싱

In [16]:
```
# 'Life' 단어를 뽑아 내는 방법
a = "Life is too short, You need Python"
b = a[0] + a[1] + a[2] + a[3]
print(b)
```

```
Life
```

In [17]:
```
# a[시작번호:끝번호]를 지정하면 끝 번호에 해당하는 것은 포함되지 않는다.
a[0:4]
```

Out[17]: 'Life'

In [18]:
```
a[0:3]
```

Out[18]: 'Lif'

In [19]:
```
a[0:5]   # 공백문자도 같은 문자와 동일하게 취급
```

Out[19]: 'Life '

In [21]:
```
a[5:7]
```

Out[21]: 'is'

In [22]:
```
a[12:17]
```

Out[22]: 'short'

문자열 슬라이싱


```
In [23]: a[19:] # 끝 번호 부분을 생략하면 시작번호부터 그 문자열의 끝까지
Out[23]: 'You need Python'

In [24]: a[:17]  # 시작번호를 생략하면 문자열의 처음부터 끝번호까지
Out[24]: 'Life is too short'

In [25]: a[:] # 시작번호와 끝 번호를 생략하면 문자열의 처음부터 끝까지
Out[25]: 'Life is too short, You need Python'

In [27]: a[19:-7]  # a[19] 부터 a[-8]까지, a[-7]은 포함하지 않는다.
Out[27]: 'You need'
```

슬라이싱으로 문자열 나누기

In [28]:
```
a = "20181210Rainy"
date = a[:8]
weather = a[8:]
date
```

Out[28]: '20181210'

In [30]:
```
weather
```

Out[30]: 'Rainy'

In [34]:
```
# "20181210Rainy" 문자열을 세 부분으로 나누는 방법
year = a[:4]
day = a[4:8]
weather = a[8:]
print(year)
print(day)
print(weather)
```

```
2018
1210
Rainy
```

'Pithon' 문자열을 'Python'으로 바꾸려면?

In [35]:
```
# 'Pithon' 문자열을 'Python'으로 바꾸려면?
a = "Pithon"
print(a[1])
a[1] = 'y'   # 문자열, 튜플 등의 자료형은 그 요소값을 변경할 수 없다.
```

```
i

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-35-60dee05f070c> in <module>()
      1 a = "Pithon"
      2 print(a[1])
----> 3 a[1] = 'y'

TypeError: 'str' object does not support item assignment
```

In [43]:
```
# 문자열, 튜플 등의 자료형은 그 요소값을 변경할 수 없다.
# 슬라이싱 기법 이용
a = "Pithon"
print(a[:1])
print(a[2:])
print(a[:1] + 'y' + a[2:])
```

```
P
thon
Python
```

문자열 포매팅

In [1]:
```
"I eat %d apples." % 3   # 숫자 바로 대입
```
Out[1]: 'I eat 3 apples.'

In [3]:
```
"I eat %s apples." % "five"   # 문자열 바로 대입
```
Out[3]: 'I eat five apples.'

In [4]:
```
number = 3
"I eat %d apples." % number   #  숫자값을 나타내는 변수로 대입
```
Out[4]: 'I eat 3 apples.'

In [14]:
```
"rate is %f" %3.234
```
Out[14]: 'rate is 3.234000'

In [15]:
```
"rate is %s" %3.234  # 문자로 인식
```
Out[15]: 'rate is 3.234'

In [11]:
```
# 2개 이상 값 넣기
number = 10
day = "three"
"I eat %d apples. so I was sick for %s days. " % (number, day)
```
Out[11]: 'I eat 10 apples. so I was sick for three days. '

"Error is 98%."를 출력하려면?

In [12]:
```
# "Error is 98%."를 출력하려면?
"Error is %d%." %98
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-ebd82f3835fa> in <module>()
      1 # "Error is 98%."를 출력하려면?
----> 2 "Error is %d%." %98

ValueError: incomplete format
```

In [13]:
```
"Error is %d%%." %98
```

Out[13]:
```
'Error is 98%.'
```

정렬과 공백


```
In [16]: # 정렬과 공백
         "%10s" % "hi"  # 오른쪽 정렬하고 앞의 나머지는 공백
Out[16]: '        hi'

In [17]: "%-10s jain" % "hi"   # hi가 왼쪽 정렬
Out[17]: 'hi         jain'

In [18]: # 소수점 표현하기
         "%0.4f" %3.42134234   # 소수점 4자리까지
Out[18]: '3.4213'

In [19]: "%10.4f" %3.42134234   # 소수점 4자리까지 표시하고 전체 길이가 10개인 문자열 공간에서 오른쪽으로 정렬
Out[19]: '    3.4213'
```

문자열 관련 함수들


```
In [20]: # 문자 갯수 세기(count) - 'b'의 갯수
         a = "hobby"
         a.count('b')
Out[20]: 2

In [21]: # 위치 알려주기 - 'b'가 처음 나온 위치
         a = "Python is best choice"
         a.find('b')
Out[21]: 10

In [22]: a.find('k')   # 찾는 문자열이 존재하지 않는다면 -1 을 반환
Out[22]: -1

In [23]: a = "Life is to short"
         a.index('t')
Out[23]: 8

In [24]: a.index('k') # 'k'가 존재하지 않는 경우
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-cb6c8ec35229> in <module>()
----> 1 a.index('k')

ValueError: substring not found
```

find와 index의 다른점은 문자열에 존재하지 않는 문자를 찾으면 오류가 발생

문자열 관련 함수들

In [25]:
```
# 문자열 삽입(join)
a = ","
a.join('abcd')
```

Out[25]: 'a,b,c,d'

In [26]:
```
a = "hi"
a.upper()  # 소문자를 대문자로(upper)
```

Out[26]: 'HI'

In [27]:
```
a = "HI"
a.lower()  # 대문자를 소문자로(lower)
```

Out[27]: 'hi'

In [28]:
```
a = " hi"
a.lstrip() # 왼쪽 공백지우기(lstrip)
```

Out[28]: 'hi'

In [30]:
```
a = " hi "
a.rstrip()  # 오른쪽 공백지우기(rstrip)
```

Out[30]: ' hi'

In [31]:
```
a = " hi "
a.strip()  # 양쪽 공백지우기(strip)
```

Out[31]: 'hi'

문자열 관련 함수들

In [33]:
```
# 문자열 바꾸기(replace)
a = "Life is too short"
a.replace("Life","Your leg")
```

Out[33]:
```
'Your leg is too short'
```

In [34]:
```
a.split()  # 공백을 기준으로 문자열 나눔(split)
```

Out[34]:
```
['Life', 'is', 'too', 'short']
```

In [35]:
```
a = "a:b:c:d"
a.split(':')
```

Out[35]:
```
['a', 'b', 'c', 'd']
```

고급 문자열 포매팅

In [1]:
```
"I eat {0} apples".format(3)  # 숫자 바로 대입하기
```
Out[1]: 'I eat 3 apples'

In [2]:
```
"I eat {0} apples".format("five")  # 문자열 바로 대입하기
```
Out[2]: 'I eat five apples'

In [3]:
```
number = 3
"I eat {0} apples".format(number)  # 변수 대입하기
```
Out[3]: 'I eat 3 apples'

In [4]:
```
# 두개 이상 값 넣기
number = 10
day = "three"
"I ate {0} apples. so I was sick for {1} days.".format(number, day)
```
Out[4]: 'I ate 10 apples. so I was sick for three days.'

In [5]:
```
# 이름으로 넣기
"I ate {number} apples. so I was sick for {day} days.".format(number=10, day=3)
```
Out[5]: 'I ate 10 apples. so I was sick for 3 days.'

In [6]:
```
# 인덱스와 이름을 혼용해서 넣기
"I ate {0} apples. so I was sick for {day} days.".format(10, day=3)
```
Out[6]: 'I ate 10 apples. so I was sick for 3 days.'

고급 문자열 포매팅


```
In [7]: "{0:<10}".format("hi") # 왼쪽 정렬(총 자리수 10개)
Out[7]: 'hi        '

In [8]: "{0:>10}".format("hi") # 오른쪽 정렬(총 자리수 10개)
Out[8]: '        hi'

In [9]: "{0:^10}".format("hi") # 가운데 정렬(총 자리수 10개)
Out[9]: '    hi    '

In [10]: "{0:=^10}".format("hi") # 가운데 정렬하고 빈 공간을 "=" 문자로 채운다.
Out[10]: '====hi===='

In [11]: "{0:!<10}".format("hi") # 왼쪽 정렬하고 빈 공간을 "!" 문자로 채운다.
Out[11]: 'hi!!!!!!!!'

In [12]: y = 3.42134234
         "{0:0.4f}".format(y)   # 소수점 4자리
Out[12]: '3.4213'

In [13]: "{0:10.4f}".format(y)    # 전체 자리수 10자리이고 소수점 4자리
Out[13]: '    3.4213'

In [14]: "{{and}}".format()   # '{' 또는 '}' 문자 표현하기
Out[14]: '{and}'
```

숫자형/문자열 자료형
리스트 자료형
튜플 자료형/딕셔너리 자료형
집합자료형과 그외

리스트 인덱싱

In [1]:
```
a = []       # 빈 리스트
b = [1,2,3]  # 숫자
c = ['Life','is','too','short']  # 문자
d = [1,2,'Life','is']     # 숫자와 문자열
e = [1,2, ['Life', 'is']] # 리스트 자체
```

In [2]:
```
a = [1,2,3]
a
```

Out[2]: [1, 2, 3]

In [3]:
```
a[0]
```

Out[3]: 1

In [4]:
```
a[0] + a[2]   # 1 + 3
```

Out[4]: 4

In [5]:
```
a[-1]
```

Out[5]: 3

리스트 인덱싱

In [6]:
```
a = [1,2,3,['a','b','c']]
a[0]
```

Out[6]: 1

In [7]:
```
a[-1]
```

Out[7]: ['a', 'b', 'c']

In [8]:
```
a[3]
```

Out[8]: ['a', 'b', 'c']

In [9]:
```
a[-1][0]  # ['a', 'b', 'c']의 첫번째 요소
```

Out[9]: 'a'

In [10]:
```
a[-1][1]
```

Out[10]: 'b'

In [13]:
```
a = [1,2,['a','b',['Life', 'is']]]
a[2][2][0]
```

Out[13]: 'Life'

리스트 슬라이싱

In [14]:
```
a = [1,2,3,4,5]
a[0:2]
```

Out[14]: [1, 2]

In [16]:
```
b = a[:2]
b
```

Out[16]: [1, 2]

In [17]:
```
c = a[2:]
c
```

Out[17]: [3, 4, 5]

In [18]:
```
a = [1,2,3,['a','b','c'],4,5]
a[2:5]
```

Out[18]: [3, ['a', 'b', 'c'], 4]

In [19]:
```
a[3][:2]
```

Out[19]: ['a', 'b']

리스트 연산자

In [20]:
```
a = [1,2,3]
b = [4,5,6]
a + b
```

Out[20]:
```
[1, 2, 3, 4, 5, 6]
```

In [21]:
```
a * 3
```

Out[21]:
```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

In [22]:
```
a[2] + "hi"   # 숫자와 문자는 형 오류(TypeError) 발생
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-22-cbafcaae950e> in <module>()
----> 1 a[2] + "hi"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [23]:
```
str(a[2]) + "hi"   # 정수나 실수를 문자열의 형태로 변환하는 내장함수
```

Out[23]:
```
'3hi'
```

리스트 수정, 변경과 삭제


```
In [41]: a = [1,2,3]
         a[2] = 4
         a
Out[41]: [1, 2, 4]

In [42]: a[1:2]
Out[42]: [2]

In [43]: a[1:2] = ['a','b','c']
         a
Out[43]: [1, 'a', 'b', 'c', 4]

In [44]: a[1] = ['a','b','c']
         a
Out[44]: [1, ['a', 'b', 'c'], 'b', 'c', 4]

In [45]: a[1:2] = ['a']
         a
Out[45]: [1, 'a', 'b', 'c', 4]

In [46]: a[1:3] = []  # 삭제
         a
Out[46]: [1, 'c', 4]

In [47]: del a[1] # del함수 이용
         a
Out[47]: [1, 4]
```

유의사항 : a[1:2] = ['a','b','c'] 와 a[1] = ['a','b','c']은 **다른 결과값**
a[1:2] = ['a','b','c'] : a[1]에서 a[2]사이의 리스트를 ['a', 'b', 'c']로 바꾼다
a[1] = ['a','b','c'] : a의 두 번째 요소를 ['a', 'b', 'c']로 바꾼다

리스트 관련 함수들

In [2]:
```
a = [1,2,3]
a.append(4)
a
```

Out[2]: [1, 2, 3, 4]

In [3]:
```
a.append([5,6])
a
```

Out[3]: [1, 2, 3, 4, [5, 6]]

In [5]:
```
a = [1,4,3,2]
a.sort()
a
```

Out[5]: [1, 2, 3, 4]

In [8]:
```
a = ['a','c','b']
a.sort()
a
```

Out[8]: ['a', 'b', 'c']

In [9]:
```
a = ['a','c','b']
a.reverse()
a
```

Out[9]: ['b', 'c', 'a']

리스트 관련 함수들

In [10]:
```
a = [1,2,3]
a.index(3)
```

Out[10]: 2

In [11]:
```
a.index(1)
```

Out[11]: 0

In [12]:
```
a.index(0)  # 0 값은 a리스트에 존재하지 않으므로 ValueError 발생
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-affdffa288f1> in <module>()
----> 1 a.index(0)

ValueError: 0 is not in list
```

In [14]:
```
a = [1,2,3]
a.insert(0,4)  # a[0]위치에 4 삽입
a
```

Out[14]: [4, 1, 2, 3]

In [15]:
```
a.insert(3,5)  # a[3]위치에 5 삽입
a
```

Out[15]: [4, 1, 2, 5, 3]

In [19]:
```
a = [1,2,4,5,3,1,2,3]
a.remove(3)  # 3을 삭제(첫번째 3만 삭제)
a
```

Out[19]: [1, 2, 4, 5, 1, 2, 3]

리스트 관련 함수들

In [21]:
```
a = [1,2,3]
a.pop()
a
```

Out[21]: [1, 2]

In [22]:
```
a = [1,2,3]
a.pop(1)  # a[1]을 리턴하고 값을 삭제
```

Out[22]: 2

In [23]:
```
a
```

Out[23]: [1, 3]

In [24]:
```
a = [1,2,3,1]
a.count(1)
```

Out[24]: 2

In [25]:
```
a = [1,2,3]
a.extend([4,5])   # a 리스트에 더한다.
a
```

Out[25]: [1, 2, 3, 4, 5]

In [26]:
```
b = [6,7]
a.extend(b)
a
```

Out[26]: [1, 2, 3, 4, 5, 6, 7]

숫자형/문자열 자료형
리스트 자료형
튜플 자료형/딕셔너리 자료형
집합자료형과 그외

튜플과 리스트

- 리스트와 거의 비슷
- 리스트는 [과]으로 둘러싸지만 튜플은 (과)으로 둘러싼다 .
- 튜플과 리스트의 가장 큰 차이는 값을 변화시킬 수 있는가 없는 가이다.
  - 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.
  - 리스트의 항목 값은 변화가 가능하고 튜플의 항목 값은 변화가 불가능하다.
- 실제 프로그램에서는 값이 변경되는 형태의 변수가 훨씬 많기 때문에 평균적으로 튜플보다는 리스트를 더 많이 사용한다.

In [1]:

```
# 튜플의 여러 유형
t1 = ()
t2 = (1,)
t3 = (1,2,3)
t4 = 1,2,3    # 괄호 생략해도 무방
t5 = ('a','b',('ab','cd'))
```

튜플의 삭제/변경시 오류

In [3]:
```
1 # 튜플 요소값 삭제시 오류(TypeError)
2 t1 = (1,2,'a','b')
3 del t1[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-5b9b422970ca> in <module>()
      2 # 튜플 요소값 삭제시 오류
      3 t1 = (1,2,'a','b')
----> 4 del t1[0]

TypeError: 'tuple' object doesn't support item deletion
```

In [4]:
```
1 # 튜플 요소값 변경시 오류(TypeError)
2 t1[0] = 'c'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-92e4992af21f> in <module>()
      1 # 튜플 요소값 변경시 오류
----> 2 t1[0] = 'c'

TypeError: 'tuple' object does not support item assignment
```

튜플의 연산

In [1]:
```
1 t1 = (1,2,'a','b')
2 t1[0]
```

Out[1]: 1

In [2]:
```
1 t1[3]
```

Out[2]: 'b'

In [3]:
```
1 t1[1:]
```

Out[3]: (2, 'a', 'b')

In [4]:
```
1 t2 = (3,4)
2 t1 + t2
```

Out[4]: (1, 2, 'a', 'b', 3, 4)

In [5]:
```
1 t2 * 3
```

Out[5]: (3, 4, 3, 4, 3, 4)

딕셔너리 자료형 - 추가/삭제

In [3]:
```
# 딕셔너리의 모습
# {Key1:Value1,Key2:Value2,Key3:Value3 ...}
dic = {'name':'pey', 'phone':'0119993333', 'birth':'1211'}
```

In [4]:
```
a = {1:'hi'}
```

In [5]:
```
a = {'a' : [1,2,3]}  # value 에 리스트도 가능
```

In [6]:
```
# 딕셔너리 쌍 추가하기
a = {1 : 'a'}
a[2] = 'b'  # {2:'b'}쌍 추가
a
```

Out[6]: {1: 'a', 2: 'b'}

In [8]:
```
a['name'] = 'pay' # {'name':'pay'}쌍 추가
a
```

Out[8]: {1: 'a', 2: 'b', 'name': 'pay'}

In [10]:
```
a[3] = [1,2,3]  # {3:[1,2,3]}쌍 추가
a
```

Out[10]: {1: 'a', 2: 'b', 'name': 'pay', 3: [1, 2, 3]}

In [11]:
```
del a[1]   # 삭제
a
```

Out[11]: {2: 'b', 'name': 'pay', 3: [1, 2, 3]}

딕셔너리 사용

In [27]:
```
grade = {'pay':10, 'julliet':99}
grade['pay']
```

Out[27]: 10

In [28]:
```
grade['julliet']
```

Out[28]: 99

In [29]:
```
a = {1:'a',2:'b'}
a[1]
```

Out[29]: 'a'

In [30]:
```
a[2]
```

Out[30]: 'b'

In [31]:
```
dic = {'name':'pey', 'phone':'0119993333', 'birth':'1211'}
dic['name']
```

Out[31]: 'pey'

In [32]:
```
dic['phone']
```

Out[32]: '0119993333'

In [33]:
```
dic['birth']
```

Out[33]: '1211'

딕셔너리 만들 때 주의사항

In [35]:
```
1 # 중복되는 Key사용 금지. 어떤 Value를 불러야 할지 알 수 없다.
2 a = {1:'a', 1:'b'}
3 a
```

Out[35]:
```
{1: 'b'}
```

In [36]:
```
1 a = {[1,2] : 'hi'}  # Key로 리스트는 쓸 수 없다.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-36-bcf766ba5a2c> in <module>()
----> 1 a = {[1,2] : 'hi'}

TypeError: unhashable type: 'list'
```

In [39]:
```
1 a = {(1,2) : 'hi'} # Key로 튜플은 쓸 수 있다. 즉, key는 변하는 값인지 변하지 않는 값인지에 달려 있다.
2 a
```

Out[39]:
```
{(1, 2): 'hi'}
```

딕셔너리 함수

In [40]:
```
# Key 리스트 만들기(keys)
a = {'name':'pey', 'phone':'0119993333', 'birth':'1211'}
a.keys()
```

Out[40]:
```
dict_keys(['name', 'phone', 'birth'])
```

In [41]:
```
# dict_keys객체 사용
for k in a.keys() :
    print(k)
```

```
name
phone
birth
```

In [42]:
```
# dict_keys객체는 리스트 고유의 함수인 append, insert, pop, remove, sort등의 함수를 수행할 수 없다.
# dict_keys객체를 리스트로 변환
list(a.keys())
```

Out[42]:
```
['name', 'phone', 'birth']
```

In [43]:
```
# Value 리스트 만들기(values)
a.values()
```

Out[43]:
```
dict_values(['pey', '0119993333', '1211'])
```

In [44]:
```
# Key, Value쌍 얻기(items)
a.items()
```

Out[44]:
```
dict_items([('name', 'pey'), ('phone', '0119993333'), ('birth', '1211')])
```

In [45]:
```
# Key, Value쌍 모두 지우기(clear)
a.clear()
a
```

Out[45]:
```
{}
```

딕셔너리 함수

In [46]:
```
# Key로 Value얻기(get)
a = {'name':'pey', 'phone':'0119993333', 'birth':'1211'}
a.get('name')
```

Out[46]: 'pey'

In [47]:
```
a.get('phone')
```

Out[47]: '0119993333'

In [49]:
```
a.get('nokey')  # None을 리턴함
a['nokey']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-49-97b651f080c1> in <module>()
      1 a.get('nokey')
----> 2 a['nokey']

KeyError: 'nokey'
```

In [50]:
```
a.get('foo','bar')
```

Out[50]: 'bar'

In [51]:
```
a = {'name':'pey', 'phone':'0119993333', 'birth':'1211'}
'name' in a
```

Out[51]: True

In [52]:
```
'email' in a
```

Out[52]: False

숫자형/문자열 자료형
리스트 자료형
튜플 자료형/딕셔너리 자료형
집합자료형과 그외

집합자료형

In [1]:
```
# 집합자료형은 set키워드를 이용
s1 = set([1,2,3])
s1
```

Out[1]: {1, 2, 3}

In [7]:
```
# set 2가지 큰 특징 : 1. 중복을 허용하지 않는다. 2. 순서가 없다.
s2 = set("Hello World")
s2
```

Out[7]: {' ', 'H', 'W', 'd', 'e', 'l', 'o', 'r'}

In [9]:
```
# 리스트로 변환
s1 = set([1,2,3])
i1 = list(s1)
i1
```

Out[9]: [1, 2, 3]

In [10]:
```
i1[0]
```

Out[10]: 1

In [11]:
```
# 튜플로 변환
t1 = tuple(s1)
t1
```

Out[11]: (1, 2, 3)

In [12]:
```
t1[0]
```

Out[12]: 1

교집합/합집합/차집합

In [13]:
```
s1 = set([1,2,3,4,5,6])
s2 = set([4,5,6,7,8,9])
# 교집합
s1 & s2
```

Out[13]: {4, 5, 6}

In [14]:
```
s1.intersection(s2)
```

Out[14]: {4, 5, 6}

In [15]:
```
# 합집합
s1 | s2
```

Out[15]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [16]:
```
s1.union(s2)
```

Out[16]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [17]:
```
# 차집합
s1 - s2
```

Out[17]: {1, 2, 3}

In [18]:
```
s1.difference(s2)
```

Out[18]: {1, 2, 3}

집합자료형 함수들

In [19]:
```
# 값 1개 추가하기(add)
s1 = set([1,2,3])
s1.add(4)
s1
```

Out[19]: {1, 2, 3, 4}

In [20]:
```
# 값 여러개 추가하기(update)
s1 = set([1,2,3])
s1.update([4,5,6])
s1
```

Out[20]: {1, 2, 3, 4, 5, 6}

In [21]:
```
# 특정값 제거하기(remove)
s1 = set([1,2,3])
s1.remove(2)
s1
```

Out[21]: {1, 3}

자료형의 참과 거짓

자료형	값	참 or 거짓
문자열	"python"	참
	""	거짓
리스트	[1, 2, 3]	참
	[]	거짓
튜플	()	거짓
딕셔너리	{}	거짓
숫자형	1	참
	0	거짓
	None	거짓

참과 거짓

In [3]:
```
a = [1,2,3,4]
while a:  # a가 참인 동안
    a.pop()
    print(a)
```

```
[1, 2, 3]
[1, 2]
[1]
[]
```

In [4]:
```
if []:
    print("True")
else:
    print("False")
```

```
False
```

In [5]:
```
if [1,2,3]:
    print("True")
else:
    print("False")
```

```
True
```

변수

In [6]:
```
# 파이썬의 모든 자료형은 객체.
a = 3  # 상수 3이 아닌 정수형 객체
type(3)
```

Out[6]: int

In [9]:
```
a = 3
b = 3
a is b  # a와 b가 동일한 객체를 가리키는지 판단
```

Out[9]: True

In [17]:
```
# 입력한 자료형에 대한 참조 갯수를 알려주는 함수
# 파이썬이 내부적으로 3이라는 자료형을 이미 사용했기 때문에 참조갯수가 많다.
import sys
sys.getrefcount(3)
```

Out[17]: 476

In [20]:
```
aa = 3
sys.getrefcount(3)
```

Out[20]: 477

In [21]:
```
bb = 3
sys.getrefcount(3)
```

Out[21]: 478

변수를 만드는 여러가지 방법

In [22]:
```
a, b = ('python','life')
```

In [23]:
```
(a, b) = 'python','life'   # 튜플은 괄호 생략 가능
```

In [24]:
```
[a, b] = ['python','life'] # 리스트로 변수 생성
```

In [25]:
```
a = b = 'python'   # 여러 개의 변수에 같은 값 대입
```

In [26]:
```
# 두 변수의 값 바꾸기
a = 3
b = 5
a, b = b, a
a
```

Out[26]: 5

In [27]:
```
b
```

Out[27]: 3

In [31]:
```
# 메모리에 생성된 변수 없애기
a = 3
b = 3
del(a)
del(b)
```

리스트를 변수에 넣고 복사

In [47]:
```
# 리스트를 변수에 넣고 복사하고자 할 때
a = [1,2,3]
b = a
a[1] = 4  # a리스트뿐만 아니라 b리스트도 바뀐다.
a
```

Out[47]: [1, 4, 3]

In [48]:
```
b   # a리스트뿐만 아니라 b리스트도 바뀐다.
```

Out[48]: [1, 4, 3]

In [49]:
```
# 다른 리스트를 가리키게하는 방법
# 1. [:]이용
a = [1,2,3]
b = a[:]  # a 리스트 전체를 복사하여 b에 대입
a[1] = 4
a
```

Out[49]: [1, 4, 3]

In [50]:
```
b
```

Out[50]: [1, 2, 3]

In [51]:
```
# 2.copy 모듈 이용
from copy import copy
b = copy(a)  # b = a[:]과 동일
b is a
```

Out[51]: False