

# PYTHON 자료형과 표현식

---

동의대학교 컴퓨터소프트웨어공학과  
이종민 교수

# 개요

- 강의 주제
  - 파이썬 프로그래밍 언어 전반: 자료 구조 위주
- 목차
  - 내장 자료 구조(list, tuple, set, dict 등)
  - 예외 처리
  - 람다 표현식
  - 정규 표현식
  - numpy

# 내장 자료형

- 숫자 자료형 (numeric types) – int, float, complex
- 순차 자료형 (sequence types) – list, tuple, range
- 텍스트 순차 자료형 (text sequence type) – str
- 이진 순차 자료형 (binary sequence type) – bytes, bytearray, memoryview
- 집합 자료형 (set types) – set, frozenset
- 사상 자료형 (mapping types) – dict

# 숫자 자료형

- 숫자 자료형: `int`, `float`, `complex`
- 지원 연산
  - `+`, `-`, `*`, `/`, `//`, `%`, 단항`+`, 단항`-`

<code>abs(x)</code>	absolute value or magnitude of <code>x</code>
<code>int(x)</code>	<code>x</code> converted to integer
<code>float(x)</code>	<code>x</code> converted to floating point
<code>complex(re, im)</code>	a complex number with real part <code>re</code> , imaginary part <code>im</code> . <code>im</code> defaults to zero.
<code>c.conjugate()</code>	conjugate of the complex number <code>c</code>
<code>divmod(x, y)</code>	the pair <code>(x // y, x % y)</code>
<code>pow(x, y)</code>	<code>x</code> to the power <code>y</code>
<code>x ** y</code>	<code>x</code> to the power <code>y</code>

# numbers 모듈

- 숫자 자료형
  - **int**: numbers.Integral 클래스  
(**bool**: int의 서브 클래스. truth testing procedure 사용하여 True/False로 변환)
  - **float**: numbers.Real 클래스
  - **complex**: numbers.Complex 클래스
- 참고: numbers 모듈 (PEP 3141 – A Type Hierarchy for Numbers)

```
class Number(metaclass=ABCMeta): pass

class Complex(Number): # 복소수(real, imag) 관련 연산 정의

class Real(Complex): # float 관련 연산 지원

class Rational(Real, Exact): # 분자, 분모

class Integral(Rational): # int 변환 & 비트-스트링 연산
```

- numbers.Real 자료형 (int, float)에서 지원하는 연산

Operation	Result
<code>math.trunc(x)</code>	<code>x</code> truncated to <code>Integral</code>
<code>round(x[, n])</code>	<code>x</code> rounded to <code>n</code> digits, rounding half to even. If <code>n</code> is omitted, it defaults to 0.
<code>math.floor(x)</code>	the greatest <code>Integral</code> <code>&lt;= x</code>
<code>math.ceil(x)</code>	the least <code>Integral</code> <code>&gt;= x</code>

# 순차 자료형

- 순차 (Sequences)
  - 0 이상 수에 의해 색인되는 유한 순서 집합(finite ordered set)
  - 불변 순차(immutable sequence)
    - **문자열 (Strings)**: 유니코드 코드 포인트를 나타내는 일련의 값
    - **튜플 (Tuples)**: 괄호 안에 쉼표로 임의의 파이썬 객체 구분. tuple() 또는 ()를 사용하여 생성.
    - **바이트 (Bytes)**:  $0 \leq x < 256$  사이의 정수로 표현되는 8비트 바이트. 불변 배열(immutable array)
  - 가변 순차(mutable sequence)
    - **리스트 (Lists)**: 대괄호 안에 쉼표로 임의의 파이썬 객체 구분. list() 또는 []를 사용하여 생성.
    - **바이트 배열 (Byte Arrays)**

```
>>> t = ()
>>> type(t)
<class 'tuple'>
>>>
>>> l = []
>>> type(l)
<class 'list'>
```

# 집합 자료형과 사상 자료형

- 집합 자료형 (Set types)
  - **집합 (Sets)**: 가변 집합. set() 생성자 사용하여 만들고 add() 사용하여 추가 가능.
  - **냉동 집합(Frozen sets)**: frozenset() 생성자 사용하여 만듦.

- 사상 자료형(Mappings)
  - **사전(Dictionaries)**
    - dict() 생성자 또는 {} 사용
    - 거의 모든 값을 색인으로 사용하는 객체의 유한 집합
    - 가변 객체

```
>>> m = {'김': '부산진구', '이': '해운대구'}
>>> m['박'] = '사상구'
>>> m['김']
'부산진구'
>>>
>>> m
{'김': '부산진구', '이': '해운대구', '박': '사상구'}
```

```
>>> s = set()
>>> type(s)
<class 'set'>
>>> s = {1, 2, 3}
>>> type(s)
<class 'set'>
>>>
>>> d = {}
>>> type(d)
<class 'dict'>
>>> d = {0: "로봇SW", 1: "Python"}
>>> type(d)
<class 'dict'>
```



# 순차 자료형

- 순차 자료형: **list, tuple, range**
- 공통 연산

Operation	Result
<code>x in s</code>	<code>True</code> if an item of <code>s</code> is equal to <code>x</code> , else <code>False</code>
<code>x not in s</code>	<code>False</code> if an item of <code>s</code> is equal to <code>x</code> , else <code>True</code>
<code>s + t</code>	the concatenation of <code>s</code> and <code>t</code>
<code>s * n</code> or <code>n * s</code>	equivalent to adding <code>s</code> to itself <code>n</code> times
<code>s[i]</code>	<code>i</code> th item of <code>s</code> , origin 0
<code>s[i:j]</code>	slice of <code>s</code> from <code>i</code> to <code>j</code>
<code>s[i:j:k]</code>	slice of <code>s</code> from <code>i</code> to <code>j</code> with step <code>k</code>
<code>len(s)</code>	length of <code>s</code>
<code>min(s)</code>	smallest item of <code>s</code>
<code>max(s)</code>	largest item of <code>s</code>
<code>s.index(x[, i[, j]])</code>	index of the first occurrence of <code>x</code> in <code>s</code> (at or after index <code>i</code> and before index <code>j</code> )
<code>s.count(x)</code>	total number of occurrences of <code>x</code> in <code>s</code>

# 가변 순차 자료형 연산

Operation	Result
<code>s[i] = x</code>	item <i>i</i> of <i>s</i> is replaced by <i>x</i>
<code>s[i:j] = t</code>	slice of <i>s</i> from <i>i</i> to <i>j</i> is replaced by the contents of the iterable <i>t</i>
<code>del s[i:j]</code>	same as <code>s[i:j] = []</code>
<code>s[i:j:k] = t</code>	the elements of <code>s[i:j:k]</code> are replaced by those of <i>t</i>
<code>del s[i:j:k]</code>	removes the elements of <code>s[i:j:k]</code> from the list
<code>s.append(x)</code>	appends <i>x</i> to the end of the sequence (same as <code>s[len(s):len(s)] = [x]</code> )
<code>s.clear()</code>	removes all items from <i>s</i> (same as <code>del s[:]</code> )
<code>s.copy()</code>	creates a shallow copy of <i>s</i> (same as <code>s[:]</code> )
<code>s.extend(t)</code> or <code>s += t</code>	extends <i>s</i> with the contents of <i>t</i> (for the most part the same as <code>s[len(s):len(s)] = t</code> )
<code>s *= n</code>	updates <i>s</i> with its contents repeated <i>n</i> times
<code>s.insert(i, x)</code>	inserts <i>x</i> into <i>s</i> at the index given by <i>i</i> (same as <code>s[i:i] = [x]</code> )
<code>s.pop([i])</code>	retrieves the item at <i>i</i> and also removes it from <i>s</i>
<code>s.remove(x)</code>	remove the first item from <i>s</i> where <code>s[i] == x</code>
<code>s.reverse()</code>	reverses the items of <i>s</i> in place

- 사용 예

```
>>> my_list = list(range(10))
>>> my_list
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> my_list[0] = 100
>>> my_list
[100, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> my_list[0:9:2]
[100, 2, 4, 6, 8]
>>> my_list[0:5:2]
[100, 2, 4]
>>> my_list[0:5:2] = [0,0,0]
>>> my_list
[0, 1, 0, 3, 0, 5, 6, 7, 8, 9]
>>> del my_list[0:4]
>>> my_list
[0, 5, 6, 7, 8, 9]
>>> # [0:4] --> 실제로는 0 ~ (4-1)번째에만 영향 미침
>>> my_list.append(100)
>>> my_list
[0, 5, 6, 7, 8, 9, 100]
>>> my_list.insert(1, 100)
>>> my_list
[0, 100, 5, 6, 7, 8, 9, 100]
>>> my_list.reverse()
>>> my_list
[100, 9, 8, 7, 6, 5, 100, 0]
```

# list 자료형

- 생성 방법
  - 빈 리스트를 나타내는 대괄호 사용: []
  - 대괄호를 쉼표로 구분: [a], [a,b,c] → ([a], [a,b,c]) 튜플 생성
  - 리스트 해석(list comprehension) 사용: [x for x in iterable]
  - list 생성자 사용: list() or list(iterable)

```
>>> s = []
>>> s
[]
>>> s.append(1)
>>> s
[1]
>>> s = [1], [1,2]
>>> s
([1], [1, 2])
>>>
>>> s = [x/2 for x in list(range(10))]
>>> s
[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5]
>>> s = list()
>>> s
[]
```

- 사용 가능한 메소드 <https://docs.python.org/3/tutorial/datastructures.html>

`list.append(x)`

Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

`list.extend(iterable)`

Extend the list by appending all the items from the iterable. Equivalent to `a[len(a):] = iterable`.

`list.insert(i, x)`

Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

`list.remove(x)`

Remove the first item from the list whose value is x. It is an error if there is no such item.

`list.pop([i])`

Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list. (The square brackets around the *i* in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

`list.clear()`

Remove all items from the list. Equivalent to `del a[:]`.

`list.index(x[, start[, end]])`

Return zero-based index in the list of the first item whose value is *x*. Raises a `ValueError` if there is no such item.

The optional arguments *start* and *end* are interpreted as in the slice notation and are used to limit the search to a particular subsequence of the list. The returned index is computed relative to the beginning of the full sequence rather than the *start* argument.

`list.count(x)`

Return the number of times `x` appears in the list.

`list.sort(key=None, reverse=False)`

Sort the items of the list in place (the arguments can be used for sort customization, see `sorted()` for their explanation).

`list.reverse()`

Reverse the elements of the list in place.

`list.copy()`

Return a shallow copy of the list. Equivalent to `a[:]`.

- 사용 예

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
0
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4)  # Find next banana starting a position 4
6
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
>>> fruits.sort()
>>> fruits
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
>>> fruits.pop()
'pear'
```



- 리스트를 스택처럼 사용하기

```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```

# collections.deque

- 리스트를 큐처럼 사용하기

```
>>> from collections import deque
>>> queue = deque(["Eric", "John", "Michael"])
>>> queue.append("Terry")           # Terry arrives
>>> queue.append("Graham")         # Graham arrives
>>> queue.popleft()                # The first to arrive now leaves
'Eric'
>>> queue.popleft()                # The second to arrive now leaves
'John'
>>> queue                          # Remaining queue in order of arrival
deque(['Michael', 'Terry', 'Graham'])
```

# 참고. collections 모듈

- 내장 컨테이너 외에 구현된 자료형 <https://docs.python.org/3/library/collections.html#collections.deque>

<code>namedtuple()</code>	factory function for creating tuple subclasses with named fields
<code>deque</code>	list-like container with fast appends and pops on either end
<code>ChainMap</code>	dict-like class for creating a single view of multiple mappings
<code>Counter</code>	dict subclass for counting hashable objects
<code>OrderedDict</code>	dict subclass that remembers the order entries were added
<code>defaultdict</code>	dict subclass that calls a factory function to supply missing values
<code>UserDict</code>	wrapper around dictionary objects for easier dict subclassing
<code>UserList</code>	wrapper around list objects for easier list subclassing
<code>UserString</code>	wrapper around string objects for easier string subclassing

# deque 사용 예

```
>>> from collections import deque
>>> d = deque('ghi')           # make a new deque with three items
>>> for elem in d:             # iterate over the deque's elements
...     print(elem.upper())
G
H
I

>>> d.append('j')              # add a new entry to the right side
>>> d.appendleft('f')          # add a new entry to the left side
>>> d                          # show the representation of the deque
deque(['f', 'g', 'h', 'i', 'j'])

>>> d.pop()                    # return and remove the rightmost item
'j'
>>> d.popleft()                # return and remove the leftmost item
'f'
>>> list(d)                    # list the contents of the deque
['g', 'h', 'i']
>>> d[0]                        # peek at leftmost item
'g'
>>> d[-1]                      # peek at rightmost item
'i'
```

```
>>> d = list('hello')
>>> d
['h', 'e', 'l', 'l', 'o']
>>> s = set('hello')
>>> s
{'e', 'l', 'h', 'o'}
>>> t = tuple('hello')
>>> t
('h', 'e', 'l', 'l', 'o')
```

# Counter() 사용 예

```
>>> # Tally occurrences of words in a list
>>> cnt = Counter()
>>> for word in ['red', 'blue', 'red', 'green', 'blue', 'blue']:
...     cnt[word] += 1
>>> cnt
Counter({'blue': 3, 'red': 2, 'green': 1})
```

- 특정 원소의 빈도 계산에 편리

```
>>> d = list('hello')
>>> d
['h', 'e', 'l', 'l', 'o']
>>> s = set('hello')
>>> s
{'e', 'l', 'h', 'o'}
>>> t = tuple('hello')
>>> t
('h', 'e', 'l', 'l', 'o')
>>>
>>>
>>>
>>> from collections import Counter
>>> c = Counter()
>>> for ch in d:
...     c[ch] += 1

>>> c
Counter({'l': 2, 'h': 1, 'e': 1, 'o': 1})
```

# 행렬 표현

- 리스트를 내포하여 임의의 행렬 표현 가능
- 3x4 행렬 예

```
>>> matrix = [  
...     [1, 2, 3, 4],  
...     [5, 6, 7, 8],  
...     [9, 10, 11, 12],  
... ]
```

- 전치 행렬 만들기 1

```
>>> [[row[i] for row in matrix] for i in range(4)]  
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

## • 전치 행렬 만들기 2

```
>>> transposed = []
>>> for i in range(4):
...     transposed.append([row[i] for row in matrix])
...
>>> transposed
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

## • 전치 행렬 만들기 3

```
>>> transposed = []
>>> for i in range(4):
...     # the following 3 lines implement the nested listcomp
...     transposed_row = []
...     for row in matrix:
...         transposed_row.append(row[i])
...     transposed.append(transposed_row)
...
>>> transposed
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

- 전치 행렬 만들기 4

- 앞의 결과와 다른 점: 내포된 리스트의 원소가 리스트가 아니라 튜플!

```
>>> list(zip(*matrix))  
[(1, 5, 9), (2, 6, 10), (3, 7, 11), (4, 8, 12)]
```

- 참고: zip(\*iterables)

- <https://docs.python.org/3/library/functions.html#zip>
- 각 반복자(iterable)의 원소를 차례대로 모아서 반복자를 만드는 함수
- 가장 적은 반복자의 원소 개수만큼 반복자 만들

```
def zip(*iterables):  
    # zip('ABCD', 'xy') --> Ax By  
    sentinel = object()  
    iterators = [iter(it) for it in iterables]  
    while iterators:  
        result = []  
        for it in iterators:  
            elem = next(it, sentinel)  
            if elem is sentinel:  
                return  
            result.append(elem)  
        yield tuple(result)
```



# tuple 자료형

- 튜플 생성 방법
  - 빈 튜플을 나타내는 괄호 사용: ()
  - 싱글턴 튜플 생성을 위해 쉼표를 뒤에 붙임: a, or (a,)
  - 쉼표로 항목 구분: a, b, c or (a, b, c)
  - tuple() 생성자 사용: tuple() or tuple(iterable)

```
>>> t = ()
>>> t
()
>>> t = 1, # or (1,)
>>> t
(1,)
>>> t = 1,2,3 # or (1,2,3)
>>> t
(1, 2, 3)
>>> t = tuple()
>>> t
()
>>> t = tuple(list(range(10)))
>>> t
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

- 사용 예

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
>>> # Tuples are immutable:
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> # but they can contain mutable objects:
... v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
```

# range 자료형

- 생성자

- range(stop)
- range(start, stop[, step])

- 사용 방법

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(0))
[]
>>> list(range(1, 0))
[]
```

```
>>> r = range(0, 20, 2)
>>> r
range(0, 20, 2)
>>> 11 in r
False
>>> 10 in r
True
>>> r.index(10)
5
>>> r[5]
10
>>> r[:5]
range(0, 10, 2)
>>> r[-1]
18
```

```
>>> r = range(10)
>>> r.start
0
>>> r.stop
10
>>> r.step
1
>>> type(r)
<class 'range'>
>>> l = list(r)
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5,10))
[5, 6, 7, 8, 9]
>>> list(range(1,10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# 텍스트 순차 자료형 str

- str 객체
  - 문자열을 다룸
  - Python 3.x에서 문자열은 유니코드 코드 포인트의 불변 순차임.
- 생성 방법
  - 작은 따옴표(single quotes) 이용: 'hello "world"'
  - 큰 따옴표(double quotes) 이용: "hello 'world'!"
  - 삼중 따옴표(triple quotes) 이용: """ or """
- 생성자

```
class str(object="")
```

```
class str(object=b", encoding='utf-8', errors='strict')
```

- 문자열 메소드 & 사용 방법
  - <https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>
  - <https://docs.python.org/3/library/text.html#textservices>

- TODO: 문자열 연산

- 앞 페이지의 문자열 관련 홈페이지 참조하여 여러 가지 문자열 연산을 해보시오.

- 주요 메소드

- str.count(sub[, start[, end]])
- str.find(sub[, start[, end]])
- str.format(\*args, \*\*kwargs)
- str.isalnum()
- str.isalpha()
- str.isdecimal()
- str.isdigit()
- str.isidentifier()
- str.islower()

- str.isnumeric()
- str.isupper()
- str.join(iterable)
- str.lower()
- str.lstrip([chars])
- str.replace(old, new[, count])
- str.rstrip([chars])
- str.split(sep=None, maxsplit=-1)
- str.splitlines([keepends])
- str.strip([chars])
- str.upper()

# 집합 자료형 set

- 집합 생성 방법
  - 중괄호 이용: {}
  - set() 생성자 이용
- 사용 가능한 연산
  - 합집합 '|', 교집합 '&', 차집합 '-', 부분 집합 '<=', 진부분 집합 '<', 포함 집합 '>=', 진포함 집합 '>'
  - 원소 개수 len(s), 원소 여부 x in s
  - 원소 관련 메소드

**add(*elem*)**Add element *elem* to the set.**remove(*elem*)**Remove element *elem* from the set. Raises `KeyError` if *elem* is not contained in the set.**discard(*elem*)**Remove element *elem* from the set if it is present.**pop()**Remove and return an arbitrary element from the set. Raises `KeyError` if the set is empty.**clear()**

Remove all elements from the set.

## • 사용 예

```
>>> s = {}
>>> s
{}
>>> s = set(range(10))
>>> s
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> s = set('Hello, world! 안녕하세요')
>>> s
{'r', 'l', '하', '세', '!', 'o', ' ', 'd', 'e', '녕', 'w', '요', 'H', '안', ','}
>>> s > set('hello')
False
>>> s > set('Hello')
True
>>> s < set('Hello')
False
```

```
>>> a = {1,2,3} ; b= {3,4,5,6}
>>> a | b # 합집합
{1, 2, 3, 4, 5, 6}
>>> a & b # 교집합
{3}
>>> a - b # 차집합
{1, 2}
>>> len(a)
3
>>> 2 in a
True
```

- 원소 추가/삭제 관련 사용 예

```
>>> a = {1,2,3}
```

```
>>> a.add(4)
```

```
>>> a
```

```
{1, 2, 3, 4}
```

```
>>> a.add({5,6})
```

```
Traceback (most recent call last):
```

```
File "<pyshell#74>", line 1, in <module>
```

```
    a.add({5,6})
```

```
TypeError: unhashable type: 'set'
```

```
>>> a.remove(5)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#75>", line 1, in <module>
```

```
    a.remove(5)
```

```
KeyError: 5
```

```
>>> a
```

```
{1, 2, 3, 4}
```

```
>>> a.remove(4)
```

```
>>> a
```

```
{1, 2, 3}
```

```
>>> a.discard(4)
```

```
>>> a
```

```
{1, 2, 3}
```

```
>>> a.discard(3)
```

```
>>> a
```

```
{1, 2}
```

```
>>> val = a.pop()
```

```
>>> val
```

```
1
```

```
>>> a
```

```
{2}
```

```
>>> a.clear()
```

```
>>> a
```

```
set()
```



# 사상 자료형 dict

- 사상(mapping) 객체
  - <Key, Value> 쌍으로 자료 관리하는 객체
  - 자바: java.util.HashMap<K,V>, C++: std::map<key, value>
- dict 생성 방법
  - 중괄호 이용: { (key:value)\* }
  - dict() 생성자 이용

```
class dict(**kwarg) ¶  
class dict(mapping, **kwarg)  
class dict(iterable, **kwarg)
```

# dict 주요 메소드

## `len(d)`

Return the number of items in the dictionary *d*.

## `d[key]`

Return the item of *d* with key *key*. Raises a `KeyError` if *key* is not in the map.

## `d[key] = value`

Set `d[key]` to *value*.

## `del d[key]`

Remove `d[key]` from *d*. Raises a `KeyError` if *key* is not in the map.

## `key in d`

Return `True` if *d* has a key *key*, else `False`.

## `key not in d`

Equivalent to `not key in d`.

## `iter(d)`

Return an iterator over the keys of the dictionary. This is a shortcut for `iter(d.keys())`.

## `clear()`

Remove all items from the dictionary.

## `copy()`

Return a shallow copy of the dictionary.

**get(*key*[, *default*])**

Return the value for *key* if *key* is in the dictionary, else *default*. If *default* is not given, it defaults to `None`, so that this method never raises a `KeyError`.

**items()**

Return a new view of the dictionary's items (`(key, value)` pairs). See the [documentation of view objects](#).

**keys()**

Return a new view of the dictionary's keys. See the [documentation of view objects](#).

**pop(*key*[, *default*])**

If *key* is in the dictionary, remove it and return its value, else return *default*. If *default* is not given and *key* is not in the dictionary, a `KeyError` is raised.

**popitem()**

Remove and return an arbitrary `(key, value)` pair from the dictionary.

`popitem()` is useful to destructively iterate over a dictionary, as often used in set algorithms. If the dictionary is empty, calling `popitem()` raises a `KeyError`.

**setdefault(*key*[, *default*])**

If *key* is in the dictionary, return its value. If not, insert *key* with a value of *default* and return *default*. *default* defaults to `None`.

**update([*other*])**

Update the dictionary with the key/value pairs from *other*, overwriting existing keys. Return `None`.

`update()` accepts either another dictionary object or an iterable of key/value pairs (as tuples or other iterables of length two). If keyword arguments are specified, the dictionary is then updated with those key/value pairs: `d.update(red=1, blue=2)`.

**values()**

Return a new view of the dictionary's values. See the [documentation of view objects](#).

# dict 생성 예

```
>>> a = dict(one=1, two=2, three=3)
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>> d = dict([('two', 2), ('one', 1), ('three', 3)])
>>> e = dict({'three': 3, 'one': 1, 'two': 2})
>>> a == b == c == d == e
True
```

```
>>> keys = ['one', 'two', 'three']; values = [1, 2, 3]
>>> c = dict(zip(keys, values))
>>> c
{'one': 1, 'two': 2, 'three': 3}
>>> len(c)
3
>>>
>>> c['one']
1
>>> c['one'] = 100
>>> c
{'one': 100, 'two': 2, 'three': 3}
>>> c.keys()
dict_keys(['one', 'two', 'three'])
>>> c.values()
dict_values([100, 2, 3])
```

# dict 사용 예

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> list(tel.keys())
['irv', 'guido', 'jack']
>>> sorted(tel.keys())
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```

# 실행 모델

- 프로그램 구조
  - (코드) 블록(block)
    - 하나의 단위로 실행되는 파이썬 프로그램 일부
    - 예: 모듈, 함수 몸체, 클래스 정의, 대화식으로 입력된 각각의 명령어, 스크립트 파일 등
- 명명과 결합 (Naming and Binding)
  - 이름 결합(binding)
    - 함수의 형식 매개변수, import 문장, 클래스/함수 정의, for 반복문 헤더 등
  - 이름 해석(resolution)
    - 범위(scope): 블록 내 이름의 가시성 정의
    - 이름이 발견되지 않으면 `NameError/UnboundLocalError` 예외 발생
- 예외 (Exceptions)
  - 오류나 예외적인 상황을 처리하기 위해 코드 블록의 정상적인 흐름을 탈출하는 수단

# 내장 예외(Built-in Exceptions)

```

BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StopAsyncIteration
    +-- ArithmeticError
    |   +-- FloatingPointError
    |   +-- OverflowError
    |   +-- ZeroDivisionError
    +-- AssertionError
    +-- AttributeError
    +-- BufferError
    +-- EOFError
    +-- ImportError
        +-- ModuleNotFoundError
    +-- LookupError
    |   +-- IndexError
    |   +-- KeyError
    +-- MemoryError
    +-- NameError
    |   +-- UnboundLocalError
    
```

```

+-- OSError
    |   +-- BlockingIOError
    |   +-- ChildProcessError
    |   +-- ConnectionError
    |       +-- BrokenPipeError
    |       +-- ConnectionAbortedError
    |       +-- ConnectionRefusedError
    |       +-- ConnectionResetError
    +-- FileNotFoundError
    +-- InterruptedError
    +-- IsADirectoryError
    +-- NotADirectoryError
    +-- PermissionError
    +-- ProcessLookupError
    +-- TimeoutError
+-- ReferenceError
+-- RuntimeError
    |   +-- NotImplementedError
    |   +-- RecursionError
+-- SyntaxError
    |   +-- IndentationError
    |   +-- TabError
+-- SystemError
+-- TypeError
    
```

```

+-- ValueError
    |   +-- UnicodeError
    |       +-- UnicodeDecodeError
    |       +-- UnicodeEncodeError
    |       +-- UnicodeTranslateError
+-- Warning
    +-- DeprecationWarning
    +-- PendingDeprecationWarning
    +-- RuntimeWarning
    +-- SyntaxWarning
    +-- UserWarning
    +-- FutureWarning
    +-- ImportWarning
    +-- UnicodeWarning
    +-- BytesWarning
    +-- ResourceWarning
    
```



# 실습: 사용자 정의 예외 처리

- 패키지: lesson02.exception
- MyInputError.py

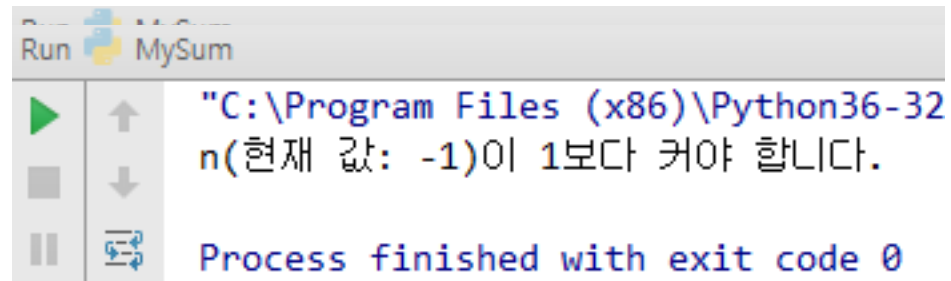
```
1  #!/usr/bin/env python
2
3
4  class MyInputError(Exception):
5      def __init__(self, message):
6          self.message = message
```

- MySum.py

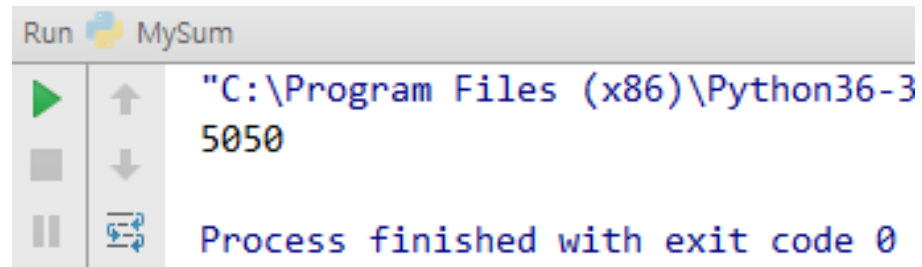
```
1      from lesson02.exception.MyInputError import MyInputError
2
3
4      def get_sum(n):
5          if n < 1:
6              raise MyInputError(f'n(현재 값: {n})이 1보다 커야 합니다.')
7
8          return int(n * (n+1) / 2)
9
10
11     if __name__ == '__main__':
12         try:
13             print(get_sum(-1))
14         except MyInputError as ex:
15             print(ex.message)
```

- 실행 화면

```
>>> get_sum(-1)
```



```
>>> get_sum(100)
```



# 람다 (Lambdas)

- 문법

```
lambda_expr      ::=  "lambda" [parameter_list]: expression
lambda_expr_nocond ::=  "lambda" [parameter_list]: expression_nocond
```

- 익명 함수를 만드는 데 사용
- 람다 표현식과 **map, reduce, filter** 함수를 사용하여 다양한 연산 가능

```
>>> data = list(range(10))
>>> data
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> data = list(map(lambda x: x+1, data))
>>> data
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> from functools import reduce
>>> result = reduce((lambda x,y: x+y), data)
>>> result
55
>>> data
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> less_than_five = list(filter(lambda x: x < 5, data))
>>> less_than_five
[1, 2, 3, 4]
```

# 실습: lambda\_test.py

```
1  #!/usr/bin/env python
2
3  from enum import Enum
4  from functools import reduce
5
6
7  class Sex(Enum):
8      FEMALE = "female"
9      MALE = "male"
10
11
12  class Person:
13      def __init__(self, name, age, sex):
14          self.name = name
15          self.age = age
16          self.sex = sex
17
18      def __lt__(self, other):
19          return self.name < other.name
```

```
22 persons = [Person('Linda', 21, Sex.FEMALE),  
23             Person('Oliver', 25, Sex.MALE),  
24             Person('Alice', 27, Sex.FEMALE),  
25             Person('Noah', 19, Sex.MALE),  
26             Person('Abby', 23, Sex.FEMALE),  
27             Person('Daisy', 25, Sex.FEMALE),  
28             Person('Samuel', 31, Sex.MALE),  
29             Person('Crystal', 31, Sex.FEMALE),  
30             Person('Tadeo', 33, Sex.MALE)]  
31  
32 # 남자는 몇 명?  
33 males = list(filter(lambda x : x.sex == Sex.MALE, persons))  
34 print('남자는 %d명입니다.' % len(males))
```

```
36 # 여자의 평균 나이는?
37 females = list(filter(lambda x : x.sex == Sex.FEMALE, persons))
38 ages = list(map(lambda x: x.age, females))
39 average_age = reduce(lambda x, y: x+y, ages) / len(ages)
40 print('여자의 평균 나이는 %.2f입니다.' % average_age)
41
42 print('20 ~ 25살인 여자의 이름을 정렬해서 출력')
43 selected_females = list(filter(lambda x: 20 <= x.age <= 25, females))
44 selected_females.sort()
45 for x in selected_females:
46     print(x.name)
```

남자는 4명입니다.

여자의 평균 나이는 25.40입니다.

20 ~ 25살인 여자의 이름을 정렬해서 출력

Abby

Daisy

Linda

# assert 문

- 방어적 프로그래밍(defensive programming) 시 입력 값의 범위를 확인하는데 유용

Assert statements are a convenient way to insert debugging assertions into a program:

```
assert_stmt ::= "assert" expression ["," expression]
```

The simple form, `assert expression`, is equivalent to

```
if __debug__:  
    if not expression: raise AssertionError
```

The extended form, `assert expression1, expression2`, is equivalent to

```
if __debug__:  
    if not expression1: raise AssertionError(expression2)
```



# assert 문 사용 방법

```
>>> def get_sum(n):  
    assert n>0  
    return int(n * (n+1)/2)
```

```
>>> get_sum(10)
```

```
55
```

```
>>> get_sum(-1)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#79>", line 1, in <module>
```

```
    get_sum(-1)
```

```
File "<pyshell#77>", line 2, in get_sum
```

```
    assert n>0
```

```
AssertionError
```

```
>>> def get_sum(n):  
    assert n>0, '0보다 큰 값을 입력해야 합니다!'  
    return int(n * (n+1)/2)
```

```
>>> get_sum(-1)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#84>", line 1, in <module>
```

```
    get_sum(-1)
```

```
File "<pyshell#83>", line 2, in get_sum
```

```
    assert n>0, '0보다 큰 값을 입력해야 합니다!'
```

```
AssertionError: 0보다 큰 값을 입력해야 합니다!
```

```
>>> try:
```

```
    get_sum(-1)
```

```
except AssertionError as ex:
```

```
    print(ex)
```

```
0보다 큰 값을 입력해야 합니다!
```

# 실습: 가위 바위 보 게임

- 파일 이름: : lesson04.rps.RPS\_game01.py
- 참고: 내장 함수 input([prompt]) <https://docs.python.org/3/library/functions.html#input>
- 실행 예  
컴퓨터> 가위(0), 바위(1), 보(2)를 선택하십시오. 세번 잘못 입력 시 컴퓨터에 의해 가위(0)를 선택하게 됩니다.  
사람> 5  
컴퓨터> 가위, 바위, 보는 0 ~ 2 사이의 정수를 입력해야 합니다. 세번 잘못 입력 시 컴퓨터가 선택합니다.  
사람> 2  
컴퓨터> '보' 를 선택하셨군요.  
컴퓨터> 당신이 이겼습니다. (또는 "당신이 졌습니다." 또는 "비겼습니다" 출력)

## • 실행 예

```
RPS_game01 (1) x
D:\PyProjects\py_design_pattern\venv\Scripts\python.exe D:/PyProjects/python_tutorial/py_tutorial_codes/lesson04/rps/RPS
FYI: computer_choice = 1
컴퓨터> 가위(0), 바위(1), 보(2)를 선택하십시오. 세번 잘못 입력 시 컴퓨터에 의해 가위(0)를 선택하게 됩니다. 나는 이미 선택했습니다.
사람> 11
컴퓨터> 가위, 바위, 보는 0~2 사이의 정수를 입력해야 합니다.세번 잘못 입력시 컴퓨터가 선택합니다.
사람> 11
컴퓨터> 가위, 바위, 보는 0~2 사이의 정수를 입력해야 합니다.세번 잘못 입력시 컴퓨터가 선택합니다.
사람> 11
컴퓨터> '가위'를 선택하셨군요.
컴퓨터> 당신이 졌습니다.
```

```
RPS_game01 (1) x
D:\PyProjects\py_design_pattern\venv\Scripts\python.exe D:/PyProjects/python_tutorial/py_tutorial_codes/lesson04/rps/RPS
FYI: computer_choice = 0
컴퓨터> 가위(0), 바위(1), 보(2)를 선택하십시오. 세번 잘못 입력 시 컴퓨터에 의해 가위(0)를 선택하게 됩니다. 나는 이미 선택했습니다.
사람> 2
컴퓨터> '보'를 선택하셨군요.
컴퓨터> 당신이 졌습니다.
```

```
RPS_game01 (1) x
D:\PyProjects\py_design_pattern\venv\Scripts\python.exe D:/PyProjects/python_tutorial/py_tutorial_codes/lesson04/rps/RPS
FYI: computer_choice = 2
컴퓨터> 가위(0), 바위(1), 보(2)를 선택하십시오. 세번 잘못 입력 시 컴퓨터에 의해 가위(0)를 선택하게 됩니다. 나는 이미 선택했습니다.
사람> 0
컴퓨터> '가위'를 선택하셨군요.
컴퓨터> 당신이 이겼습니다.
```

# RPS\_game01.py

- 패키지: lesson04.rps

```
1  # RPS_game01.py
2
3  from random import seed
4  from random import randint
5
6
7  def print_initial_message():
8      """
9      초기 메시지를 출력
10     :return: None
11     """
12     print("컴퓨터> 가위(0), 바위(1), 보(2)를 선택하십시오. "\
13           "세번 잘못 입력 시 컴퓨터에 의해 가위(0)를 선택하게 됩니다. "\
14           "나는 이미 선택했습니다.")
15     return
```

```

18 def get_user_choice():
19     """
20     사용자 입력을 받아 처리. 5초 안에 입력하지 못할 경우
21     0(가위)를 강제로 반환하도록 함.
22     :return: 0(가위), 1(바위), 2(보)를 반환
23     """
24     g = None
25     count = 0
26     while not (g == 0 or g == 1 or g == 2):
27         count += 1
28         g = int(input('사람> '))
29         if count >= 3:
30             g = 0
31         if not (g == 0 or g == 1 or g == 2):
32             print('컴퓨터> 가위, 바위, 보는 0~2 사이의 정수를 입력해야 합니다.' \
33                   '세번 잘못 입력시 컴퓨터가 선택합니다.')
34
35     return g
    
```

```
38 def get_winner_status(user_choice, computer_choice):
39     """
40     사용자 입력과 컴퓨터가 선택한 값을 비교
41     :return: 1 - 사용자가 이김, 0 - 비김, -1 - 컴퓨터가 이김
42     """
43     status = -1
44     if (user_choice == 0 and computer_choice == 2) or \
45         (user_choice == 1 and computer_choice == 0) or \
46         (user_choice == 2 and computer_choice == 1):
47         status = 1
48     elif user_choice == computer_choice:
49         status = 0
50     return status
```

```
53 ▶ if __name__ == "__main__":
54     rps_dict = {0: '가위', 1: '바위', 2: '보'}
55     seed()
56     computer_choice = randint(0,2)
57     print('FYI: computer_choice = {0}'.format(computer_choice))
58
59     print_initial_message()
60
61     while True:
62         user_choice = get_user_choice()
63         if user_choice == 0 or user_choice == 1 or user_choice == 2:
64             break
65
66     selected = rps_dict[user_choice]
67     print(f"컴퓨터> '{selected}'를 선택하셨습니다.")
68
69     status = get_winner_status(user_choice, computer_choice)
70
71     if status == 1:
72         print('컴퓨터> 당신이 이겼습니다.')
73     elif status == 0:
74         print('컴퓨터> 비겼습니다.')
75     else:
76         print('컴퓨터> 당신이 졌습니다.')
```

# 정규 표현식(Regular Expression)

- 정규 표현식
  - Regular expressions, REs, regexs, regex patterns
  - 다음과 같은 문자열 처리에 활용 가능: 동일 문자열 찾기, 이메일 주소 형식 일치 여부, 암호 작성 규칙 정합 여부
  - 정규 표현식은 일련의 바이트 코드로 **컴파일** 된 후, C언어로 작성된 **정합 엔진(matching engine)**에 의해 실행
- 메타 문자(metacharacters)
  - 일반적인 문자와 달리 특수한 의미를 지니는 문자

. ^ \$ \* + ? { } [ ] \ | ( )



- 메타 문자 [ ... ]
  - 처리를 원하는 문자 집합을 나타내는 문자 클래스
  - [abc], [a-c] : 'a', 'b', 또는 'c'
  - [a-z] : 알파벳 소문자 중 하나를 의미
  - [akm\$] : '\$'도 원래 메타 문자이지만, 여기서는 '\$' 문자를 의미
  - [^5]; 5를 제외한 다른 모든 문자
  - [ab\w[c\w\w]] : 앞에 있는 '\w'는 확장 비트열(escape sequence)를 의미. '[', '\w'도 포함
  - [\w\W]: [a-zA-Z0-9\_]를 의미

# 특수 문자열

- `\d` : 한자리 십진수(decimal number) [0-9]
- `\D` : 숫자가 아닌 문자 [^0-9]
- `\s` : 여백 문자(whitespace character) [\t\n\r\f\v]
- `\S` : 비 여백 문자 [^\t\n\r\f\v]
- `\w` : 영숫자 문자 [a-zA-Z0-9\_]
- `\W` : 비영숫자 문자 [^a-zA-Z0-9\_]

# 반복 문자

- \* : 이전 RE가 0번 이상 있는지 비교.  $ab^* \rightarrow a, ab, abb, \dots$
- + : 이전 RE가 1번 이상 있는지 비교.  $ab^+ \rightarrow ab, abb, \dots$
- ? : 이전 RE가 있거나 없는지 비교.  $ab? \rightarrow a, ab$
- {m,n} : m번 이상, n번 이하

# 특수 문자

- "." : 개행 문자('\n')를 제외한 모든 문자
- "^" : 문자열의 시작

```
>>> print(re.search('^From', 'From Here to Eternity'))
<_sre.SRE_Match object; span=(0, 4), match='From'>
>>> print(re.search('^From', 'Reciting From Memory'))
None
```

- "\$" : 문자열의 끝

```
>>> print(re.search('{}$', '{block}'))
<_sre.SRE_Match object; span=(6, 7), match='}'>
>>> print(re.search('{}$', '{block} '))
None
>>> print(re.search('{}$', '{block}\n'))
<_sre.SRE_Match object; span=(6, 7), match='}'>
```

- "|" : "or" 연산자

# re 모듈 함수

- 참고 [URL:https://docs.python.org/3/library/re.html](https://docs.python.org/3/library/re.html)
- **re.compile(pattern, flags=0)**
  - 정규 표현식 문자열 pattern을 컴파일하여 RE 객체 (regex) 반환
  - match(), search() 등의 메소드에서 사용
- **re.search(pattern, string, flags=0)**
  - string을 처음부터 끝까지 훑어보면서 regex와 일치하는 문자열 찾기
  - 일치하는 문자열 없으면 None 반환
- **re.match(pattern, string, flags=0)**
  - string 시작에서부터 일치 여부 확인하여 일치하는 경우 match 객체 반환, 일치하지 않으면 None 반환

- **re.split(pattern, string, maxsplit=0, flags=0)**

- pattern에 맞추어 string을 분리하여 list 객체로 반환
- 그룹 괄호가 사용되면 패턴의 모든 그룹 텍스트가 결과 리스트에 포함
  - string 시작부터 pattern이 일치하면 해당 단어부터 결과 리스트에 포함
  - string 시작부터 pattern이 일치하지 않으면 ""(empty string)부터 결과 리스트에 포함
  - string 끝부분도 시작 부분의 경우와 동일하게 처리됨
- maxsplit > 0이면 최대 maxsplit만큼만 string을 분리

```
D: #>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37)
Type "help", "copyright", "credits" or "license()" for more
>>> import re
>>> data = 'Words, words, words.'
>>> re.split('\W+', data)
['Words', 'words', 'words', '']
>>> re.split('(\W+)', data)
['Words', '', 'words', '', 'words', '', '']
>>> re.split('\W+', data, 1)
['Words', 'words, words.']
>>> re.split('[a-f]+', '0a3B9', flags=re.IGNORECASE)
['0', '3', '9']
>>> re.split('(\W+)', '...words, woreds...')
['', '...', 'words', '', 'woreds', '...', '']
```

- `re.findall(pattern, string, flags=0)`
  - 일치하는 문자열을 찾아서 list 객체로 반환
- `re.finditer(pattern, string, flags=0)`
  - 일치하는 문자열을 찾아서 iterator 객체로 반환
- `re.sub(pattern, repl, string, count=0, flags=0)`
  - string에서 해당하는 pattern을 찾아서 repl로 대체
  - count: 최대 대체 회수
- `re.escape(pattern)`
  - pattern에서 ASCII 문자, 숫자, '\_'를 제외하고 모든 문자를 확장
- `re.purge()`
  - 정규 표현식 캐시를 삭제

# regex 객체

- re.compile()에 의하여 반환된 객체
- <https://docs.python.org/3/library/re.html#re-objects>
- 주요 메소드
  - **regex.match(string[, pos[, endpos]])**
    - string 시작에서부터 일치 여부 확인하여 일치하는 경우 match 객체 반환, 일치하지 않으면 None 반환
  - **regex.search(string[, pos[, endpos]])**
    - string을 처음부터 끝까지 훑어보면서 regex와 일치하는 문자열 찾을
    - 일치하는 문자열 없으면 None 반환
  - **regex.fullmatch(string[, pos[, endpos]])**
  - **regex.split(string, maxsplit=0)**
  - **regex.findall(string[, pos[, endpos]])**
  - **regex.finditer(string[, pos[, endpos]])**
  - **regex.sub(repl, string, count=0)**



# match 객체

- regex.match()에 의해 반환된 객체
- <https://docs.python.org/3/library/re.html#match-objects>
- 주요 메소드
  - match.group([group1, ...])
    - 일치하는 문자열에서 하나 이상의 부그룹(subgroup)을 반환

```
>>> m = re.match(r"(\w+) (\w+)", "Isaac Newton, physicist")
>>> m.group(0)      # The entire match
'Isaac Newton'
>>> m.group(1)      # The first parenthesized subgroup.
'Isaac'
>>> m.group(2)      # The second parenthesized subgroup.
'Newton'
>>> m.group(1, 2)   # Multiple arguments give us a tuple.
('Isaac', 'Newton')
```

- `match.groups(default=None)`
  - 1부터 끝까지 일치하는 부그룹을 가지는 튜플을 반환

```
>>> m = re.match(r"(\d+)\.(\d+)", "24.1632")
>>> m.groups()
('24', '1632')
```

- `match.groupdict(default=None)`
  - 이름 붙여진 부그룹에 포함되는 dict 객체를 반환

```
>>> m = re.match(r"(?P<first_name>\w+) (?P<last_name>\w+)", "Malcolm Reynolds")
>>> m.groupdict()
{'first_name': 'Malcolm', 'last_name': 'Reynolds'}
```

- `match.start([group]), match.end([group])`
  - `group`과 일치하는 substring의 시작과 끝 색인을 반환
  - `m.group(g) == m.string[m.start(g):m.end(g)]`

```
>>> email = "tony@tiremove_thisger.net"
>>> m = re.search("remove_this", email)
>>> email[:m.start()] + email[m.end():]
'tony@tiger.net'
```

- `match.span([group])`
  - `(m.start(group), m.end(group))` 튜플을 반환

# 사용 예

```
>>> import re
>>> p = re.compile('[a-z]+')
>>> p
re.compile('[a-z]+')
```

```
>>> m = p.match('tempo')
>>> m
<_sre.SRE_Match object; span=(0, 5), match='tempo'>
```

```
>>> m.group()
'tempo'
>>> m.start(), m.end()
(0, 5)
>>> m.span()
(0, 5)
```

```
>>> print(p.match('::: message'))
None
>>> m = p.search('::: message'); print(m)
<_sre.SRE_Match object; span=(4, 11), match='message'>
>>> m.group()
'message'
>>> m.span()
(4, 11)
```

# 일반적인 사용법

- match 객체를 저장한 다음 조건문으로 확인
- 일치하는 게 없으면 None을 반환

```
p = re.compile( ... )  
m = p.match( 'string goes here' )  
if m:  
    print('Match found: ', m.group())  
else:  
    print('No match')
```

# findall() & finditer() 사용

- `regex.findall(string[, pos[, endpos]])`
  - 일치하는 문자열을 찾아서 list 객체로 반환
  - 참고: `re.findall(pattern, string, flags=0)`

```
>>> p = re.compile('\d+')
>>> p.findall('12 drummers drumming, 11 pipers piping, 10 lords a-leaping')
['12', '11', '10']
```

- `regex.finditer(string[, pos[, endpos]])`
  - 일치하는 문자열을 찾아서 iterator 객체로 반환
  - 참고: `re.finditer(pattern, string, flags=0)`

```
>>> iterator = p.finditer('12 drummers drumming, 11 ... 10 ...')
>>> iterator
<callable_iterator object at 0x...>
>>> for match in iterator:
...     print(match.span())
...
(0, 2)
(22, 24)
(29, 31)
```

# 참고. iterable, iterator

- iterable
  - iterator 객체를 반환하는 `__iter__()` 메소드가 있는 객체
  - `__getitem__()` 메소드로 객체를 한 개씩 반환
- iterator
  - `__next__()` 메소드가 있는 객체
- 자바의 경우 iterable은 collection 객체를 의미
- 참고 URL: <https://docs.python.org/3/tutorial/classes.html#iterators>

## “^” & “\$” 사용 예

- 문자열 앞에서 “From”을 찾음.

```
>>> print(re.search('^From', 'From Here to Eternity'))
<_sre.SRE_Match object; span=(0, 4), match='From'>
>>> print(re.search('^From', 'Reciting From Memory'))
None
```

- 문자열 뒤에서 “}”를 찾음

```
>>> print(re.search('}$', '{block}'))
<_sre.SRE_Match object; span=(6, 7), match='}'>
>>> print(re.search('}$', '{block} '))
None
>>> print(re.search('}$', '{block}\n'))
<_sre.SRE_Match object; span=(6, 7), match='}'>
```



# 문자 모음(Grouping)\*\*\*

- 그룹 시작: '('
- 그룹 끝: ')'
- 사용법
  - (ab)\* : ab, abab, ababab, ...

```
>>> p = re.compile('(ab)*)  
>>> print(p.match('ababababab').span())  
(0, 10)
```

```
>>> p = re.compile('(a(b)c)d')  
>>> m = p.match('abcd')  
>>> m  
<_sre.SRE_Match object; span=(0, 4), match='abcd'>  
>>> m.groups()  
('abc', 'b')  
>>> m.group()  
'abcd'
```

# 실습: simple\_search.py

```
1  import re
2
3  text = 'Hello, world! This is a hello.'
4  p1 = re.compile('[hH]ello')
5  m1 = p1.match(text)
6  print('match = ', m1)
7
8  m2 = p1.search(text)
9  print('search = ', m2)
10
11 m3 = p1.findall(text)
12 print('findall = ', m3)
13
14 m4 = p1.finditer(text)
15 print('finditer = ', m4)
16
17 for item in m4:
18     print(item)
19     start, stop = item.span()
20     print(item.group(), ': start = ', start, 'stop = ', stop)
21     print(item.group(0))
```

```
23 p2 = re.compile('[Hh]ello')
24 m1 = p2.match(text)
25 if m1 is not None:
26     print('match span = ', m1.span())
27
28 m2 = p2.search(text)
29 if m2 is not None:
30     print('search span ', m2.span())
31
32 m3 = p2.findall(text)
33 print(m3)
```

# 실행 결과

```
simple_search x
D:\PyProjects\py_design_pattern\venv\Scripts\python.exe D:/Py
match = <re.Match object; span=(0, 5), match='Hello'>
search = <re.Match object; span=(0, 5), match='Hello'>
findall = ['Hello', 'hello']
finditer = <callable_iterator object at 0x000001DD64B9FF60>
<re.Match object; span=(0, 5), match='Hello'>
Hello : start = 0 stop = 5
Hello
<re.Match object; span=(24, 29), match='hello'>
hello : start = 24 stop = 29
hello
match span = (0, 5)
search span (0, 5)
['Hello', 'hello']

Process finished with exit code 0
```

# 실습: file\_search.py

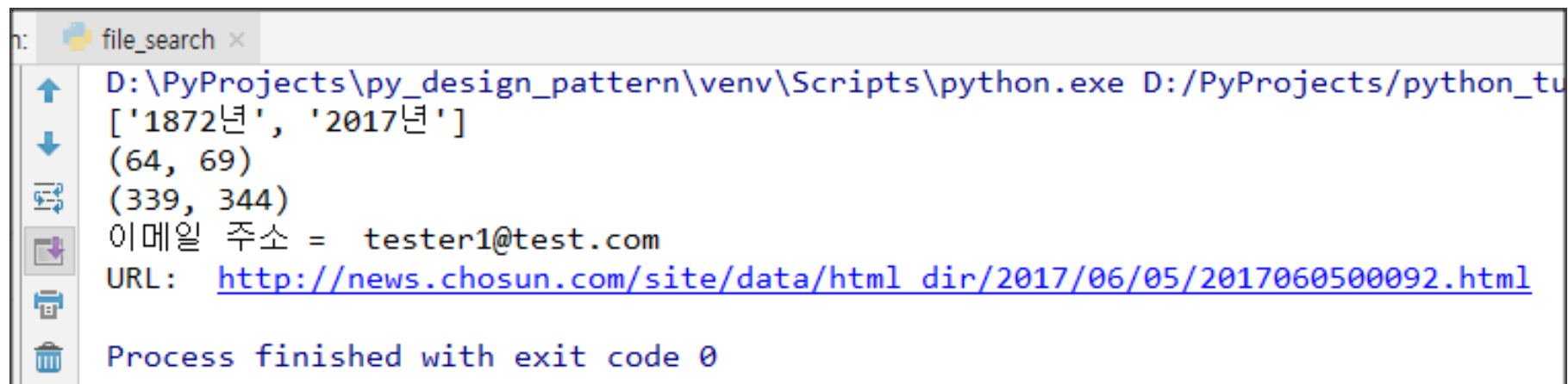
```
1  import re
2
3  data_f = open('data.txt', encoding='UTF-8')
4  text = data_f.read()
5  # print(text)
6
7  # 연도 찾기
8  regex = re.compile('\d{4,4}년')
9  m = regex.findall(text)
10 print(m)
11
12 for match in regex.finditer(text):
13     print(match.span())
14
15 # 이메일 주소 찾기
16 regex = re.compile('[a-zA-Z]\w+@\w+\.\w+')
17 for match in regex.finditer(text):
18     start, stop = match.span()
19     print('이메일 주소 = ', text[start:stop])
20
21
22 # URL 주소 찾기
23 regex = re.compile('http://[\w./]+')
24 print('URL: ', regex.search(text).group(0))
```

# data.txt

- 교수 홈페이지 (<https://compnet.deu.ac.kr> 공지사항 1020번)

제목	객체지향모델링 Python 실습 자료: data.txt
글 쓴 사람	이종민교수
글 쓴 날짜	2020-09-03
작성자: 박승혁 기자 (tester1@test.com) 입력 : 2017.06.05 03:03	
<p>미국 하버드대는 1872년부터 신입생 전원에게 '하버드 글쓰기 프로그램' 강좌를 146년간 하고 있다. 적어도 한 학기는 수강을 의무화했다. 매해 입학생 1700여명이 문·이과 전공에 관계없이 '학술적 글쓰기' 능력을 체득하는 것이다. 하버드대에 따르면, 이 수업을 들은 학생의 73%는 "글쓰기 능력 향상은 물론 대학 수업에 더 적극적으로 참여하게 됐다"고 했다.</p>	
출처 : <a href="http://news.chosun.com/site/data/html_dir/2017/06/05/2017060500092.html">http://news.chosun.com/site/data/html_dir/2017/06/05/2017060500092.html</a>	
2017년 6월 26일 부터 30일까지 python 특강이 동의대학교 컴퓨터소프트웨어공학과에서 진행되었다.	
첨부 파일: <u>data.txt</u>	

# 실행 결과



```
n: file_search x
D:\PyProjects\py_design_pattern\venv\Scripts\python.exe D:/PyProjects/python_tu
['1872년', '2017년']
(64, 69)
(339, 344)
이메일 주소 = tester1@test.com
URL: http://news.chosun.com/site/data/html\_dir/2017/06/05/2017060500092.html
Process finished with exit code 0
```

# 파일 객체

- 참고: <https://docs.python.org/3/glossary.html#term-file-object>
- read(), write()와 같은 파일 접근 API를 사용할 수 있는 객체
- 생성 방법에 따라서 디스크의 파일 뿐만 아니라 표준 입출력, 인메모리 버퍼, 소켓, 파이프 등 다양한 통신 장치 접근 가능
- 파일 객체: aka file-like object 또는 스트림(streams)
- 파일 객체 종류 – io 모듈(<https://docs.python.org/3/library/io.html#module-io>) 참조
  - 원시 이진 파일(raw binary files)
  - 버퍼 이진 파일(buffered binary files)
  - 텍스트 파일(text files)



# 텍스트 입출력

- 텍스트 스트림 생성

```
f = open("myfile.txt", "r", encoding="utf-8")
```

- 인메모리 텍스트 스트림 생성

```
f = io.StringIO("some initial text data")
```

# 이진 입출력

- 또는 버퍼 입출력(buffered input/output)이라고도 함.
- 이진 스트림 생성

```
f = open("myfile.jpg", "rb")
```

- 인메모리 이진 스트림 생성

```
f = io.BytesIO(b"some initial binary data: \x00\x01")
```

# 원시 입출력(Raw I/O)

- 이진/텍스트 스트림을 만들기 위한 저수준 API
- 생성 방법

```
f = open("myfile.jpg", "rb", buffering=0)
```

- RawIOBase 클래스 참조
  - <https://docs.python.org/3/library/io.html#io.RawIOBase>

# io 모듈 ABC

ABC	Inherits	Stub Methods	Mixin Methods and Properties
<code>IOBase</code>		<code>fileno</code> , <code>seek</code> , and <code>truncate</code>	<code>close</code> , <code>closed</code> , <code>__enter__</code> , <code>__exit__</code> , <code>flush</code> , <code>isatty</code> , <code>__iter__</code> , <code>__next__</code> , <code>readable</code> , <code>readline</code> , <code>readlines</code> , <code>seekable</code> , <code>tell</code> , <code>writable</code> , and <code>writelines</code>
<code>RawIOBase</code>	<code>IOBase</code>	<code>readinto</code> and <code>write</code>	Inherited <code>IOBase</code> methods, <code>read</code> , and <code>readall</code>
<code>BufferedIOBase</code>	<code>IOBase</code>	<code>detach</code> , <code>read</code> , <code>read1</code> , and <code>write</code>	Inherited <code>IOBase</code> methods, <code>readinto</code>
<code>TextIOBase</code>	<code>IOBase</code>	<code>detach</code> , <code>read</code> , <code>readline</code> , and <code>write</code>	Inherited <code>IOBase</code> methods, <code>encoding</code> , <code>errors</code> , and <code>newlines</code>

# 파일 객체 입출력 메소드

- `close()`: 스트림을 내보내고(flush) 종료
- `fileno()`: 파일 기술자(정수) 반환
- `flush()`: 스트림의 쓰기 버퍼에 있는 내용을 모두 내보냄.
- `readable()`, `writable()`: 읽기/쓰기 가능한지 여부(True/False)를 반환
- `read(size=-1)`
  - size 바이트만큼 (또는 size보다 적게) 읽어 반환, size == -1 또는 지정하지 않으면 `readall()` 호출
- `readline(size=-1)`
  - 한 줄(줄의 끝: b'\\n') 읽어 반환
  - 텍스트 파일의 경우 `open()`에서 `newline` 인자 지정 가능
  - size > 0이면 최대 size 바이트만 읽음
- `readlines(hint=-1)`

- write(b)
  - 바이트 객체(byte-like object, <https://docs.python.org/3/glossary.html#term-bytes-like-object>)를 출력하고, 출력한 바이트 수를 반환
- writelines(lines)
  - 스트림에 lines(줄 리스트)를 출력
  - 줄 구분자(line separator) 추가 없이 출력하므로 lines의 각 항목 끝에 줄 구분자 추가해 주어야 함.
- 참고: 16.2 io –Core tools for working with streams, <https://docs.python.org/3/library/io.html#high-level-module-interface> (2017년 월)

# open() 내장 함수

- `open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)`
- mode 사용 방법

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)
'U'	universal newlines mode (deprecated)

# 참고. print() 내장 함수

- `print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)`
  - 참고: <https://docs.python.org/3/library/functions.html#print>
  - 텍스트 스트림 file에 object를 출력
  - 구분자는 sep로 지정: 기본값 "
  - 줄 끝은 end로 지정: 기본값 '\n'



## 실습: 파일 읽기 - file\_demo\_01.py

```
3  ▶ if __name__ == '__main__':  
4      file_name = 'hello.txt'  
5      f1 = open(file_name, 'r', encoding='UTF-8')  
6      read_data = f1.read()  
7      print(f'[{file_name}]')  
8      print(read_data)  
9      print()  
10     f1.close()  
11  
12     file_name = 'menu.json'  
13     with open(file_name, 'r') as f2:  
14         read_data = f2.read()  
15         print('{}'.format(file_name))  
16         print(read_data)  
17         f2.close()
```

- 'hello.txt'

```

1 Hello, world!
2 안녕하세요, 텍스트 파일 입출력을 위한 테스트 파일입니다.
3 English, 한글 모두 잘 보여야 합니다.
    
```

- menu.json

```

1 {
2   "menu": {
3     "id": "file",
4     "value": "File",
5     "popup": {
6       "menuitem": [
7         {
8           "value": "New",
9           "onclick": "CreateNewDoc()"
10        },
11        {
12          "value": "Open",
13          "onclick": "OpenDoc()"
14        },
15        {
16          "value": "Close",
17          "onclick": "CloseDoc()"
18        }
19      ]
20    }
21  }
22 }
    
```

# 실행 결과

Run file\_demo\_01

"C:\Program Files\Python36\python.exe" E:/User/jong  
[hello.txt]

Hello, world!

안녕하세요, 텍스트 파일 입출력을 위한 테스트 파일입니다.

English, 한글 모두 잘 보여야 합니다.

[menu.json]

```
{  
  "menu": {  
    "id": "file",  
    "value": "File",  
    "popup": {  
      "menuitem": [  
        {  
          "value": "New",
```

# 실습: 파일 한 줄씩 읽기 - file\_demo\_02.py

```
2 import sys
3
4 ► if __name__ == '__main__':
5     file_name = 'hello.txt'
6     with open(file_name, encoding='UTF-8') as f:
7         line = f.readline()
8         while line != "":
9             print(line, end='')
10            line = f.readline()
11        f.close()
12
13    print('-----')
14    with open(file_name, encoding='UTF-8') as f:
15        for line in f:
16            # Q: print(line)과의 차이점은?
17            sys.stdout.write(line)
18        f.close()
```

# 실행 결과

```
Run file_demo_02
"C:\Program Files\Python36\python.exe" E:/User/jongn
Hello, world!
안녕하세요, 텍스트 파일 입출력을 위한 테스트 파일입니다.
English, 한글 모두 잘 보여야 합니다.
-----
Hello, world!
안녕하세요, 텍스트 파일 입출력을 위한 테스트 파일입니다.
English, 한글 모두 잘 보여야 합니다.

Process finished with exit code 0
```

# 파일 쓰기: file\_demo\_03.py

```

2
3 ▶ if __name__ == '__main__':
4     f = open('demo.txt', 'w')
5     f.write('파일 쓰기를 시작합니다.')
6     f.write('어떻게 되는지 잘 봅시다.\n')
7     for i in range(10):
8         f.write(str(i))
9         f.write(', ')
10    f.flush()
11    f.close()

```

## • 실행 결과

demo.txt

```

1 파일 쓰기를 시작합니다.어떻게 되는지 잘 봅시다.
2 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

```

File was loaded in the wrong encoding: 'UTF-8'

```

1 0000 000< 00000J00.0000 0G000 00 000ô0.
2 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

```

# JSON 파일 처리: file\_demo\_json.py

```
3 import json
4
5 if __name__ == '__main__':
6     with open('test_data.json', 'w') as f:
7         # 메모리에서의 JSON 처리
8         json_scores = '{"id_list": [1001, 1002, 1003], ' \
9                        '"course" : ["국어", "영어", "수학"], ' \
10                       '"1001" : ["홍길동", 100, 95, 90], ' \
11                       '"1002" : ["전우치", 90, 95, 95], ' \
12                       '"1003" : ["Tom", 85, 95, 95] ' \
13                       '}'
14         data = json.loads(json_scores)
15         print('data type = ', type(data))
16         print(data)
17         for id in data['id_list']:
18             sum = 0
19             for index in range(3):
20                 sum += int(data[str(id)][index + 1])
21             # printf-style String formatting
22             # https://docs.python.org/3/library/stdtypes.html#
23             print('%s : %s 평균 점수 = %3.1f'
24                   % (id, data[str(id)][0], sum / 3))
```

```
26     json.dump(data, f)
27     f.close()
28     print('-----')
29
30     with open('menu.json') as f:
31         # 파일에서의 JSON 처리
32         data = json.load(f)
33         # print(data)
34         # print(json.dumps(data))
35         # print(json.dumps(data, sort_keys=True, indent=4))
36
37         key_list = data['menu'].keys()
38         print(key_list)
39         for key in key_list:
40             print(key, ':', data['menu'][key])
41             #print(key, ':', json.dumps(data['menu'][key], indent=4))
42
43     f.close()
```



# 실행 결과

```
Run file_demo_json
"C:\Program Files\Python36\python.exe" E:/User/jongmin/강의/외부세미나
/170626-IoT-파이썬특강/PyProjects/IoT_Python/code/lesson19
/file_demo_json.py
data type = <class 'dict'>
{'id_list': [1001, 1002, 1003], 'course': ['국어', '영어', '수학'],
 '1001': ['홍길동', 100, 95, 90], '1002': ['전무치', 90, 95, 95],
 '1003': ['Tom', 85, 95, 95]}
1001 : 홍길동 평균 점수 = 95.0}
1002 : 전무치 평균 점수 = 93.3}
1003 : Tom 평균 점수 = 91.7}
-----
dict_keys(['id', 'value', 'popup'])
id : file
value : File
popup : {'menuitem': [{'value': 'New', 'onclick': 'CreateNewDoc()'},
 {'value': 'Open', 'onclick': 'OpenDoc()'}, {'value': 'Close',
 'onclick': 'CloseDoc()'}]}
```

Process finished with exit code 0

# numpy

- 다차원 배열과 행렬을 지원하는 파이썬 패키지

- 발음: 넘파이 / 넘피

- 설치 여부 확인

C:₩> pip search numpy

- 설치 방법

C:₩> pip install numpy

```
C:₩Users₩skylo>pip -V
pip 19.1.1 from C:₩Users₩skylo₩AppData₩Roaming₩Python₩Python37₩site-packages₩pip (python 3.7)

C:₩Users₩skylo>pip search numpy
numpy (1.17.2) - NumPy is the fundamental package for array computing with Python.
  INSTALLED: 1.16.0
  LATEST: 1.17.2
numpy-utils (0.1.6) - NumPy utilities.
numpy-cloud (0.0.5) - Numpy in the cloud
numpy-turtle (0.1) - Turtle graphics with NumPy
numpy-sugar (1.5.0) - Missing NumPy functionalities
root-numpy (4.8.0) - The interface between ROOT and NumPy
msgpack-numpy (0.4.4.3) - Numpy data serialization using msgpack
numpy-quaternion (2019.7.23.15.26.49) - Add built-in support for quaternions to numpy_
```

# 실습: numpy 사용 예 - numpy\_test.py

```

1  #!/usr/bin/env python3
2
3  import numpy as np
4  import cv2
5
6  data = [x for x in range(10)] # list comprehension
7  print('type(data) = ', type(data))
8  print(data)
9
10 a = np.array(data) # numpy 배열 초기화
11 print('type(a) = %s, shape = %s' % (type(a), a.shape))
12 print(a)
13
14 b = a.reshape(2,5)
15 print('type(b) = %s, shape = %s' % (type(b), b.shape))
16 print(b)

```

```

type(data) = <class 'list'>
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```

```

type(a) = <class 'numpy.ndarray'>, shape = (10,)
[0 1 2 3 4 5 6 7 8 9]

```

```

type(b) = <class 'numpy.ndarray'>, shape = (2, 5)
[[0 1 2 3 4]
 [5 6 7 8 9]]

```

```

18 n1 = np.ones((2,2))
19 print('ones =\n', n1)
20
21 n2 = np.zeros((2,2))
22 print('zeros =\n', n2)
23
24 n3 = np.eye(2)
25 print('eye =\n', n3)
26
27 # slicing example
28 a = np.array([x for x in range(1,13)]).reshape((3,4))
29 print('a = \n', a)
30
31 b = a[:2, 1:3] # 2, 3: excluded. equals to a[0:2, 1:3]
32 print('b =\n', b)
33
34 print(b[0,0] == a[0,1])
35
36 print('a.dtype =', a.dtype)
37 c = a.astype(np.int8) # overflow에 의한 data loss 발생 가능
38 print('c.dtype =', c.dtype)

```

```

ones =
[[1. 1.]
 [1. 1.]]
zeros =
[[0. 0.]
 [0. 0.]]
eye =
[[1. 0.]
 [0. 1.]]

```

```

a =
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
b =
[[2 3]
 [6 7]]
True

```

```

a.dtype = int32
c.dtype = int8

```

```

40 a = a / 4
41 print('a/4 =\n', a)
42
43 d = np.array([x for x in range(1,13)], dtype=np.float64).reshape((3,4))
44 print('d.dtype =', d.dtype)
45 print('d = \n', d)
46
47 d = d / 4
48 print('d/4 =\n', d)
49
50 # jpg 읽기
51 img = cv2.imread('dog.jpg')
52 print('type(img) = %s, size = %s, shape = %s' % (type(img), img.size, img.shape))
53                                     |type(img) = <class 'numpy.ndarray'>, size = 4321800, shape = (1470, 980, 3)
54 cv2.imshow('img', img)
55 cv2.waitKey(3000)
56
57 height, width, _ = img.shape
58 small_img = cv2.resize(img, (int(width/2), int(height/2)))
59 # cv2.imshow('small_img', small_img)
60 # cv2.waitKey(3000)

```

```

a/4 =
[[0.25 0.5  0.75 1.  ]
 [1.25 1.5  1.75 2.  ]
 [2.25 2.5  2.75 3.  ]]
d.dtype = float64
d =
[[ 1.  2.  3.  4.]
 [ 5.  6.  7.  8.]
 [ 9. 10. 11. 12.]]
d/4 =
[[0.25 0.5  0.75 1.  ]
 [1.25 1.5  1.75 2.  ]
 [2.25 2.5  2.75 3.  ]]

```

```
62 cv2.imwrite('small_img.png', small_img)
63
64 small_img2 = np.array(small_img)
65
66 h, w, _ = small_img2.shape
67 print(small_img2.shape)           |(735, 490, 3)
68 small_img2[int(h/2):, :] = [0, 0, 0]
69
70 cv2.imshow('small_img', small_img)
71 cv2.imshow('small_img2', small_img2)
72 cv2.waitKey(5000)
```



