



Python

함수

김선녕(sykim.lecture@gmail.com)



내장함수
외장함수

abs(), all(), any() 함수

```
In [21]: abs(3)
```

```
Out[21]: 3
```

```
In [22]: abs(-3)
```

```
Out[22]: 3
```

```
In [23]: # 반복가능한 자료형 : for 문으로 값을 출력할 수 있는 것을 의미  
# 반복가능한 자료형으로 입력값에 따라 true/false 리턴  
all([1,2,3])
```

```
Out[23]: True
```

```
In [24]: all([1,2,3,0]) # 0은 false
```

```
Out[24]: False
```

```
In [25]: # 입력값이 하나라도 참이 있을 경우 true리턴. 1,2,3이 참이므로 True 리턴  
any([1,2,3,0])
```

```
Out[25]: True
```

```
In [26]: any([0,""])
```

```
Out[26]: False
```

chr(), ord() 함수

```
In [28]: # 아스키(ASCII)코드값을 입력받아 문자 출력  
chr(97)
```

```
Out[28]: 'a'
```

```
In [29]: chr(48)
```

```
Out[29]: '0'
```

```
In [87]: # 아스키(ASCII) 코드값을 리턴하는 함수  
ord('a')
```

```
Out[87]: 97
```

```
In [88]: ord('0')
```

```
Out[88]: 48
```

```
In [30]: # 객체가 자체적으로 가지고 있는 변수나 함수를 보여준다  
dir([1,2,3])
```

```
Out[30]: ['__add__',  
          '__class__',  
          '__contains__',  
          'delattr',  
          '  
          .....,  
          'index',  
          'insert',  
          'pop',  
          'remove',  
          'reverse',  
          'sort']
```

divmod() 함수

```
In [33]: # 나는 몫과 나머지를 튜플 형태로 리턴하는 함수  
divmod(7,3) # 7나누기 3의 몫은 2, 나머지는 1
```

```
Out[33]: (2, 1)
```

```
In [34]: divmod(1.3,0.2)
```

```
Out[34]: (6.0, 0.09999999999999998)
```

```
In [37]: # 자료형(리스트, 튜플, 문자열)을 입력받아 인덱스 값을 포함하여 객체를 리턴  
for i, name in enumerate(['body', 'foo', 'bar']):  
    print(i, name)
```

```
0 body  
1 foo  
2 bar
```

eval() 함수

```
In [38]: # 실행가능한 문자열을 입력받아 문자열을 실행한 결과값을 리턴  
eval('1+2')
```

Out[38]: 3

```
In [39]: eval("'hi' + 'a'")
```

Out[39]: 'hia'

```
In [40]: eval('divmod(4,3)')
```

Out[40]: (1, 1)

```
In [41]: def positive(l):  
         result = []  
         for i in l:  
             if i > 0:  
                 result.append(i)  
         return result  
  
print(positive([1,-3,2,0,-5,6]))  
  
[1, 2, 6]
```

```
In [42]: def positive(x):  
         return x > 0  
  
print(list(filter(positive, [1, -3, 2, 0, -5, 6])))  
  
[1, 2, 6]
```

```
In [43]: print(list(filter(lambda x: x > 0, [1, -3, 2, 0, -5, 6])))  
  
[1, 2, 6]
```


hex(), id() 함수

```
In [44]: hex(234)
```

```
Out[44]: '0xea'
```

```
In [45]: hex(3)
```

```
Out[45]: '0x3'
```

```
In [46]: # 객체의 고유 주소값(레퍼런스)를 리턴하는 함수  
a = 3  
id(3)
```

```
Out[46]: 1857121360
```

```
In [47]: id(a)
```

```
Out[47]: 1857121360
```

```
In [48]: b = a
```

```
In [49]: id(b) # 3, a, b는 모두 같은 객체를 가리킴
```

```
Out[49]: 1857121360
```

```
In [50]: id(4)
```

```
Out[50]: 1857121392
```

input(), int() 함수

10

```
In [56]: a = input()
```

hi

```
In [57]: a
```

```
Out[57]: 'hi'
```

```
In [58]: b = input("Enter: ")
```

Enter: hi

```
In [52]: # 문자열 형태의 숫자나 소수점이 있는 수자들을 정수 형태로 리턴하는 함수  
int('3')
```

```
Out[52]: 3
```

```
In [53]: int(3.4)
```

```
Out[53]: 3
```

```
In [54]: # int(x, radix) : radix 진수로 표현된 문자열 x를 10진수로 반환하여 리턴  
int('11', 2)
```

```
Out[54]: 3
```

```
In [55]: int('1A', 16)
```

```
Out[55]: 26
```

isinstance(), len() 함수

11

```
In [59]: # isinstance(object, class) 첫 번째 인수로 인스턴스, 두 번째 인수로 클래스 이름을 받는다  
# 입력으로 받은 인스턴스가 그 클래스의 인스턴스인지를 판단 : True/False 리턴  
class Person: pass  
  
a = Person()  
isinstance(a, Person)
```

Out [59]: True

```
In [60]: b = 3  
isinstance(b, Person)  
False
```

Out [60]: False

```
In [61]: len("python")
```

Out [61]: 6

```
In [62]: len([1,2,3])
```

Out [62]: 3

```
In [63]: len((1, 'a'))
```

Out [63]: 2

lambda() 함수

12

```
In [64]: ''' lambda는 함수를 생성할 때 사용. def와 동일한 역할  
          함수를 한줄로 간결하게 만들 때 사용 '''  
sum = lambda a,b:a+b  
sum(3,4)
```

Out[64]: 7

```
In [65]: def sum(a,b):  
          return a+b
```

```
In [66]: # myList에 람다 함수가 2개 추가됨  
myList = [lambda a,b:a+b, lambda a,b:a*b]  
myList
```

Out[66]: [<function __main__.<lambda>(a, b)>, <function __main__.<lambda>(a, b)>]

```
In [67]: myList[0]
```

Out[67]: <function __main__.<lambda>(a, b)>

```
In [68]: myList[0](3,4)
```

Out[68]: 7

```
In [69]: myList[1](3,4)
```

Out[69]: 12

list(), oct() 함수

13

```
In [70]: list("python")
```

```
Out[70]: ['p', 'y', 't', 'h', 'o', 'n']
```

```
In [71]: list((1,2,3))
```

```
Out[71]: [1, 2, 3]
```

```
In [72]: a = [1,2,3]
          b = list(a)
          b
```

```
Out[72]: [1, 2, 3]
```

```
In [77]: oct(34)
```

```
Out[77]: '0o42'
```

```
In [78]: oct(12345)
```

```
Out[78]: '0o30071'
```

```
In [79]: '''map(f, iterable)은 함수(f)와 반복 가능한(iterable) 자료형을 입력으로 받는다.  
map은 입력받은 자료형의 각 요소가 함수 f에 의해 수행된 결과를 묶어서 리턴하는 함수'''  
def two_times(numberList):  
    result = [ ]  
    for number in numberList:  
        result.append(number*2)  
    return result  
  
result = two_times([1, 2, 3, 4])  
print(result)  
  
[2, 4, 6, 8]
```

```
In [82]: def two_times(x): return x*2  
list(map(two_times, [1, 2, 3, 4]))
```

```
Out[82]: [2, 4, 6, 8]
```

```
In [83]: list(map(lambda a:a*2, [1,2,3,4]))
```

```
Out[83]: [2, 4, 6, 8]
```

```
In [85]: def plus_one(x):  
    return x + 1  
print(list(map(plus_one, [1,2,3,4,5])))
```

```
[2, 3, 4, 5, 6]
```

max(), min() 함수

15

```
In [73]: max([1,2,3])
```

```
Out[73]: 3
```

```
In [74]: max("python")
```

```
Out[74]: 'y'
```

```
In [75]: min([1,2,3])
```

```
Out[75]: 1
```

```
In [76]: min("python")
```

```
Out[76]: 'h'
```

```
In [ ]: f = open("binary_file", "rb") # 바이너리 읽기 모드  
        fread = open("read_mode.txt", 'r') # 파일읽기 모드  
        fread2 = open("read_mode.txt") # 파일읽기 모드. 위와 동일  
        fappend = open("append_mode.txt", 'a') # 추가 모드
```

pow(), range() 함수

16

```
In [90]: pow(2,4)
```

```
Out[90]: 16
```

```
In [91]: pow(3,3)
```

```
Out[91]: 27
```

```
In [92]: # 인수가 하나일 경우  
list(range(5))
```

```
Out[92]: [0, 1, 2, 3, 4]
```

```
In [94]: # 인수가 2개일 경우  
list(range(5,10))
```

```
Out[94]: [5, 6, 7, 8, 9]
```

```
In [95]: # 인수가 3개일 경우  
list(range(1,10,2))
```

```
Out[95]: [1, 3, 5, 7, 9]
```


sorted(), sort() 함수

17

```
In [96]: sorted([3,1,2])
```

```
Out[96]: [1, 2, 3]
```

```
In [97]: sorted(['a','c','b'])
```

```
Out[97]: ['a', 'b', 'c']
```

```
In [98]: sorted("zero")
```

```
Out[98]: ['e', 'o', 'r', 'z']
```

```
In [99]: sorted((3,2,1))
```

```
Out[99]: [1, 2, 3]
```

```
In [102]: # sorted함수와 리스트 자료형의 sort함수의 차이점  
a = [3,1,2]  
result = a.sort() # sort함수로 a 리스트 정렬  
print(result) # 리턴값이 없기 때문에 None이 출력
```

```
None
```

```
In [103]: a
```

```
Out[103]: [1, 2, 3]
```

round(), str() 함수

18

```
In [104]: round(4.6)
```

```
Out[104]: 5
```

```
In [105]: round(4.2)
```

```
Out[105]: 4
```

```
In [106]: round(5.678, 2)  # 소수점 2자리까지만 반올림
```

```
Out[106]: 5.68
```

```
In [107]: str(3)
```

```
Out[107]: '3'
```

```
In [108]: str('hi')
```

```
Out[108]: 'hi'
```

```
In [109]: str('hi').upper()
```

```
Out[109]: 'HI'
```

tuple(), type() 함수

19

```
In [110]: tuple("abc")
```

```
Out[110]: ('a', 'b', 'c')
```

```
In [111]: tuple([1, 2, 3])
```

```
Out[111]: (1, 2, 3)
```

```
In [112]: tuple((1, 2, 3))
```

```
Out[112]: (1, 2, 3)
```

```
In [113]: type("abc")
```

```
Out[113]: str
```

```
In [114]: type([ ])
```

```
Out[114]: list
```

```
In [115]: # zip(*iterable)은 동일한 개수로 이루어진 자료형을 묶어 주는 역할을 하는 함수  
list(zip([1, 2, 3], [4, 5, 6]))
```

```
Out[115]: [(1, 4), (2, 5), (3, 6)]
```

```
In [116]: list(zip([1, 2, 3], [4, 5, 6], [7, 8, 9]))
```

```
Out[116]: [(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

```
In [117]: list(zip("abc", "def"))
```

```
Out[117]: [('a', 'd'), ('b', 'e'), ('c', 'f')]
```



내장함수
외장함수

- 명령 행에서 인수 전달하기 - sys.argv

```
C:/User/home>python test.py abc pey guido
```

```
# argv_test.py
import sys
print(sys.argv)
```

```
C:/doit/mymod>python argv_test.py you need python
['argv_test.py', 'you', 'need', 'python']
```

- 강제로 종료하기

```
>>> sys.exit()
```

In [119]: *# 객체의 형태를 그대로 유지하면서 파일에 저장하고 불러올 수 있게 하는 모듈*

```
import pickle
f = open("test.txt", 'wb')
data = {1: 'python', 2: 'you need'}
pickle.dump(data, f)
f.close()
```

In [120]: *f = open("test.txt", 'rb')*

```
data = pickle.load(f)
print(data)
```

```
{1: 'python', 2: 'you need'}
```

```
In [123]: # os 모듈 : 환경변수나 디렉토리, 파일등의 OS자원을 제어  
import os  
os.environ
```

```
In [124]: os.environ['PATH']
```

```
In [ ]: # 현재 디렉토리 위치를 변경할 수 있다.  
os.chdir("C:\\WINDOWS\\TEMP")
```

```
In [128]: # 현재 자신의 디렉토리 위치 리턴  
os.getcwd()
```

```
In [127]: # 시스템 자체의 프로그램이나 기타 명령어들을 호출  
os.system("dir")
```



```
In [ ]: # 파일 복사하기
import shutil
shutil.copy("src.txt", "dst.txt")
```

```
In [132]: # 디렉토리에 있는 파일들을 리스트로 만들기 (*,? 등의 메타 문자를 써서 원하는 파일만 읽어들이 수도 있다.)
import glob
glob.glob("C:/Windows/s*")
```

```
Out[132]: ['C:/Windows###SchCache',
           'C:/Windows###schemas',
           'C:/Windows###security',
           'C:/Windows###ServiceProfiles',
           'C:/Windows###ServiceState',
           .....
           .....
           'C:/Windows###System32',
           'C:/Windows###SystemApps',
           'C:/Windows###SystemResources',
           'C:/Windows###SysWOW64']
```

```
In [1]: # 파일을 임시로 만들어서 사용할 때 유용한 모듈.  
# tempfile.mktemp()는 중복되지 않는 임시파일의 이름을 무작위로 만들어서 리턴  
import tempfile  
filename = tempfile.mktemp()  
filename
```

```
Out[1]: 'C:\\\\Users\\\\user\\\\AppData\\\\Local\\\\Temp\\\\tmp4qik4n0'
```

```
In [4]: # tempfile.TemporaryFile() 임시 저장 공간으로 사용될 파일 객체를 선택  
# 이 파일은 기본적으로 바이너리 쓰기 모드(wb)를 갖는다.  
# f.close()가 호출되면 이 파일 객체는 자동으로 사라진다.  
import tempfile  
f = tempfile.TemporaryFile()  
f.close()
```

```
In [8]: ''' UTC(Universal Time Coordinated 협정 세계 표준시)로 현재 시간을 실수 형태로 리턴.  
1970년 1월 1일 0시 0분 0초를 기준으로 지난 시간을 초 단위로 리턴'''  
import time  
time.time()
```

Out[8]: 1544960275.6035604

```
In [9]: # time.time()에 의해서 반환된 실수값을 이용해서 연도, 월, 일, 시, 분, 초,.. 의 형태로  
# 바꾸어 주는 함수  
time.localtime(time.time())
```

Out[9]: time.struct_time(tm_year=2018, tm_mon=12, tm_mday=16, tm_hour=20, tm_min=37, tm_sec=56,
tm_wday=6, tm_yday=350, tm_isdst=0)



```
In [11]: # time.localtime에 의해서 반환된 튜플 형태의 값을 인수로 받아서 날짜와 시간을  
# 알아보기 쉬운 형태로 리턴하는 함수  
time.asctime(time.localtime(time.time()))
```

Out[11]: 'Sun Dec 16 20:38:53 2018'

```
In [12]: # time.asctime(time.localtime(time.time()))은 time.ctime()을 이용해 간편하게 표시.  
# asctime과 다른점은 ctime은 항상 현재 시간만을 리턴한다는 점.  
time.ctime()
```

Out[12]: 'Sun Dec 16 20:41:15 2018'

시간 포맷 코드

포맷코드	설명	예
%a	요일 줄임말	Mon
%A	요일	Monday
%b	달 줄임말	Jan
%B	달	January
%c	날짜와 시간을 출력함	06/01/01 17:22:21
%d	날(day)	[00,31]
%H	시간(hour)-24시간 출력 형태	[00,23]
%I	시간(hour)-12시간 출력 형태	[01,12]
%j	1년 중 누적 날짜	[001,366]
%m	달	[01,12]
%M	분	[01,59]

포맷코드	설명	예
%p	AM or PM	AM
%S	초	[00,61]
%U	1년 중 누적 주-일요일을 시작으로	[00,53]
%w	숫자로 된 요일	[0(일요일),6]
%W	1년 중 누적 주-월요일을 시작으로	[00,53]
%x	현재 설정된 로케일에 기반한 날짜 출력	06/01/01
%X	현재 설정된 로케일에 기반한 시간 출력	17:22:21
%Y	년도 출력	2001
%Z	시간대 출력	대한민국 표준시
%%	문자	%
%y	세기부분을 제외한 년도 출력	01

time.strftime('출력할 형식 포맷 코드', time.localtime(time.time()))

```
In [13]: time.strftime('%x', time.localtime(time.time()))
```

```
Out[13]: '12/16/18'
```

```
In [14]: time.strftime('%c', time.localtime(time.time()))
```

```
Out[14]: 'Sun Dec 16 20:51:38 2018'
```

```
In [17]: # 10이면 1초, 0.5면 0.5초  
for i in range(10):  
    print(i)  
    time.sleep(1)
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

```
In [19]: # 해당 년도의 전체 달력
import calendar
print(calendar.calendar(2018))
```

```
In [ ]: # calendar.calendar(2018)와 동일한 결과
calendar.prcal(2018)
```

```
In [20]: calendar.prmonth(2018, 12)
```

```
December 2018
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

```
In [21]: # 월요일:0, 화요일:1, 수요일:2, 목요일:3, 금요일:4, 토요일:5, 일요일:6
calendar.weekday(2018, 12, 31)
```

```
Out[21]: 0
```

```
In [22]: # monthrange(연도, 월) 함수는 입력받은 달의 1일이 무슨 요일인지와
# 그 달이 며칠까지 있는지를 튜플 형태로 리턴
calendar.monthrange(2018, 12)
```

```
Out[22]: (5, 31)
```

```
In [23]: # 0.0에서 1.0 사이의 실수 중에서 난수값을 리턴  
import random  
random.random()
```

Out[23]: 0.47771460565674784

```
In [24]: # 1에서 10 사이의 정수 중에서 난수값을 리턴  
random.randint(1, 10)
```

Out[24]: 4

```
In [25]: # 1에서 55 사이의 정수 중에서 난수값을 리턴  
random.randint(1, 55)
```

Out[25]: 3

In [27]: *# random_pop 함수: 리스트의 요소 중에서 무작위로 하나를 선택하여 꺼낸 다음 그 값을 리턴
물론 꺼내진 요소는 pop 메서드에 의해 사라진다.*

```
import random
def random_pop(data):
    number = random.randint(0, len(data)-1)
    return data.pop(number)

if __name__ == "__main__":
    data = [1, 2, 3, 4, 5]
    while data: print(random_pop(data))
```

5
2
4
1
3

In [29]: *# 리스트의 항목을 무작위로 섞고 싶을 때는 random.shuffle 함수를 이용*

```
def random_pop(data):
    number = random.choice(data)
    data.remove(number)
    return number
```

```
data = [1, 2, 3, 4, 5]
random.shuffle(data)
data
```

Out[29]: [2, 5, 3, 4, 1]


```
In [30]: # 자신의 시스템에서 사용하는 기본 웹 브라우저가 자동으로 실행되게 하는 모듈  
import webbrowser  
webbrowser.open("http://google.com")
```

Out[30]: True

```
In [31]: # 이미 웹 브라우저가 실행된 상태이더라도 새로운 창으로 해당 주소가 열리도록 한다.  
webbrowser.open_new("http://google.com")
```

Out[31]: True

```
In [31]: import threading
import time

def say(msg):
    while True:
        time.sleep(1)
        print(msg)

# 스레드 프로그램
for msg in ['you', 'need', 'python']:
    t = threading.Thread(target=say, args=(msg,))
    # daemon플래그를 설정하면 주 프로그램이 종료되는 순간 데몬 스레드도 함께 종료
    t.daemon = True
    t.start()

# 주 프로그램
for i in range(100):
    time.sleep(0.1)
    print(i)
```

```
In [31]: # MyThread로 생성된 객체의
# start 메서드를 실행할 때는 MyThread 클래스의 run 메서드가 자동으로 수행된다.
import threading
import time

class MyThread(threading.Thread):
    def __init__(self, msg):
        threading.Thread.__init__(self)
        self.msg = msg
        self.daemon = True

    def run(self):
        while True:
            time.sleep(1)
            print(self.msg)

for msg in ['you', 'need', 'python']:
    t = MyThread(msg)
    t.start()

for i in range(100):
    time.sleep(0.1)
    print(i)
```