

## ■ 외장함수

- 파이썬에서 제공되는 기본 라이브러리
- 내장함수와는 다르게 모듈을 불러들이는 import 과정을 거쳐야됨

sys	pickle	os	shutil
glob	tempfile	time	datetime
dateutil	calendar	random	webbrowser
namedtuple	defaultdict		

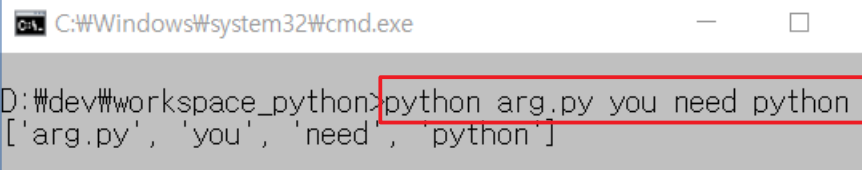
## ■ sys

- 파이썬이 제공하는 변수 및 함수를 직접 제어할 수 있게 해주는 모듈

- sys.argv

- 명령모드로 실행하면서 전달된 인자 확인

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



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt shows the command `python arg.py you need python` being executed. The output is `['arg.py', 'you', 'need', 'python']`. The command and its output are highlighted with a red rectangular box.

- sys.exit()

- 프로그램 강제 종료

```
import sys  
  
print('프로그램 시작')  
  
sys.exit()  
  
print('프로그램 종료')
```

프로그램 시작

## ■ sys

### ● sys.path

- 파이썬의 모듈이 저장되어 있는 위치를 나타냄
- 외부 모듈을 사용하기 위해 특정 위치 지정 가능

```
>>> import sys
>>> sys.path
['', 'C:\\Python36\\Lib\\idlelib', 'C:\\Python36\\python36.zip', 'C:\\Python36\\DLLs', 'C:\\Python36\\lib', 'C:\\Python36', 'C:\\Python36\\lib\\site-packages']
>>>
>>> sys.path.append('D:/dev/workspace_python')
>>> sys.path
['', 'C:\\Python36\\Lib\\idlelib', 'C:\\Python36\\python36.zip', 'C:\\Python36\\DLLs', 'C:\\Python36\\lib', 'C:\\Python36', 'C:\\Python36\\lib\\site-packages', 'D:/dev/workspace_python']
>>>
```

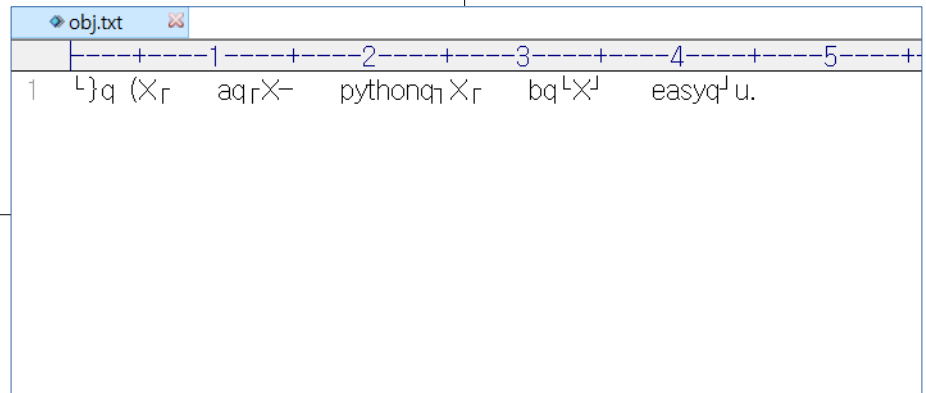
## ■ pickle

- 객체의 형태를 그대로 유지하면서 파일에 저장하고 불러올 수 있게 하는 모듈

- pickle.dump()

- Dictionary 객체의 형태를 그대로 유지하면서 저장

```
import pickle
file = open('obj.txt', 'wb')
data = {'a': 'python', 'b': 'easy'}
pickle.dump(data, file)
file.close()
```



- pickle.load()

- 객체의 형태를 그대로 유지하면서 불러오기

```
import pickle
file = open('obj.txt', 'rb')
data = pickle.load(file)
print(data)
file.close()
```

```
{'a': 'python', 'b': 'easy'}
```

## ■ pickle – 연습문제

- 좌표(x, y) 정보를 가지는 NowData클래스를 파일로 저장 후 다시 읽어서 정보 확인

```
import pickle

class NowData:
    def __init__(self):
        self.x = 0
        self.y = 0
    def move(self, x, y):
        self.x = x
        self.y = y
    def get_location(self):
        return self.x, self.y

data = NowData()
data.move(100, 200)
```

# 코드 작성

저장된 좌표 (100, 200)

## ■ OS

- 환경변수, 디렉토리, 파일 등의 운영체제 자원을 제어할 수 있게 해주는 모듈
- os.environ
  - 환경변수 확인

```
>>> import os
>>> os.environ
environ({'ALLUSERSPROFILE': 'C:\\ProgramData', 'ANDROID_SDK_HOME': 'D:\\dev\\Android\\avd', 'APPDATA': 'C:\\Users\\GGoReb\\AppData\\Roaming', 'COMMONPROGRAMFILES': 'C:\\Program Files\\Common Files', 'COMMONPROGRAMFILES(X86)': 'C:\\Program Files (x86)\\Common Files', 'COMMONPROGRAMW6432': 'C:\\Program Files\\Common Files', 'COMPUTERNAME': 'DESKTOP-VMV850T', 'COMSPEC': 'C:\\Windows\\system32\\cmd.exe', 'FPS_BROWSER_APP_PROFILE_STRING': 'Internet Explorer', 'FPS_BROWSER_USER_PROFILE_STRING': 'Default', 'GTK_BASEPATH': 'C:\\Program Files (x86)\\GtkSharp\\2.12', 'HOME': 'C:\\Users\\GGoReb', ... , 'WINDIR': 'C:\\Windows'})

>>> os.environ['HOME']
'C:\\Users\\GGoReb'
```

## ■ OS

### ● os.chdir()

- 현재 디렉토리의 위치 변경

```
>>> import os  
>>> os.chdir('C:/Windows')
```

### ● os.getcwd()

- 현재 지정된 디렉토리 확인

```
>>> import os  
>>> os.getcwd()  
'C:\\Windows'
```

### ● os.system()

- 시스템(운영체제 또는 외부 프로그램) 명령어 호출하기

```
>>> import os  
>>> os.system('dir')
```

## ■ OS

### ● os.popen()

– 시스템(운영체제 또는 외부 프로그램) 명령어 호출 결과 받기

```
import os
```

```
os.chdir('d:/dev/workspace_python')
```

```
content = os.popen('dir/w')
```

```
while True:
```

```
    line = content.readline()
```

```
    if not line:
```

```
        break
```

```
    print(line)
```

```
content.close()
```

D 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: 8818-267F

d:\dev\workspace\_python 디렉터리

[.]	[..]
03_01_if.py	03_02_if_else.py
03_03_if_elif_else.py	03_04_while.py
03_05_for.py	03_06_for.py
03_07_for_range.py	03_08_for_gugudan.py
03_09_list_comprehension.py	03_10_list_comprehension.py
04_01_function.py	04_02_function.py
04_03_function.py	04_04_function.py
04_05_function.py	04_06_return.py
04_07_return.py	04_08_function.py
04_09_var_range.py	04_10_lambda.py
04_11_lambda.py	04_12_lambda.py
05_01_input.py	05_02_print.py



## ■ OS

### ● os.mkdir()

- 디렉토리를 만들어주는 함수

```
os.mkdir('D:/python')
```

### ● os.rmdir()

- 디렉토리를 삭제해주는 함수 (디렉토리가 비어있어야 가능)

```
os.rmdir('D:/python')
```

### ● os.unlink()

- 파일을 지워주는 함수

```
os.unlink('D:/python/python.txt')
```

### ● os.rename()

- 디렉토리 또는 파일의 이름을 변경해주는 함수

```
os.rename('D:/python/sub', 'D:/python/main')
```

## ■ OS

- `os.path.abspath()` : 절대 경로

```
os.path.abspath('.')
```

```
'C:\\\\Users\\\\GGoReb\\\\work_python'
```

- `os.path.basename()` : 기본 경로

```
os.path.basename('C:/Users/GGoReb/work_python')
```

```
'work_python'
```

- `os.path.join()` : 경로 합치기

```
os.path.join('c:/upload', 'file.txt')
```

```
'c:/upload\\\\file.txt'
```

- `os.path.split()` : 기본 경로 분리

```
os.path.split(os.path.abspath('.'))
```

```
('C:\\\\Users\\\\GGoReb', 'work_python')
```

- `os.path.splitdrive()` : 드라이브 분리

```
os.path.splitdrive(os.path.abspath('.'))
```

```
('C:', '\\\\Users\\\\GGoReb\\\\work_python')
```

- `os.path.splitext()` : 확장자 분리

```
os.path.splitext('C:/Users/GGoReb/work_python/test.9.png')
```

```
('C:/Users/GGoReb/work_python/test.9', '.png')
```

## ■ os - 연습문제

○ 아래와 같은 구조 디렉토리와 파일을 생성 (os)

```
python_dir/
```

```
  a/
```

```
    a.txt
```

```
  b/
```

```
    b.txt
```

```
# 코드 작성
```

## ■ shutil

- 파일을 복사해주는 모듈
- shutil.copy()

```
>>> import shutil
>>> shutil.copy('d:/python/source.txt', 'd:/python/target.txt')
```

## ■ glob

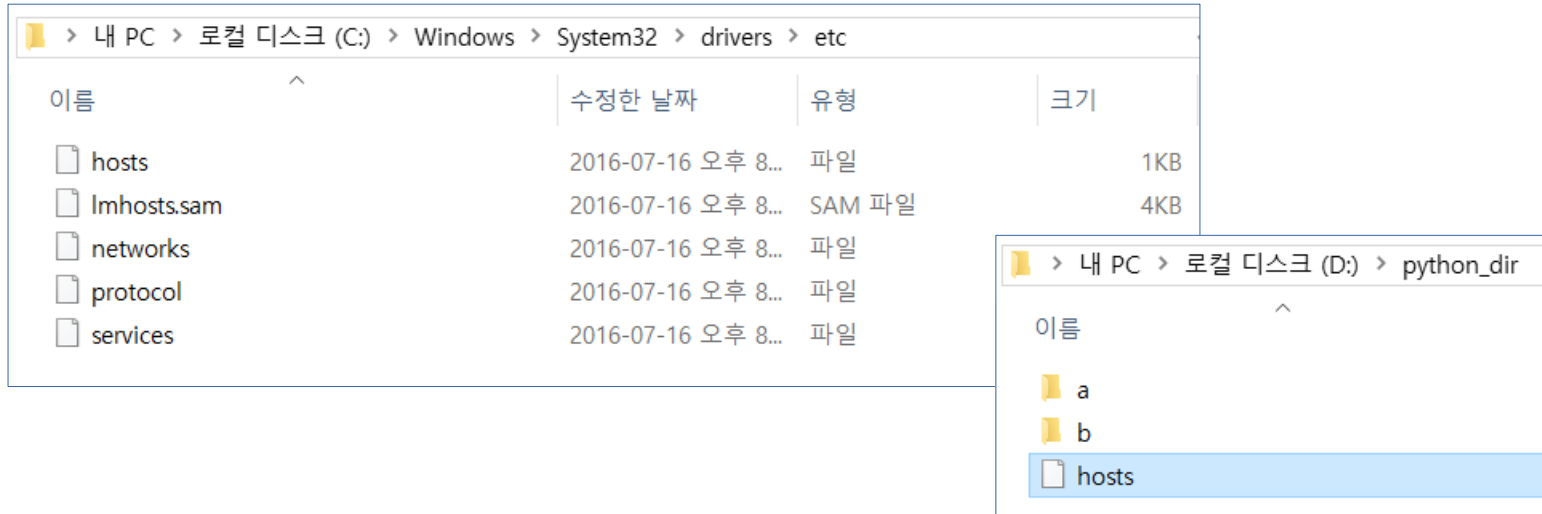
- 디렉토리에 있는 파일들을 리스트로 만들어주는 모듈
- glob.glob()

```
import glob
fileList = glob.glob('C:/Windows/a*')
print(fileList)
```

```
['C:/Windows###ACU.ico', 'C:/Windows###addins', 'C:/Windows###AhnInst.log', 'C:/Windows###appcompat', 'C:/Windows###AppPatch', 'C:/Windows###AppReadiness', 'C:/Windows###assembly']
```

## ■ shutil / glob - 연습문제

### 1. Windows/System32/drivers/etc/hosts 파일을 python\_dir 디렉토리로 복사 (shutil)



### 2. C:\Users\W[사용자명]\Miniconda3\Lib의 파일 중 확장자가 py인 파일의 개수 확인하기 (glob)

파일개수 171

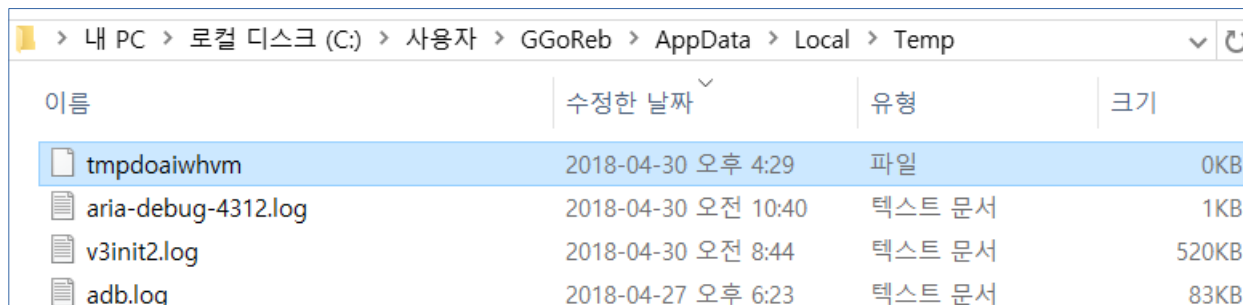
## ■ tempfile

- 파일을 임시로 만들어서 사용하게 해주는 모듈
- `tempfile.mktemp()`
  - 중복되지 않는 임시 파일의 이름을 무작위로 만들어서 돌려주는 함수

```
>>> import tempfile
>>> filename = tempfile.mktemp()
>>> filename
'C:\\Users\\WWGGoReb\\AppData\\Local\\Temp\\Wtmppv7v8b8'
```

- `tempfile.TemporaryFile()`
  - 임시 저장공간으로 사용될 파일 객체를 돌려주는 함수

```
>>> import tempfile
>>> file = tempfile.TemporaryFile()
>>> file.close()
```



이름	수정한 날짜	유형	크기
tmpdoaiwhvm	2018-04-30 오후 4:29	파일	0KB
aria-debug-4312.log	2018-04-30 오전 10:40	텍스트 문서	1KB
v3init2.log	2018-04-30 오전 8:44	텍스트 문서	520KB
adb.log	2018-04-27 오후 6:23	텍스트 문서	83KB

## ■ time

- 시간과 관련된 여러 기능을 제공하는 모듈
- time.time()
  - UTC(Universal Time Coordinated 협정 세계 표준시)를 이용하여 현재 시각을 실수 형태로 돌려주는 함수
  - 1970년 1월 1일 0시 0분 0초를 기준으로 지난 시간을 초 단위로 표시

```
>>> import time
>>> time.time()
1525075157.2935598
```

- time.localtime()
  - time.time() 으로 반환된 값을 이용하여 년, 월, 일, 시, 분, 초 등의 Tuple 형태로 변환해주는 함수

```
>>> date = time.time()
>>> date2 = time.localtime(date)
>>> date2
time.struct_time(tm_year=2018, tm_mon=4, tm_mday=30, tm_hour=17, tm_min=3,
tm_sec=30, tm_wday=0, tm_yday=120, tm_isdst=0)
>>> nowDate = str(date2.tm_year) + '/' + str(date2.tm_mon) + '/' + str(date2.tm_mday)
>>> nowDate
'2018/4/30'
```

## ■ time

### ● time.asctime()

- time.localtime() 으로 반환된 값을 날짜와 시간을 알아보기 쉬운 형태로 돌려주는 함수

```
>>> time.asctime( time.localtime( time.time() ) )  
'Mon Apr 30 17:48:00 2018'
```

### ● time.ctime()

- time.asctime() 과 같은 내용을 출력해주는 함수 (현재 시각만 출력)

```
>>> time.ctime()  
'Mon Apr 30 17:48:00 2018'
```



## ■ time

### ● time.strftime()

- Tuple 형태의 time.localtime(time.time()) 과 함께 출력 형태를 지정하여 세밀하게 표현해주는 함수

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

포맷 코드	설명	출력 예	포맷 코드	설명	출력 예
%Y	년	2002	%y	년	02
%m	월	01 - 12	%B	월(영어)	January
			%b	월(영어)	Jan
%d	일	01 - 31			
%H	시	00 - 23	%I	시	01 - 12
%M	분	00 - 59			
%S	초	00 - 61			
%w	요일	0 - 6	%A	요일	Monday
			%a	요일	Mon
%p	오전, 오후	AM, PM			
%c	날짜와 시간	11/11/11 11:11:11			

## ■ time

### ● time.sleep()

- 일정시간 동안 멈추도록 해주는 함수, 주로 반복문 안에서 사용

```
import time
```

```
for i in range(1, 11):
```

```
    print(i)
```

```
    time.sleep(1)
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

## ■ time – 연습문제

1. time 모듈을 이용하여 실행 결과와 같이 출력될 수 있도록 코드 작성

```
# 코드 작성
```

```
2018-09-27 22:13:31
```

2. 현재 일자에서 100일 뒤의 날짜를 실행 결과와 같은 형식으로 출력

```
# 코드 작성  
# time.time() + 초  
# 1시간 = 60초 * 60분
```

```
2019-01-05 22:13:31
```

## ■ datetime

### ● datetime.datetime.now()

– 현재 시각을 출력해주는 함수

```
import datetime
```

```
dt = datetime.datetime.now()  
print(dt)
```

```
datetime.datetime(2018, 11, 22, 17, 26, 44, 972778)
```

– 각 정보 확인

```
print(dt.year, dt.month, dt.day)  
print(dt.hour, dt.minute, dt.second)  
print(dt.microsecond)  
print(dt.weekday()) # 0:월, 1:화, 2:수, 3:목, 4:금, 5:토, 6:일
```

```
2018 11 22  
17 26 44  
972778  
3
```

## ■ datetime / dateutil

### ● strftime()

- 형식을 지정하여 출력해주는 함수

```
datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S %A')
```

```
'2018-11-22 18:04:39 Thursday'
```

### ● strptime()

- 날짜 형태의 문자열을 datetime으로 변환

```
datetime.datetime.strptime('2018-11-11 12:12', '%Y-%m-%d %H:%M')
```

```
datetime.datetime(2018, 11, 11, 12, 12)
```

### ● dateutil.parser.parse()

- 날짜 형태의 문자열을 datetime으로 변환

```
from dateutil.parser import parse  
parse('2018-11-11 12:12')
```

```
datetime.datetime(2018, 11, 11, 12, 12)
```

## ■ datetime

### ● 날짜 간 연산

```
dt1 = datetime.datetime(2012, 3, 25)
dt2 = datetime.datetime.now()
result = dt2 - dt1
print(result)
```

```
2433 days, 18:10:25.708335
```

#### – 각 정보 출력

```
print(result.days)
print(result.seconds)
print(result.microseconds)
```

```
2433
64028
222013
```

#### – timedelta()

```
now_time = datetime.datetime.now()
add_time = datetime.timedelta(days=100)
print(now_time + add_time)
```

```
2019-03-02 18:18:07.715992
```

## ■ calendar

- 달력을 볼 수 있게 해주는 모듈
- calendar.calendar(), calendar.prcal()

```
>>> import calendar
```

```
>>> print(calendar.calendar(2018)) #>>> calendar.prcal(2018)
```

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

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

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

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

## ■ calendar

### ● calendar.prmonth()

– 년, 월을 입력받은 후 해당 월의 달력을 보여주는 함수

```
>>> calendar.prmonth(2012, 3)
```

March 2012						
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	

### ● calendar.weekday()

– 년, 월, 일을 입력받은 후 날짜에 해당하는 요일 정보를 돌려주는 함수

```
>>> calendar.weekday(2012, 3, 25) # 일요일  
6
```

### ● calendar.monthrange()

– 년, 월을 입력받은 후 해당 월의 1일의 요일과 마지막 일을 돌려주는 함수

```
>>> calendar.monthrange(2012, 3)  
(3, 31)
```



## ■ calendar - 연습문제

### 1. 2017년 5월의 달력 출력

# 코드 작성

```
May 2017
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
```

### 2. 다음 딕셔너리를 이용하여 2017년 5월 9일의 요일을 출력

```
import calendar
week_dict = { 0 : '월', 1 : '화', 2 : '수', 3 : '목', 4 : '금', 5 : '토', 6 : '일' }
```

화

# 코드 작성

### 3. 2004년 8월의 달력과 요일이 월요일인 날짜 출력

# 코드 작성  
# 해당 월의 마지막 날짜를 구한 후  
# 반복문과 조건문을 이용

```
August 2004
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
```

2  
9  
16  
23  
30

## ■ random

- 난수를 만들어주는 모듈

- random.random()

–  $0.0 \leq N < 1.0$  범위의 난수값(실수)을 돌려주는 함수

```
>>> import random
>>> random.random()
0.5031093544872511
```

- random.randint(A, B)

–  $A \leq N \leq B$  범위의 난수값(정수)을 돌려주는 함수

```
>>> import random
>>> random.randint(0, 3)
3
```

## ■ random

### ● random.choice()

– 반복이 가능한 자료형에서 무작위로 요소 하나를 돌려주는 함수

```
>>> import random
>>> random.choice([1, 2, 3, 4, 5])
1
>>> random.choice('12345')
'4'
```

### ● random.shuffle()

– List 자료형의 요소 순서를 무작위로 섞어주는 함수

```
>>> import random
>>> list = [1, 2, 3, 4, 5]
>>> random.shuffle(list)
>>> list
[1, 5, 4, 2, 3]
```

## ■ random - 연습문제

### 1. 0.0 보다 크거나 같고 5.0 보다 적은 난수(실수) 생성

```
# 코드 작성
# round() 함수를 이용하여 소수점 둘째 자리까지만 표시
# ex) round(3.1555, 2) → 3.16
```

2.07

0.59

### 2. 주사위 두개를 던진 결과를 출력하고 두 눈이 같으면 종료되는 코드 작성

```
count = 0

# 코드 작성

print('주사위를 던진 횟수', count)
```

```
6 4
6 3
5 2
6 3
1 1
주사위를 던진 횟수 5
```

### 3. 1 ~ 25 빙고판 만들기

```
# bingo 1 ~ 25 숫자 입력
bingo = list()
for i in range(1, 26):
    if i < 10: i = '0' + str(i)
    bingo.append(str(i))
print(bingo)

# 코드 작성
```

```
['01', '02', '03', '04', '05',
'06', '07', '08', '09', '10',
'11', '12', '13', '14', '15',
'16', '17', '18', '19', '20',
'21', '22', '23', '24', '25']
```

```
20 15 16 19 13
25 01 11 14 09
04 08 12 18 17
10 06 03 07 21
22 23 05 24 02
```

## ■ webbrowser

- 시스템에 지정된 기본 웹 브라우저를 실행시키는 모듈

- webbrowser.open()

- 웹 브라우저를 실행시키면서 입력된 URL로 이동하는 함수 (기존 창)

```
>>> import webbrowser
>>> webbrowser.open('http://ggoreb.com')
True
```

- webbrowser.open\_new()

- 웹 브라우저를 실행시키면서 입력된 URL로 이동하는 함수 (새 창)

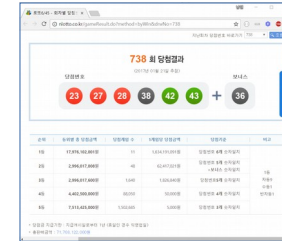
```
>>> import webbrowser
>>> webbrowser.open_new('http://ggoreb.com')
True
```

## ■ webbrowser - 연습문제

- 1 ~ 800 사이의 난수를 생성하고 브라우저를 이용하여 로또 추천결과 웹페이지를 확인하는 코드 작성 (random, webbrowser)

```
url = 'http://nlotto.co.kr/gameResult.do?method=byWin&drwNo='
```

# 코드 작성



## ■ namedtuple

- Tuple의 값을 속성으로 제어할 수 있게 해주는 함수
- Tuple 자료형만을 이용한 값 접근

```
a = ('Galaxy6', 100, 'Samsung')  
b = ('G6', 100, 'LG')  
c = ('MI6', 90, 'Xiaomi')  
  
for phone in [a, b, c]:  
    print('name - %s' % phone[0])  
    print('price - %s' % phone[1])  
    print('corp - %s' % phone[2])
```

```
name - Galaxy6  
price - 100  
corp - Samsung  
  
name - G6  
price - 100  
corp - LG  
  
name - MI6  
price - 90  
corp - Xiaomi
```

## ■ namedtuple

### ● 속성으로 제어하기 위해 클래스 활용

```
class Phone:
    def __init__(self, name, price, corp):
        self.name = name
        self.price = price
        self.corp = corp

a = Phone(name='Galaxy6', price=100, corp='Samsung')
b = Phone(name='G6', price=100, corp='LG')
c = Phone(name='MI6', price=90, corp='Xiaomi')

for phone in [a, b, c]:
    print('name - %s' % phone.name)
    print('price - %s' % phone.price)
    print('corp - %s' % phone.corp)
```

```
name - Galaxy6
price - 100
corp - Samsung
```

```
name - G6
price - 100
corp - LG
```

```
name - MI6
price - 90
corp - Xiaomi
```



## ■ namedtuple

### ● namedtuple 사용

```
from collections import namedtuple
```

```
Phone = namedtuple('Phone', ['name', 'price', 'corp'])
```

```
a = Phone(name='Galaxy6', price=100, corp='Samsung')
```

```
b = Phone(name='G6', price=100, corp='LG')
```

```
c = Phone(name='MI6', price=90, corp='Xiaomi')
```

```
for phone in [a, b, c]:
```

```
    print('name - %s' % phone.name)
```

```
    print('price - %s' % phone.price)
```

```
    print('corp - %s' % phone.corp)
```

```
name - Galaxy6  
price - 100  
corp - Samsung
```

```
name - G6  
price - 100  
corp - LG
```

```
name - MI6  
price - 90  
corp - Xiaomi
```

## ■ namedtuple – 연습문제

- 아래의 Student 클래스 대신 namedtuple을 이용하여 동일한 결과가 나오도록 작성

```
class Student:
    def __init__(self, name, score):
        self.name = name
        self.score = score
```

```
a = Student('홍길동', 30)
```

# 코드 작성

```
print(a.name)
print(a.score)
```

홍길동  
30

## ■ defaultdict

- Dictionary에 기본값을 할당해주는 함수
- Dictionary 사용 시 값 누적 (등장한 단어 개수 확인)

```
text = 'Life is too short, you need python'
```

```
d = dict()
```

```
print()
```

```
for t in text:
```

```
    if t in d: # dictionary에 key가 있다면 기존값 + 1
```

```
        d[t] += 1
```

```
    else:      # dictionary에 key가 없다면 초기값 1 입력
```

```
        d[t] = 1
```

```
print(d.items())
```

```
dict_items([('L', 1), ('i', 2), ('f', 1),  
, ('e', 3), (' ', 6), ('s', 2), ('t', 3),  
, ('o', 5), ('h', 2), ('r', 1), (',', 1),  
, ('y', 2), ('u', 1), ('n', 2), ('d', 1),  
, ('p', 1)])
```

## ■ defaultdict

### ● defaultdict를 사용한 값 누적 (등장한 단어 개수 확인)

```
from collections import defaultdict
```

```
text = 'Life is too short, you need python'
```

```
d = defaultdict(int) # 사용할 기본값의 자료형 지정
```

```
print()
```

```
for t in text:
```

```
    d[t] += 1 # dictionary에 key 존재여부 무관
```

```
print(d.items())
```

```
dict_items([('L', 1), ('i', 2), ('f', 1),  
, ('e', 3), (' ', 6), ('s', 2), ('t', 3),  
, ('o', 5), ('h', 2), ('r', 1), ('.', 1),  
, ('y', 2), ('u', 1), ('n', 2), ('d', 1),  
, ('p', 1)])
```

## ■ defaultdict

### ● defaultdict를 사용한 리스트 요소 추가 (상품별 매출 확인)

```
from collections import defaultdict
```

```
info = [  
    ('가', 100), ('나', 110), ('다', 120), ('라', 130),  
    ('가', 200), ('다', 210), ('라', 220)  
];
```

```
d = defaultdict(list) # 사용할 기본값의 자료형 지정
```

```
for k, v in info:  
    d[k].append(v)
```

```
print(d.items())
```

```
dict_items([('가', [100, 200]),  
            ('나', [110]), ('다', [120, 210]),  
            ('라', [130, 220])])
```