외장함수

sys

- 파이썬 인터프리터가 제공하는 변수들과 함수들을 직접 접근하여 제어할 수 있게 해 주는 모듈
- **sys.argv**
  - 파이썬을 실행하기 위해 넣은 인자 리스트

```
import sys
print(sys.argv)
```

- **sys.exit()**
  - 프로그램을 중단시키는 명령. Ctrl-Z나 Ctrl-D의 명령과 같은 기능 수행
- **sys.path**
  - 파이썬 모듈이 저장되어 있는 위치, 이 위치에 있는 파이썬 모듈들은 경로에 상관없이 불러올 수 있음

```
print(sys.path)
sys.path.append("경로")
```

pickle

- 객체의 형태를 그대로 유지하게 하여 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()
```

```
import pickle
f = open("test.txt", 'rb')
data = pickle.load(f)
print(data)
```

os

- 각 시스템별 os의 기능을 수행할 수 있게 해 주는 모듈
- **os.environ**
  - 시스템 환경변수 값을 보여주는 모듈
  - 딕셔너리 객체로 반환함

```
import os
print(os.environ)
```

- 디렉토리에 관한 것들
  - os.chdir('c:/')
    - 현재 디렉토리의 위치 변경
  - os.getcwd()
    - 현재 자신의 디렉토리 위치를 반환
- 시스템 명령
  - os.system('dir') / os.startfile(path)
    - 현재 디렉토리에서 dir을 실행함 / 파이썬을 멈추지 않고 실행
  - os.popen('dir')
    - 실행시킨 명령의 결과를 파일에 저장

```
f = os.popen("dir")
print(f.read())
```

os

- os.mkdir(디렉토리) / os.mkdirs(디렉토리)
  - 디렉토리를 생성함 / 전달된 디렉토리를 재귀적으로 생성
- os.rmdir(디렉토리) / os.removedirs(디렉토리)
  - 디렉토리가 비어있을 경우 삭제함 / 반복적으로 삭제
- os.remove(파일) / os.unlink(파일)
  - 파일을 지움
- os.rename(src, dst) / os.renames(src, dst)
  - Src라는 이름의 파일을 dst라는 이름으로 바꾸거나 이동 / 필요한 디렉토리 생성
- os.walk(path)
  - 디렉토리 내에있는 경로, 하위 디렉토리, 파일 리스트를 반환

```
for (path, dir, files) in os.walk('.'):
    for filename in files:
        print(filename)
```

- 파일 복사
  - shutil : 파일을 복사해 주는 모듈

```
import shutil
shutil.copy("src.txt", "dst.txt")
```

os.path

- 파일 경로를 생성 및 수정하고 파일 정보를 쉽게 다룰 수 있는 모듈
- abspath(path)
  - 현재 경로를 절대 경로로 바꿔서 반환함

```
abspath('python.exe')
```

- basename(path)
  - 입력 받은 경로의 기본이름을 반환함
  - abspath와 반대 개념

```
basename('c:/python34/python.exe')
```

- commonprefix(path_list)
  - 입력 받은 path_list로 부터 공통적인 prefix를 추출해서 반환

```
commonprefix(['c:/python34/python.exe','c:/python34/pythonw.exe'])
```

os.path

- dirname(path)
  - 입력받은 파일이나 디렉토리의 경로를 반환

```
dirname('c:/python34/python.exe')
```

- exists(path)
  - 입력받은 파일이나 디렉토리가 존재하면 True, 아니면 false 반환
    - 읽기 권한이 없는 경우 false 반환

```
exists('c:/python34/python.exe')
```

- expanduser(path)
  - ~ 기호대신 사용자 홈디렉토리로 대체

```
expanduser('~₩₩python.exe')
```

- expandvars(path)
  - 환경변수로 대체하여 경로 생성

```
expandvars('$JAVA_HOME₩₩python.exe')
```

os.path

- getatime(path) / getmtime(path) / getctime(path)
  - Path에 있는 파일의 접근시간, 변경시간, 생성시간을 반환
  - 1970년 1월 1일을 기준으로 초단위로 반환

```
time.gmtime(os.path.getatime('python.exe'))
```

- isfile(path) / isdir(path)
  - 파일인지 디렉토리 인지 판단
  - 경로가 없어도 false 반환

- join(path1, path2, …)
  - 입력 받은 경로를 연결함
  - 중간에 절대 경로가 나오면 다시 연결 함

```
join('c:₩₩python34₩₩','Greenjoa', 'c:₩₩','test.py')
```

os.path

- split(path)
  - 입력 받은 경로의 디렉토리 부분과 파일 부분으로 나눔
- splitdrive(path)
  - 입력 받은 경로를 드라이브 부분과 나머지 부분으로 나눔

```
splitdrive('C:₩₩Python34₩₩python.exe')
```

- splittext(path)
  - 입력받은 경로를 확장자 부분과 그외 부분으로 나눔

```
splitext('C:₩₩Python34₩₩python.exe')
```

glob

- 디렉토리에 있는 파일 리스트를 반환하는 모듈
  - dir 이나 ls 와 유사한 기능을 제공함
- glob(pathname)
  - 해당 디렉토리 내의 파일들을 읽어서 반환함
  - *, ?, [ ]를 사용한 문자열 비교가 가능함

```
glob.glob('*.exe')
glob.glob('??.*')
glob.glob('./[0-9].*')
glob.glob(abspath('.'+'₩₩*.exe'))
```

- iglob(path)
  - glob와 동일한 동작을 수행하지만, iterator를 반환함
  - 결과데이터가 매우 많은 경우 유용

```
for i in glob.iglob('*'):
    print (i)
```

예제

```
import glob, os.path
ndir = nfile = 0
def traverse(dir, depth):
    global ndir, nfile
    for obj in glob.glob(dir + '/*'):
        if depth == 0:
            prefix = '|--'
        else:
            prefix = '|' + '   ' * depth + '|--'
        if os.path.isdir(obj):
            ndir += 1
            print(prefix + os.path.basename(obj))
            traverse(obj, depth+1)
        elif os.path.isfile(obj) :
            nfile +=1
            print(prefix + os.path.basename(obj))
        else:
            print(prefix + 'unknown object:',obj)

if __name__ == '__main__':
    traverse('..', 0)
    print('₩n',ndir,'directories,',nfile, 'files')
```

임시파일(tempfile)

- 임시파일이나 디렉토리를 만들어서 쓸 때 유용한 모듈
- TemporaryFile()
  - 임시 파일을 만들어서 반환, 파일을 close하면 자동으로 삭제됨
  - 기본 모드가 'w+b'임, delete=True

```
import tempfile
with tempfile.TemporaryFile('w+') as fp:
    fp.write('Hello world!')
    fp.seek(0)
    data = fp.read()
    print(data)
```

생성된 파일명 : fp.name

- TemporaryDirectory()
  - 임시 디렉토리를 만들어서 반환

```
with tempfile.TemporaryDirectory() as tmpdirname:
    print('created temporary directory', tmpdirname)
```

Time

- 시간 표현하는 모듈
- 컴퓨터 시간 표현
  - 타임스탬프 (TimeStamp)
    - 컴퓨터에서 시간을 측정하는 방법으로 1970년 1월 1일 자정 이후 초단위로 측정한 절대시간
  - 협정세계시 (UTC, Universal Time Coordinated)
    - 1972년부터 시행된 국제 표준시
  - 그리니치 평균시 (GMT, Greenwich, Mean Time)
    - 런던 그리니치 천문대의 자오선상에서의 평균 태양시
  - 지방표준시 (LST, Local Standard Time)
    - UTC를 기준으로 경도 15도마다 1시간 차이가 발생하는 시간
  - 일광절약 시간제 (DST, Daylight Saving Time)
    - 서머타임으로 알고 있는 것으로, 에너지 절약을 목적으로 시간을 한 시간씩 앞당기거나 뒤로 미루는 제도

Time

- 속성
  - Tm_year : 년도 (ex, 2015)
  - Tm_mon : 월 (1~12)
  - Tm_mday : 일 (1~31)
  - Tm_hour : 시 (0~23)
  - Tm_min : 분 (0~59)
  - Tm_sec : 초 (0~61)
  - Tm_wday : 각 요일을 숫자로 나타냄 (월요일 0)
  - Tm_yday : 1월 1일부터 오늘까지 누전된 날짜를 반환
  - Tm_isdst : 일광절약 시간제(서머타임) (0,1,-1)
- 형식 지시자

%y	연도를 축약 표시	%H	24시 기준 시	%A	축약되지 않은 요일
%Y	연도를 축약 않고 표시	%I	12시 기준 시	%w	요일을 숫자로 표시(일요일 0)
%b	축약된 월 이름	%M	분	%j	1월 1일부터 누적된 날짜
%B	축약되지 않은 월 이름	%s	초		
%m	숫자로 표현한 월(1~12)	%p	오전/오후		
%d	일(01~31)	%a	축약된 요일		

time

- 함수
  - Time()
    - 1970년 1월 1일 자정 이후로 누적된 초를 float 단위로 반환
  - Ctime(sec)
    - 입력된 초를 스트링 객체로 반환
  - Gmtime([secs])
    - 입력된 초를 변환하여 UTC기준의 struct_time 시퀀스 객체로 반환
    - 인자가 입력되지 않은 경우, time()을 이용하여 현재 시간을 반환

```
t = time.gmtime(1234567)
```

  - Localtime([secs])
    - 입력된 초를 변환하여, 지방표준시 기준의 struct_time 시퀀스 객체를 반환
    - 인자가 입력되지 않은 경우, time()을 이용하여 현재 시간을 반환
  - Asctime([t])
    - Struct_time 시퀀스 객체를 인자로 받아서 'Sun Mar 15 18:49:28:2009'와 같은 형태로 시간을 반환

```
time.asctime(t)
```

  - Mktime(t)
    - 지방표준시이 struct_time 시퀀스 객체를 인자로 받아서 time()과 같은 누적된 초를 반환

time

- Sleep(sec)
  - 현재 동작 중인 프로세스를 주어진 초만큼 정지

```
t1 = time.time()
time.sleep(10)
t2 = time.time()
spendtime = t2-t1
```

- Strftime(format,[,t])
  - Struct_time 객체를 사용자가 정의한 형식으로 변경하여 문자열로 반환

```
time.strftime("%B %dth %A %I:%M", time.localtime())
time.strftime("%Y-%m-%d %I:%M", time.localtime()
```

- Strptime(string[, format])
  - 사용자가 정의한 형식 문자열을 struct_time 객체로 변환

```
time1 = time.ctime(1234567)
t = time.strptime(time1)
```

Calendar

- 파이썬에서 달력을 볼 수 있게 해주는 모듈
- Calendar(year)
  - 전체 달력을 볼 수 있음

```
import calendar
calendar.calendar(2000)
calendar.prcal(2000)
```

- prmonth(year, month)
  - 해당 년도의 달만 보여줌
- weekday(year, month, day)
  - 해당하는 날짜의 요일 정보를 반환
  - 월요일 0 ~ 일요일 6
- monthrange(year,month)
  - 해당하는 달의 1일이 무슨 요일이고, 몇 일까지 있는지에 대한 정보를 튜플로 반환

Random

- 난수를 발생시키는 모듈
- Random()
  - 0과 1.0사이의 실수값 중에서 난수값을 반환
- Randint(1,10)
  - 범위 내에서의 난수값을 반환
- Randrange(10) / randrange(0,101, 2)
  - 범위 내의 숫자 반환 (0~9 , 0~100 /2의 배수)
- Choice([])
  - 입력 받은 리스트에서 무작위로 하나를 선택하여 반환
- Suffle([])
  - 무작위로 섞어 주는 함수
- Sample(population, k)
  - Population 중에 k개 선택

random.sample(range(100), 5)

```
import random
weighted_choices = [('Red', 3), ('Blue', 2), ('Yellow', 1), ('Green', 4)]
population = [val for val, cnt in weighted_choices for i in range(cnt)]
```

Webbrowser

- 기본 웹브라우저가 자동으로 실행되게 하는 모듈

```
import webbrowser
url = 'http://google.com'
webbrowser.open_new_tab(url)
webbrowser.open_new(url)
```