

파이썬 프로그래밍

파일과 디렉토리 다루기



한국기술교육대학교
온라인평생교육원

■ 파일 다루기

1. 파일 목록 얻기

- `os.listdir('경로')`
 - 디렉토리 안에 들어 있는 각 파일 목록 반환

```
import os
```

```
print os.listdir('.') # 현재 디렉토리의 파일 목록 얻기  
print
```

```
print os.listdir('../') # 현재 디렉토리의 부모 디렉토리의 파일 목록 얻기
```

```
['.DS_Store', '.git', '.gitignore', '.ipynb_checkpoints', 'example', 'files', 'images',  
'python01.ipynb', 'python02.ipynb', 'python03.ipynb', 'python04.ipynb',  
'python05.ipynb', 'python06.ipynb', 'python07.ipynb', 'python08.ipynb',  
'python09.ipynb', 'python10.ipynb', 'python11.ipynb', 'python12.ipynb',  
'python13.ipynb', 'python14.ipynb', 'python15.ipynb', 'python16.ipynb',  
'python17.ipynb', 'python18.ipynb', 'python19.ipynb', 'README.md', 'removeme.txt',  
'sample.txt', 't1.txt', 'teststring2.pyc']
```

```
['.DS_Store', 'calendar', 'calendar-dev', 'ipython', 'jms', 'jms-scheduler', 'mySpringFrame',  
'python-e-learning', 'python-e-learning-movie', 'reviewer-recommend']
```

- `os`(운영체제) 안에 파일을 다루는 다양한 매소드 존재
- `os.listdir()` → 디렉토리에 존재하는 전체내용을 list로 반환
- 라인 맨 앞 'd' 존재하면 폴더(디렉토리)를 의미
- `rwx` → 디렉토리의 소유자가 가지고 있는 읽기, 쓰기, 실행 권한
- `r-x` → 그룹에 속한 사람이 이용할 수 있는 권한 (읽기, 실행)
- 마지막 `r-x` → 그룹 외 사람들이 이용할 수 있는 권한(읽기, 실행)
- `os.listdir(경로명)` → 경로명에 . 이므로 현재 디렉토리
- `../` → 부모 디렉토리
- `os.path.isfile(경로명)`

■ 파일 다루기

2. 파일 종류 알아보기

- os.path 모듈의 다음 함수들은 파일의 종류를 판단하여 True 또는 False를 반환한다.
 - isfile(filepath) : 순수 파일이면 True
 - isdir(filepath) : 디렉토리이면 True
 - islink(filepath) : 심볼릭링크이면 True

```
import os
def filetype(fpath):
    print fpath, ':',
    if os.path.isfile(fpath):
        print 'Regular file'
    if os.path.isdir(fpath):
        print 'Directory'
    if os.path.islink(fpath):
        print 'Symbolic link'
```

```
flist = os.listdir('.')
for fname in flist:
    filetype(fname)
```

- 경로명 : 디렉토리, 파일 자체
- os.path.isfile(파일) = true, (파일X)=false
- os.path.isdir(디렉토리)=true, (디렉토리X)=false
- os.path.islink(링크파일)=true
- windows os → 바로가기 파일이 링크, 리눅스, 맥 → 심볼릭 링크
- examples → 디렉토리
- 대부분의 경로는 파일
- tem → 디렉토리

■ 파일 다루기

2. 파일 종류 알아보기

```
.DS_Store : Regular file
.git : Directory
.gitignore : Regular file
.ipynb_checkpoints : Directory
example : Directory
files : Directory
images : Directory
module_test.py : Regular file
module_test.pyc : Regular file
music : Regular file
mymath.py : Regular file
mymath.pyc : Regular file
pickle.txt : Regular file
pickle2.txt : Regular file
python01.ipynb : Regular file
python02.ipynb : Regular file
python03.ipynb : Regular file
python04.ipynb : Regular file
python05.ipynb : Regular file
python06.ipynb : Regular file
python07.ipynb : Regular file
python08.ipynb : Regular file
python09.ipynb : Regular file
python10.ipynb : Regular file
python11.ipynb : Regular file
python12.ipynb : Regular file
python13.ipynb : Regular file
python14.ipynb : Regular file
python15.ipynb : Regular file
python16.ipynb : Regular file
python17.ipynb : Regular file
python18.ipynb : Regular file
python19.ipynb : Regular file
python20.ipynb : Regular file
python21.ipynb : Regular file
README.md : Regular file
removeme.txt : Regular file
sample.txt : Regular file
sample_new.txt : Regular file
t.txt : Regular file
t1.txt : Regular file
t2.txt : Regular file
```

■ 파일 다루기

3. 파일의 허가권

1) 파일의 허가권 알아보기

- `os.access(filepath, mode)`
 - mode에 들어갈 값
 - * `os.F_OK`: 파일 자체가 존재하는 것을 테스트
 - * `os.R_OK`: 읽기 권한이 있는 것을 테스트
 - * `os.W_OK`: 쓰기 권한이 있는 것을 테스트
 - * `os.X_OK`: 실행 권한이 있는 것(또는 디렉토리인지)을 테스트

```
import os
def fileaccess(fpath):
    print fpath, ':',
    if os.access(fpath, os.F_OK):
        print 'Exists',
    else:
        return
    if os.access(fpath, os.R_OK):
        print 'R',
    if os.access(fpath, os.W_OK):
        print 'W',
    if os.access(fpath, os.X_OK):
        print 'X',
    print

flist = os.listdir('.')
for fname in flist:
    fileaccess(fname)
```

■ 파일 다루기

3. 파일의 허가권

```
.DS_Store : Exists R W
.git : Exists R W X
.gitignore : Exists R W
.ipynb_checkpoints : Exists R W X
example : Exists R W X
files : Exists R W X
images : Exists R W X
module_test.py : Exists R W
module_test.pyc : Exists R W
music : Exists R W
mymath.py : Exists R W
mymath.pyc : Exists R W
pickle.txt : Exists R W
pickle2.txt : Exists R W
python01.ipynb : Exists R W
python02.ipynb : Exists R W
python03.ipynb : Exists R W
python04.ipynb : Exists R W
python05.ipynb : Exists R W
python06.ipynb : Exists R W
python07.ipynb : Exists R W
python08.ipynb : Exists R W
python09.ipynb : Exists R W
python10.ipynb : Exists R W
python11.ipynb : Exists R W
python12.ipynb : Exists R W
python13.ipynb : Exists R W
python14.ipynb : Exists R W
python15.ipynb : Exists R W
python16.ipynb : Exists R W
python17.ipynb : Exists R W
python18.ipynb : Exists R W
python19.ipynb : Exists R W
python20.ipynb : Exists R W
python21.ipynb : Exists R W
README.md : Exists R W
removeme.txt : Exists R W
sample.txt : Exists R W X
sample_new.txt : Exists R W
t.txt : Exists R W
t1.txt : Exists R W
t2.txt : Exists R W
```

■ 파일 다루기

3. 파일의 허가권

- `os.access(fpath,2)` → 첫번째 인자가 존재하는지 확인
- `os.access()` → 파일이 존재, 권한 둘 다 확인 가능
- `os.F_OK` : 파일 자체가 존재하는 것을 테스트
- `os.R_OK` : 읽기 권한이 있는 것을 테스트
- `os.W_OK` : 쓰기 권한이 있는 것을 테스트
- `os.X_OK` : 실행 권한이 있는 것을 테스트
- 일반적으로 소유자가 가지고 있는 권한을 테스트

2) 파일의 허가권 변경하기

- `os.chmod(filepath, mode)`

`os.chmod('sample.txt', 0777)` # 리눅스에서의 실행 예

- `chmod` → 리눅스나 유닉스에 있는 명령어 그대로
- `os.chmod()` : 파일의 허가권을 변경하는 명령어
- `r`은 4, `w`는 2, `x`는 1의 값을 가짐
- 첫 번째 7은 소유자의 권한을 조정 → $1+2+4=7$ → 권한을 다 줌
- 두 번째 7은 그룹 권한을 조정
- 마지막 7은 그룹 밖의 권한을 조정

▣ 파일 다루기

4. 파일 조작하기

1) 파일 이름 변경하기

- `os.rename(old_filepath, new_filepath)`

```
import os
os.rename('t.txt', 't1.txt') # t.txt를 t1.txt로 바꾼다
print os.access('t.txt', os.F_OK)
print os.access('t1.txt', os.F_OK)
```

False
True

- `os.rename('t.txt', 't1.txt')` → t.txt를 t1.txt로 변경
- `os.access('t.txt', os.F_OK)` → 파일의 존재성을 테스트

2) 파일 이동하기

- `os.rename(old_filepath, new_filepath)`

```
os.rename('t1.txt', 'example/t1.txt') # 현재 작업 디렉토리의 t1.txt를 example에
                                     t1.txt이름으로 옮긴다.
print os.access('example/t1.txt', os.F_OK)
```

True

- 파일 이동 → 폴더를 다른 폴더로 보내는 것
- 파일명A, 폴더명/파일명A → 파일명은 같으나 폴더명 안으로 이동
- `os.F_OK` : 파일이 존재하는지 테스트

▣ 파일 다루기

4. 파일 조작하기

3) 파일 복사하기

- shutil 모듈 활용
- shutil.copyfile(src_filepath, dest_filepath)

```
import shutil
shutil.copyfile('sample.txt', 'sample_new.txt')
print os.access('sample_new.txt', os.F_OK)
```

True

- 파일 복사하기는 os 모듈에서는 사용 X
- copyfile : 파일을 복사하는 것
- sample.txt 파일을 sample_new.txt 파일로 copy
- sample.txt 에 내용이 없으므로 복사된 파일에도 내용 X

▣ 파일 다루기

5. 파일 이름 다루기

1) 상대 경로를 절대 경로로 변환하기

- `os.path.abspath(상대경로)`
 - * 실제 파일 존재와는 무관하게 절대경로로 변경함

```
import os
print os.path.abspath('o.txt')
```

```
/Users/yhhan/git/python-e-learning/o.txt
```

- `abspath()` → 지정한 파일을 절대 경로로 돌려줌
- 맥, 리누스의 절대경로는 항상 슬래쉬(/)로 시작됨
- `o.txt`는 상대 경로
- 상대 경로: 기준이 되는 경로 존재 → 현재 디렉토리가 기준
- 파일의 존재유무 관계 없이 상대경로를 덧붙여 절대경로 만듦

■ 파일 다루기

5. 파일 이름 다루기

2) 주어진 경로의 파일이 존재하는지 확인

- `os.path.exists(filepath)`

```
f = '/Users/yhhan/git/python-e-learning/sample.txt'
print os.path.exists(f)
print os.path.exists('sample.txt')
print os.path.exists('asdf.txt')
```

```
True
True
False
```

- `os.path.exists()` : 주어진 경로에 파일이 존재하는지 확인

3) 현재/부모 디렉토리를 가리키는 이름 얻기

```
print os.getcwd() #현재 디렉토리
print os.pardir #부모 디렉토리
```

```
.
```

```
..
```

- `os.getcwd()` → `os`가 가지고 있는 현재 디렉토리의 파라미터(기호) 확인
- `os.pardir` → `os`가 가지고 있는 부모 디렉토리의 파라미터(기호) 확인

▣ 파일 다루기

5. 파일 이름 다루기

4) 디렉토리 분리 문자 얻기

```
print os.sep
```

```
/
```

- os.sep → separation의 약자로 디렉토리를 구분하는 기호 확인

▣ 파일 다루기

6. 경로명 분리하기

1) 경로와 파일명으로 분리

```
f = '/Users/yhhan/git/python-e-learning/t.txt'
```

```
print os.path.basename(f) # 파일명만 추출  
print os.path.dirname(f) # 디렉토리 경로 추출
```

```
t.txt  
/Users/yhhan/git/python-e-learning
```

- 파일 존재 유무와 상관 없음
- 파일 경로에서 파일명만 추출 → `os.basename(f)`
- `dirname(f)` → 경로명에 존재하는 디렉토리 경로

2) 경로명과 파일명을 한번에 분리

```
print os.path.split(f)
```

```
('/Users/yhhan/git/python-e-learning', 't.txt')
```

- `split(f)` → `basename`과 `dirname`을 한꺼번에 튜플로 확인

■ 파일 다루기

6. 경로명 분리하기

3) MS 윈도우즈에서 드라이브명과 파일 경로명을 분리

```
print os.path.splitdrive(f)
```

```
('', '/Users/yhhan/git/python-e-learning/t.txt')
```

- `splitdrive()` → ms 윈도우에서 드라이브명 확인

4) 확장자 분리

```
print os.path.splitext(f)
```

```
('/Users/yhhan/git/python-e-learning/t', '.txt')
```

- `splitext()` → 확장자 분리
- `ext`가 `extension`의 약자

파이썬 프로그래밍

파일과 디렉토리 다루기



한국기술교육대학교
온라인평생교육원

■ 디렉토리 다루기

1. 디렉토리에 관련된 일반 작업

1) 현재 작업 디렉토리 알아보기

```
import os  
print os.getcwd()
```

```
/Users/yhhan/git/python-e-learning
```

- cwd → current working directory의 약자
- os.getcwd() → 현재 작업하고 있는 디렉토리 불러오기

2) 작업 디렉토리 변경하기

```
os.chdir('/Users/yhhan/Public/')  
print os.getcwd()
```

```
/Users/yhhan/Public
```

- os.chdir() → 작업 디렉토리 변경

■ 디렉토리 다루기

1. 디렉토리에 관련된 일반 작업

3) 디렉토리 만들기

```
import os
```

```
os.mkdir('temp')      # 0755 기본 모드(rwxr-xr-x)로 만들어짐
```

```
os.mkdir('temp2', 0700) # 0700 모드(rwx-----)로 만들어짐
```

```
os.makedirs('temp/level1/level2') #0755 기본 모드, 중간에 필요한 디렉토리도 모두생성
```

```
print os.access('/Users/yhhan/Public/temp', os.F_OK)
```

```
print os.access('/Users/yhhan/Public/temp2', os.F_OK)
```

```
print os.access('/Users/yhhan/Public/temp/level1/level2', os.F_OK)
```

```
-----
OSError                                Traceback (most recent call last)
<ipython-input-13-acd03c396068> in <module>()
      1 import os
      2
----> 3 os.mkdir('temp')      # 0755 기본 모드(rwxr-xr-x)로 만들어짐
      4 os.mkdir('temp2', 0700) # 0700 모드(rwx-----)로 만들어짐
      5 os.makedirs('temp/level1/level2') #0755 기본 모드, 중간에 필요한
                                         디렉토리도 모두생성
```

```
OSError: [Errno 17] File exists: 'temp'
```

- `os.mkdir('temp')` → 현재 작업 디렉토리에 `temp` 디렉토리 생성
- `0700` → 권한 모드 설정
- 소유자에게는 `r`, `w`, `x` 모든 권한 부여
- 일반적인 기본 디렉토리 권한 → `755` 모드
- `7(rwx)`, `5(r이 4, x가 1)`, `5(r이 4, x가 1)`
- `makedirs()` → 재귀적으로 여러 개의 디렉토리 생성
- 이미 `temp` 디렉토리가 존재하기 때문에 `error` 발생

■ 디렉토리 다루기

1. 디렉토리에 관련된 일반 작업

4) 디렉토리 삭제

```
os.rmdir('temp2') #디렉토리에 내용이 없을 때 삭제가능
```

```
os.rmdir('temp') #디렉토리에 다른 파일이 있으면 삭제할 수 없음
```

```
-----  
OSError                                Traceback (most recent call last)  
<ipython-input-43-cb2ef2e59e2c> in <module>()  
----> 1 os.rmdir('temp') #디렉토리에 다른 파일이 있으면 삭제할 수 없음  
  
OSError: [Errno 66] Directory not empty: 'temp'
```

- rmdir() → 디렉토리에 내용이 없을 경우 삭제 가능
- 디렉토리에 내용이 있으면 삭제 불가능 (OS error 발생)

▣ 디렉토리 다루기

1. 디렉토리에 관련된 일반 작업

5) 다단계 디렉토리 삭제

- `os.removedirs(filepath)`
 - `filepath`에 지정된 디렉토리들 중 맨 오른쪽 디렉토리 부터 차례차례로 삭제한다.
 - 디렉토리에 다른 파일이 있으면 삭제하기 않고 중단

```
os.removedirs('temp/level1/level2')
```

- `os.removedirs()` → 다단계 디렉토리 삭제 (↔`makedirs`함수)
- `removedirs`에 적힌 디렉토리의 맨 오른쪽부터 단계별 삭제

6) 하위 디렉토리까지 모두 한번에 삭제

- `shutil.rmtree()`
 - 파일은 물론 하위 디렉토리까지 모두 한번에 삭제 가능
 - 조심해서 사용해야 함

```
import shutil  
shutil.rmtree('temp')
```

- `shutil.rmtree(디렉토리)` → 디렉토리 안 내용 삭제
- `tree` → 디렉토리 모양이 나무인데 나무 자체를 모두 삭제

▣ 디렉토리 다루기

1. 디렉토리에 관련된 일반 작업

7) 디렉토리 복사

- `shutil.copytree(src_filepath, dest_filepath)`
 - 하위 디렉토리와 파일등을 지니고 있는 디렉토리를 복사

```
os.mkdir('temp')  
os.mkdir('temp/temp2', 0700)  
shutil.copytree('temp', 'myweb_backup')
```

- `temp/temp2` → `temp` 밑에 `temp2` 생성
- `copytree(디렉토리, 디렉토리2)` → 디렉토리를 디렉토리2로 복사
- 하위 디렉토리가 복잡하게 존재해도 `copytree`는 모두 복사

▣ 디렉토리 다루기

2. 디렉토리(트리) 탐색하기

- `os.walk(filepath)`
 - `filepath` 부터 시작하여 재귀적으로 모든 하위 디렉토리까지 탐색을 하는 함수
 - 탐색시 발견하는 모든 파일에 대해서는 다음 튜플을 리턴함
 - * (`dirpath`, `dirnames`, `filenames`)
 - + `dirpath`: 탐색하고 있는 디렉토리 경로
 - + `dirnames`: `dirpath` 안에 존재하는 서브 디렉토리의 리스트
 - + `filenames`: `dirpath` 안에 존재하는 파일 리스트

- 아래 예는 현재 디렉토리부터 모든 하위 디렉토리 내에 존재하는 모든 `.pyc` 파일을 삭제하는 프로그램

```
import os
os.chdir('/Users/yhhan/git/python-e-learning')
print os.getcwd()
print
for path, subdirs, files in os.walk(os.getcwd()):
    for fname in files:
        if fname.endswith('.pyc'):
            fullpath = os.path.join(path, fname)
            print 'removing', fullpath
            os.remove(fullpath)
```

```
/Users/yhhan/git/python-e-learning
```

```
removing /Users/yhhan/git/python-e-learning/module_test.pyc
removing /Users/yhhan/git/python-e-learning/mymath.pyc
```

■ 디렉토리 다루기

2. 디렉토리(트리) 탐색하기

- removing → 삭제
- os.walk(디렉토리) → 디렉토리의 모든 하위 디렉토리를 탐색
- path → 방문하고 있는 디렉토리의 위치를 경로명으로 반환
- subdirs → 현재 방문하고 있는 디렉토리의 하위 디렉토리 목록
- files → 현재 방문하고 있는 디렉토리 안 파일 목록
- fname.endswith('.txt') → fname이 txt로 끝나는지 살펴봄
- path + frame = 절대 경로
- 현재 디렉토리에 있는 모든 txt로 끝나는 것 삭제
- os.walk로 다시 올라와서 하위 디렉토리도 동일 작업 진행