note.nkmk.me

Pythonでパス文字列からファイル名・フォルダ名・拡張子を取得、結合

Posted: 2018-04-16 / Modified: 2019-12-08 / Tags: Python, ファイル処理, 文字列処理

Pythonでパス文字列からファイル名・フォルダ名・拡張子を取得したり、文字列を結合してパスを生成したりするには、標準ライブラリの `os.path` モジュールを使う。

- 11.2. os.path — 共通のパス名操作 — Python 3.6.5 ドキュメント

ここでは以下の内容について説明する。

- OSによるパスの区切り文字の違い
- ファイル名を取得: `os.path.basename()`
  - 拡張子ありのファイル名
  - 拡張子なしのファイル名
- フォルダ名を取得: `os.path.dirname()`
- ファイル名とフォルダ名のペアを取得: `os.path.split()`
- パス文字列がフォルダを示す場合の注意点
- 拡張子を取得: `os.path.splitext()`
  - 拡張子を変更したパス文字列を作成
  - ドット（ピリオド）なしの拡張子を取得
  - `.tar.gz` のような場合
- ファイル名とフォルダ名を結合してパス文字列を作成: `os.path.join()`
  - 同じフォルダの別のファイルのパス文字列を作成
- 異なるOSのフォーマットを利用
- Windowsの場合の例
  - 区切り文字バックスラッシュとraw文字列
  - ファイル名、フォルダ名、拡張子の取得例

- ドライブ文字を取得・結合: `os.path.splitdrive()`

以下のパス文字列を例とする。

```
import os

filepath = './dir/subdir/filename.ext'
```

source: os_path_basename_dirname_split_splitext.py

実行環境はMac。Windowsの場合の例は最後に示す。

なお、ここで説明するのはパス文字列からファイル名・フォルダ名（ディレクトリ名）を取得（抽出）する方法。フォルダ内に存在するファイル名一覧・パス一覧などを取得したい場合は以下の記事を参照。

- **関連記事:** Pythonでファイル名・ディレクトリ名の一覧をリストで取得

また、Python3.4以降ではパスをオブジェクトとして操作できるpathlibモジュールを使って同様にファイル名やフォルダ名、拡張子などを抽出することもできる。慣れるとpathlibのほうが使いやすい。

- **関連記事:** Python, pathlibでファイル名・拡張子・親ディレクトリを取得

OSによるパスの区切り文字の違い

パスの区切り文字はOSによって異なる。

UNIX（Macを含む）ではスラッシュ `/`、Windowsではバックスラッシュ `\` が使われる。

Pythonが動作しているOSにおける区切り文字は `os.sep` または `os.path.sep` で取得、確認できる。どちらでも同じ。

```
print(os.sep)
# /

print(os.sep is os.path.sep)
# True
```

source: os_path_basename_dirname_split_splitext.py

ファイル名を取得: os.path.basename()

パス文字列からファイル名を取得するには `os.path.basename()` を使う。

拡張子ありのファイル名

`os.path.basename()` は拡張子を含むファイル名部分の文字列を返す。

```
filepath = './dir/subdir/filename.ext'
```

source: os_path_basename_dirname_split_splitext.py

```
basename = os.path.basename(filepath)
print(basename)
# filename.ext

print(type(basename))
# <class 'str'>
```

source: os_path_basename_dirname_split_splitext.py

拡張子なしのファイル名

拡張子を含まないファイル名を取得するには後述の `os.path.splitext()` と組み合わせる。

```
basename_without_ext = os.path.splitext(os.path.basename(filepath))[0]
print(basename_without_ext)
# filename
```

source: os_path_basename_dirname_split_splitext.py

なお、 `os.path.splitext()` は最後（一番右側）のドット `.` で分割する。最初の（一番左側）のドット `.` で分割したい場合は文字列 `str` の `split()` メソッドを使う。

- **関連記事:** Pythonで文字列を分割（区切り文字、改行、正規表現、文字数）

```
filepath_tar_gz = './dir/subdir/filename.tar.gz'

print(os.path.splitext(os.path.basename(filepath_tar_gz))[0])
# filename.tar

print(os.path.basename(filepath_tar_gz).split('.', 1)[0])
# filename
```

source: os_path_basename_dirname_split_splitext.py

フォルダ名（ディレクトリ名）を取得: os.path.dirname()

パス文字列からフォルダ名（ディレクトリ名）を取得するには `os.path.dirname()` を使う。フォルダ名部分の文字列が返される。

```
filepath = './dir/subdir/filename.ext'
```

source: os_path_basename_dirname_split_splitext.py

```
dirname = os.path.dirname(filepath)
print(dirname)
# ./dir/subdir

print(type(dirname))
# <class 'str'>
```

source: os_path_basename_dirname_split_splitext.py

ファイルの直上のフォルダ名のみを取得したい場合は`os.path.basename()`と組み合わせる。

```
subdirname = os.path.basename(os.path.dirname(filepath))
print(subdirname)
# subdir
```

source: os_path_basename_dirname_split_splitext.py

ファイル名とフォルダ名のペアを取得: os.path.split()

ファイル名とフォルダ名（ディレクトリ名）を両方取得するには`os.path.split()`を使う。

`os.path.basename()`で取得できるファイル名の文字列と`os.path.dirname()`で取得できるフォルダ名の文字列のタプルが返される。

```
filepath = './dir/subdir/filename.ext'
```

source: os_path_basename_dirname_split_splitext.py

```
base_dir_pair = os.path.split(filepath)
print(base_dir_pair)
# ('./dir/subdir', 'filename.ext')

print(type(base_dir_pair))
# <class 'tuple'>

print(os.path.split(filepath)[0] == os.path.dirname(filepath))
# True

print(os.path.split(filepath)[1] == os.path.basename(filepath))
# True
```

source: os_path_basename_dirname_split_splitext.py

タプルのアンパックを利用してそれぞれの変数に代入できる。

- **関連記事:** Pythonでタプルやリストをアンパック（複数の変数に展開して代入）

```
dirname, basename = os.path.split(filepath)
print(dirname)
# ./dir/subdir

print(basename)
# filename.ext
```

source: os_path_basename_dirname_split_splitext.py

再度連結するには後述の `os.path.join()` を使う。

パス文字列がフォルダ（ディレクトリ）を示す場合の注意点

パス文字列がフォルダを示す場合、末尾に区切り文字があるかないかで結果が異なるので注意。

末尾に区切り文字がない場合。

```
dirpath_without_sep = './dir/subdir'
print(os.path.split(dirpath_without_sep))
# ('./dir', 'subdir')

print(os.path.basename(dirpath_without_sep))
# subdir
```

source: os_path_basename_dirname_split_splitext.py

末尾に区切り文字がある場合。最下層のフォルダ名を取得するには、 `os.path.dirname()` と `os.path.basename()` を組み合わせる。

```
dirpath_with_sep = './dir/subdir/'
print(os.path.split(dirpath_with_sep))
# ('./dir/subdir', '')

print(os.path.basename(os.path.dirname(dirpath_with_sep)))
# subdir
```

source: os_path_basename_dirname_split_splitext.py

拡張子を取得: os.path.splitext()

拡張子を取得するには `os.path.splitext()` を使う。

拡張子とそれ以外に分割されてタプルとして返される。拡張子はドット `.` 込みの文字列。

```
filepath = './dir/subdir/filename.ext'
```

source: os_path_basename_dirname_split_splitext.py

```
root_ext_pair = os.path.splitext(filepath)
print(root_ext_pair)
# ('./dir/subdir/filename', '.ext')

print(type(root_ext_pair))
# <class 'tuple'>
```

source: os_path_basename_dirname_split_splitext.py

`+` 演算子で結合すると元のパス文字列に戻る。

- **関連記事:** Pythonで文字列を連結・結合（+演算子、joinなど）

```
root, ext = os.path.splitext(filepath)
print(root)
# ./dir/subdir/filename

print(ext)
# .ext

path = root + ext
print(path)
# ./dir/subdir/filename.ext
```

source: os_path_basename_dirname_split_splitext.py

拡張子を変更したパス文字列を作成

元のパス文字列から拡張子だけを変更したパス文字列を作成するには、`os.path.splitext()`で取得できるタプルの最初（0始まりの0番目）の要素と任意の拡張子の文字列を結合する。

```
other_ext_filepath = os.path.splitext(filepath)[0] + '.jpg'
print(other_ext_filepath)
# ./dir/subdir/filename.jpg
```

source: os_path_basename_dirname_split_splitext.py

ドット（ピリオド）なしの拡張子を取得

ドット`.`なしの拡張子を取得したい場合はスライス`[1:]`で2文字目以降を指定する。

- **関連記事:** Pythonのスライスによるリストや文字列の部分選択・代入

```
ext_without_dot = os.path.splitext(filepath)[1][1:]
print(ext_without_dot)
# ext
```

source: os_path_basename_dirname_split_splitext.py

.tar.gzのような場合

上述のファイル名取得の例で示したように、`os.path.splitext()`は最後（一番右側）のドット`.`で分割する。`.tar.gz`のような拡張子の場合は要注意。

```
filepath_tar_gz = './dir/subdir/filename.tar.gz'
```

source: os_path_basename_dirname_split_splitext.py

```
print(os.path.splitext(filepath_tar_gz))
# ('./dir/subdir/filename.tar', '.gz')
```

source: os_path_basename_dirname_split_splitext.py

ファイル名における最初（一番左側）のドット`.`で分割したい場合、文字列の`split()`メソッドを使うが、フォルダ部分にもドット`.`が含まれているとうまくいかない。

```
print(filepath_tar_gz.split('.', 1))
# ['', '/dir/subdir/filename.tar.gz']
```

source: os_path_basename_dirname_split_splitext.py

`os.path.split()`で分割したあと、文字列の`split()`メソッドを適用し、後述の`os.path.join()`で連結すればよい。

文字列の`split()`メソッドでは区切り文字が含まれないので、`os.path.splitext()`と同様に拡張子にドット`.`をつけたい場合は注意。

```
dirname, basename = os.path.split(filepath_tar_gz)
basename_without_ext, ext = basename.split('.', 1)
path_without_ext = os.path.join(dirname, basename_without_ext)
print(path_without_ext)
# ./dir/subdir/filename

print(ext)
# tar.gz

ext_with_dot = '.' + ext
```

```
print(ext_with_dot)
# .tar.gz
```

source: os_path_basename_dirname_split_splitext.py

ファイル名とフォルダ名を結合してパス文字列を作成: os.path.join()

ファイル名とフォルダ名を結合して新たなパス文字列を作成するには `os.path.join()` を使う。

引数に指定した文字列がパス区切り文字で区切られて結合される。複数の文字列を指定できる。

```
path = os.path.join('dir', 'subdir', 'filename.ext')
print(path)
# dir/subdir/filename.ext
```

source: os_path_basename_dirname_split_splitext.py

同じフォルダの別のファイルのパス文字列を作成

あるファイルと同一フォルダの別のファイルのパス文字列を作成したい場合は、 `os.path.dirname()` と `os.path.join()` を組み合わせる。

```
filepath = './dir/subdir/filename.ext'
```

source: os_path_basename_dirname_split_splitext.py

```
other_filepath = os.path.join(os.path.dirname(filepath), 'other_file.ext')
print(other_filepath)
# ./dir/subdir/other_file.ext
```

source: os_path_basename_dirname_split_splitext.py

異なるOSのフォーマットを利用

現在Pythonが動作しているOSではないOSのフォーマットでパス文字列を操作する場合は、`os`モジュールではなくそれぞれ別のモジュールをインポートして使う。

- UNIX（現行のMac含む）：`posixpath`
- Windows: `ntpath`
- Macintosh 9以前のMac: `macpath`

いずれのモジュールも`os.path`と同一のインターフェイスを持っているため、これまでのサンプルコードの`os.path`の部分をそれぞれのモジュール名（`ntpath`など）に変更すれば動作する。

Windowsの場合の例

Windowsの場合の例を示す。

ここでは上述の`ntpath`モジュールを使ってMac上で実行している。

Windowsで実行する場合は以下のサンプルコード中の`ntpath`を`os.path`に置き換えても同様の結果となる。

区切り文字バックスラッシュとraw文字列

Windowsのパスの区切り文字はバックスラッシュ`\`。

コード中の文字列でバックスラッシュを記述するにはエスケープする必要があるため、バックスラッシュを2文字連続して書く。`print()`ではバックスラッシュ1文字が出力される。

```
import ntpath

print(ntpath.sep)
# \

print('\\')
# \

print(ntpath.sep is '\\')
# True
```

source: os_ntpath.py

raw文字列（`r'xxx'`）を使うとエスケープが無効になるのでWindowsで深い階層のパスを記述するのが楽。raw文字列と通常の文字列は書き方が違うだけで値としては等価。どちらを使ってもよい。

```
file_path = 'c:\\dir\\subdir\\filename.ext'
file_path_raw = r'c:\dir\subdir\filename.ext'

print(file_path == file_path_raw)
# True
```

source: os_ntpath.py

raw文字列については以下の記事を参照。バックスラッシュ1文字をraw文字列で表すことはできないので注意。

- **関連記事:** Pythonでエスケープシーケンスを無視（無効化）するraw文字列

ファイル名、フォルダ名、拡張子の取得例

Windowsでも変わらず動作する。

```
print(ntpath.basename(file_path))
# filename.ext

print(ntpath.dirname(file_path))
# c:\dir\subdir

print(ntpath.split(file_path))
# ('c:\\dir\\subdir', 'filename.ext')
```

source: os_ntpath.py

ドライブ文字を取得・結合: os.path.splitdrive()

ドライブ文字を取得するには`os.path.splitdrive()`を使う。サンプルコードでは`ntpath.splitdrive()`になっている。

コロン `:` 込みのドライブ文字とそれ以外に分割しタプルを返す。

```
print(ntpath.splitdrive(file_path))
# ('c:', '\\dir\\subdir\\filename.ext')
```

source: os_ntpath.py

ドライブ文字のみ取得したい場合は1文字目を選択する。

```
drive_letter = ntpath.splitdrive(file_path)[0][0]

print(drive_letter)
# c
```

source: os_ntpath.py

ドライブ文字を結合するには注意が必要。

`os.path.join()` にそのまま渡すと上手くいかない。

```
print(ntpath.join('c:', 'dir', 'subdir', 'filename.ext'))
# c:dir\subdir\filename.ext
```

source: os_ntpath.py

区切り文字 `os.sep` （サンプルコードでは `ntpath.sep` ）も合わせて `os.path.join()` の引数として指定するか、ドライブ文字に区切り文字を加えてしまえばOK。

```
print(ntpath.join('c:', ntpath.sep, 'dir', 'subdir', 'filename.ext'))
# c:\dir\subdir\filename.ext

print(ntpath.join('c:\\', 'dir', 'subdir', 'filename.ext'))
# c:\dir\subdir\filename.ext
```

source: os_ntpath.py

関連カテゴリー

- Python
- ファイル処理
- 文字列処理

関連記事

- Pythonでファイル内の任意の文字列を含む行を抽出（grep的処理）
- Pythonでゼロ埋めなしの数字の文字列リストをソート
- Pythonで文字列を比較（完全一致、部分一致、大小関係など）
- Pythonで文字列生成（引用符、strコンストラクタ）
- Pythonで文字列を連結・結合（+演算子、joinなど）
- Pythonで改行を含む文字列の出力、連結、分割、削除、置換
- Pythonで文字列・数値をゼロ埋め（ゼロパディング）
- Pythonのprint関数で文字列、数値および変数の値を出力
- Pythonで文字列を折り返し・切り詰めして整形するtextwrap
- Pythonのf文字列（フォーマット済み文字列リテラル）の使い方
- Pythonでファイル・ディレクトリを削除するos.remove, shutil.rmtreeなど
- Pythonでカレントディレクトリを取得、変更（移動）

- Pythonで文字列のリスト（配列）と数値のリストを相互に変換
- Pythonで文字列を分割（区切り文字、改行、正規表現、文字数）
- PythonでUnicodeエスケープされた文字列・バイト列を変換
