

Walking Directories

The first approach I will cover is to use the `os.scandir` function to parse all the files and directories in a given path and build a list of all the directories and all the files.

```
folders = []
files = []

for entry in os.scandir(p):
    if entry.is_dir():
        folders.append(entry)
    elif entry.is_file():
        files.append(entry)

print("Folders - {}".format(folders))
print("Files - {}".format(files))
```

```
Folders - [<DirEntry 'Scorecard_Raw_Data'>]
Files - [<DirEntry 'HS_ARCHIVE9302017.xls'>]
```

The key items to remember with this approach is that it does not automatically walk through any subdirectories and the returned items are `DirEntry` objects. This means that you manually need to convert them to `Path` objects if you need that functionality.

If you need to parse through all the subdirectories, then you should use `os.walk`. Here is an example that shows all the directories and files within the `data_analysis` folder.

```
for dirName, subdirList, fileList in os.walk(p):
    print('Found directory: %s' % dirName)
    for fname in fileList:
        print('\t%s' % fname)
```

```
Found directory: /media/chris/KINGSTON/data_analysis
    HS_ARCHIVE9302017.xls
Found directory: /media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data
    MERGED1996_97_PP.csv
    MERGED1997_98_PP.csv
    MERGED1998_99_PP.csv
    <...>
    MERGED2013_14_PP.csv
    MERGED2014_15_PP.csv
    MERGED2015_16_PP.csv
Found directory: /media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/Crosswalks_20170
    CW2000.xlsx
    CW2001.xlsx
    CW2002.xlsx
    <...>
    CW2014.xlsx
    CW2015.xlsx
```

```
Found directory: /media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/Crosswalks_20170
CW2002_v3.xlsx
CW2003_v1.xlsx
CW2000_v1.xlsx
CW2001_v2.xlsx
```

This approach does indeed walk through all the subdirectories and files but once again returns a `str` instead of a Path object.

These two approaches allow a lot of manual control around how to access the individual directories and files. If you need a simpler approach, the path object includes some additional options for listing files and directories that are compact and useful.

The first approach is to use `glob` to list all the files in a directory:

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

```
HS_ARCHIVE9302017.xls
```

As you can see, this only prints out the file in the top level directory. If you want to recursively walk through all directories, use the following glob syntax:

```
for i in p.glob('**/*.'):
    print(i.name)
```

```
HS_ARCHIVE9302017.xls
MERGED1996_97_PP.csv
<...>
MERGED2014_15_PP.csv
MERGED2015_16_PP.csv
CW2000.xlsx
CW2001.xlsx
<...>
CW2015.xlsx
CW2002_v3.xlsx
<...>
CW2001_v2.xlsx
```

There is another option to use the `rglob` to automatically recurse through the subdirectories. Here is a shortcut to build a list of all of the csv files:

```
list(p.rglob('*.csv'))
```

```
[PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED1996_97_PP.csv')
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED1997_98_PP.csv')
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED1998_99_PP.csv')
<...>]
```

```
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED2014_15_PP.csv')
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED2015_16_PP.csv')
```

This syntax can also be used to exclude portions of a file. In this case, we can get everything except xlsx extensions:

```
list(p.rglob('*.(!xlsx)*'))
```

```
[PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED1996_97_PP.csv')
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED1997_98_PP.csv')
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED1998_99_PP.csv')
<...>
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED2014_15_PP.csv')
PosixPath('/media/chris/KINGSTON/data_analysis/Scorecard_Raw_Data/MERGED2015_16_PP.csv')]
```

There is one quick note I wanted to pass on related to using `glob`. The syntax may look like a regular expression but it is actually a much more limited subset. A couple of useful resources are [here](#) and [here](#).