# File Recursion

The provided solution is divided up in two components:

- `_find_files(...)` , which is a generator function
- `find_files(...)`

## Design Choices

The generator function `_find_files()` first uses `scandir()` to list all items in the path directory According to PEP 471, using `scandir()` instead of `listdir()` can improve the performance of the code. Another benefit is that the attribute `.path` provides the full path.

While iterating over it, the first condition checks if this entry is an file and its suffix. If the condition is true, it passes back the full file path:

```
for entry in os.scandir(path):
    if entry.is_file(...) and entry.name.endswith(suffix):
        yield entry.path
```

*Note that in this problem, we ignore symlinks to files and folders*

The next condition checks if this entry is a directory. If it is indeed a directory, the generator function recursively calls itself in a for loop. This works, because a generator is a special type of iterator:

```
elif entry.is_dir(follow_symlinks=False):
    for path in _find_files(suffix, entry.path):
        yield path
```

The parent function `find_files()` needs to return a list of file paths. As required, we can do this in an one-liner that turns the generator expression into a list:

```
return list(_find_files(suffix,path))
```

## Time & Space Complexity

The time complexity is $O(N)$ considering that `N` is the total count of all files and directories. It will go through every file and directory only once with the recursive calls. Same for space complexity, so it is $O(N)$ too.