Python OS (Most used methods)

os.listdir() :

❖ it returns the whole contents of a folder (folders + files)

Use case:

- we can use it to find a file in a specific folder

```python
def iterate_files(path):
    # Loop through all the items in the path
    for item in os.listdir(path):
        # Join the path and the item name
        item_path = os.path.join(path, item)
        # Check if the item is a file
        if os.path.isfile(item_path):
            # Do something with the file
            print(item_path)
```

os.walk() :

❖ it returns a tuple of (root, dirs, files) for each subdirectory
❖ iterate through files and directories recursively in a given directory

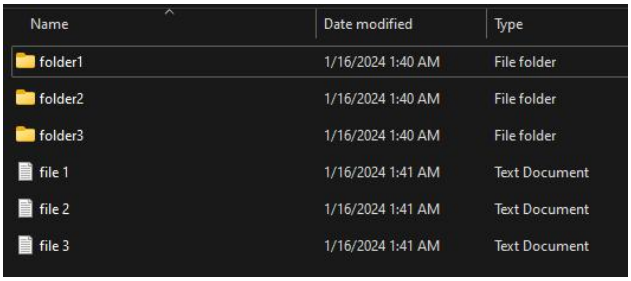
Use case:

- we can use it to find a file in a specific folder recursively

Example:

In this folder , the python script will

1 - iterate at the given folder first and find (sub-folders + files)
2 - will enter each sub-folders and walk inside it at each (sub-folders + files)
3 - It will return 3 tuples (root, dirs, files) as shown in the picture



Example :

Search for a file (in 2 ways) using :

1- os
2- pathlib

```python
from pathlib import Path
import os


def search_file(path, name):
    # Loop through all subdirectories in the path
    for root, dirs, files in os.walk(path):
        # Loop through all files in the current directory
        for file in files:
            # Check if the file name matches the name
            if file == name:
                # Return the absolute path of the file
                return os.path.join(root, file)
    # Return None if no file is found
    return None

def search_file2(path, name):

    files_pathes = [files for files in path.rglob(name)]
    for file_path in files_pathes:
                return file_path
    return None


# Get the parent directory of this script
script_directory = Path(__file__).parent
# create object path
anyother_directory = Path(r"D:\1_Embedded\GitHup atmega32 driver\Folder0")


# join pathes
log_file_path = script_directory / "logfile.txt"

# search for the file
myfile_path1 = search_file(script_directory,"logfile.txt")
myfile_path2 = search_file2(script_directory,"logfile.txt")

print(type(myfile_path1))
print(type(myfile_path2))

with open(myfile_path1, 'r') as file2:
    # Process the contents of file2
    contents2 = file2.read()
    print(contents2)

with open(myfile_path2, 'r') as file2:
    # Process the contents of file2
    contents2 = file2.read()
    print(contents2)


```

Find Current Python_File Location :

Using OS:

```python
import os


myFile = os.path.realpath(__file__)
myPath = os.path.dirname(os.path.realpath(__file__))

print(myFile) # out : D:\1_Embedded\My Repo's\Studying Python\main.py
print(myPath) # out : D:\1_Embedded\My Repo's\Studying Python
```

Using Pathlib:

```python
from pathlib import Path
script_path = Path(__file__).absolute()
script_directory = script_path.parent

print("Script path     :", script_path)       # Script path     : d:\1_Embedded\My Repo's\Studying Python\main.py
print("Script directory:", script_directory)  # Script directory: d:\1_Embedded\My Repo's\Studying Python
```

Difference between getcwd() and abspath() :

```python
# Get the current directory path of the script
current_directory = os.path.dirname(os.path.abspath(__file__))
print(current_directory) # d:\1_Embedded\My Repo's\Studying Python


# Get the current Working directory path
# Might not always be the same as the script's directory, especially if the script was executed from a different location.
current_directory = os.getcwd()
print(current_directory) # D:\1_Embedded\My Repo's
```

Python Pathlib(Most used methods)

Get Current Script Path :

```python
from pathlib import Path

script_name = Path(__file__).name
script_path = Path(__file__).absolute()         # Get Abslute Path of Current Script

script_directory_1 = script_path.parent         # Move one step  back from the script directory
script_directory_2 = Path(__file__).parent      # same as the previous line

parent_directory = script_path.parent.parent    # Move two steps back from the script directory

print("Script name       :", script_name)         # Script name     : main.py
print("Script path       :", script_path)         # Script path     : d:\1_Embedded\My Repo's\Studying Python\main.py
print("script_directory_1:", script_directory_1)  # Script directory: d:\1_Embedded\My Repo's\Studying Python
print("script_directory_2:", script_directory_2)  # Script directory: d:\1_Embedded\My Repo's\Studying Python
print("parent_directory  :", parent_directory)    # parent directory: d:\1_Embedded\My Repo's\
```

Make New Directory :

```python
""" Make New Directories inside new Directory """

new_directory = script_directory / 'python-file-paths' / 'foo' / 'bar' / 'baz.file'
new_directory.mkdir(parents=True, exist_ok=True)

# exist_ok=True : Fix error when trying to make a directory that already exists
# parents=True  : Fix error when trying to make a directory that doesn't exists like 'bar'
```

Searching For Files or directories:

```python
""" Searching for files Using rglob """

# Search for Specific file Extension
found_files = script_directory.rglob('*.py')

# The asterisk (*) is used to search for all files and folder in the current directory.
found_files = script_directory.rglob('*')
# Search with Partial Name
found_files = script_directory.rglob('**/*{file_name_partial}*')

# Search for Files in a Specific Subdirectory .. its goes to script_directory and find folder mypackages then start searching in it
specific_subdirectory_files = [file for file in script_directory.rglob('mypackages/*') if file.is_file()]

# Search with Multiple Criteria
specific_subdirectory_python_files = [file for file in script_directory.rglob('mypackages/*.py') if file.is_file()]
```

Searching For Files only or directories only in specific path :

```python
""" Looping on the files or directories after getting it """

found_files = script_directory.rglob('*')
MyFiles = [file for file in found_files if file.is_file()]        # Filter the results to include only files

found_files = script_directory.rglob('*')
MyFolders = [folder for folder in found_files if folder.is_dir()] # Filter the results to include only Folders
```

Search Recursively using rglob

```
Syntax :
    Path.rglob(pattern)
```

Use Folder.parts To Exclude specific folder and its content from the search

```
"""
    folder.parts : ('path', 'to', 'start', 'directory')
    is an attribute that returns a tuple representing the individual components of the path.
    Each element in the tuple corresponds to a part of the path, such as directories and the final file or directory.
"""
print(script_directory.parts) # ('d:\\', '1_Embedded', "My Repo's", 'Studying Python')

Example :
# find all folder and sub folder and exclude (.git) and its content
all_directories = [folder for folder in script_directory.rglob('*') if folder.is_dir() and folder.name != '.git' and not any(part == '.git' for part in folder.parts)]
```

Save File paths in 2 ways:

```
""" Search For Folder and Access it using 2 Ways """

# 1- Converting result to a List
found_path = list(script_directory.rglob('mypackages'))
print(found_path) # [WindowsPath("d:/1_Embedded/My Repo's/Studying Python/mypackages")]

# 2- Using list comprehension to Loop on files and return them in list
found_path = script_directory.rglob('mypackages')
MyFolders = [folder for folder in found_path]
print(MyFolders)  # [WindowsPath("d:/1_Embedded/My Repo's/Studying Python/mypackages")]
```

Note:

```
rglob("*.c") = glob("**/*.c"):
This is like calling Path.glob() with "**/" added in front of the given relative pattern
```

Remove Directory & its Content

Delete Empty Folder

```python
def delete_folder(folder_path):
    path = Path(folder_path)
    if path.exists() and path.is_dir():
        try:
            # Attempt to delete the folder if it's empty
            path.rmdir()
            print(f"Folder '{folder_path}' was successfully deleted.")
        except OSError:
            print(f"Folder '{folder_path}' is not empty. Use delete_folder_recursively if you want to remove it and all its contents.")
    else:
        print(f"Folder '{folder_path}' does not exist.")

# Example usage for deleting an empty folder
folder_path = "Test Folder"
delete_folder(folder_path)
```

Delete non Empty Folder

```python
def delete_folder_recursively(folder_path):
    path = Path(folder_path)
    if path.exists() and path.is_dir():
        try:
            # Use shutil.rmtree to delete the directory tree; this removes the directory and all its contents
            shutil.rmtree(path)
            print(f"Folder '{folder_path}' and all its contents were successfully deleted.")
        except Exception as e:
            print(f"An error occurred while deleting the folder and its contents: {e}")
    else:
        print(f"Folder '{folder_path}' does not exist.")

# Example usage for deleting a folder and all its contents
folder_path_recursive = "Test Folder"
delete_folder_recursively(folder_path_recursive)
```

Delete Folder Contents

```python
def delete_folder_contents(folder_path):
    path = Path(folder_path)
    if path.exists() and path.is_dir():
        for item in path.iterdir():
            if item.is_file() or item.is_symlink():
                item.unlink()  # Removes files and symbolic links
            elif item.is_dir():
                shutil.rmtree(item)  # Recursively removes directories with their contents
        print(f"All contents of '{folder_path}' have been deleted.")
    else:
        print(f"Folder '{folder_path}' does not exist or is not a directory.")

# Example usage
folder_path = "test"
delete_folder_contents(folder_path)
```