OS Module

OS is a predefined module in python. This module provides predefined functions to communicate with operating system. OS module functions are operating system dependent.

OS module provides the following functions.

1. Creating directory → mkdir()
2. Changing directory → chdir()
3. Finding current working directory → getcwd()
4. Removing directory → rmdir()
5. Renaming file
6. Listing files
7. Examine the properties of file

Example:

```
# write a python program to create folder

import os
fname=input("Enter Folder Name")
os.mkdir(fname)
print("Folder Created")
```

Output:

```
========= RESTART: F:/python6pmaug/ostest1.py =========
Enter Folder Namefolder1
Folder Created
```

Example:

```
# write a python program to change current working directory

import os

print(os.getcwd())
os.chdir("f:\\python6pmaug\\folder1")
print(os.getcwd())
f=open("file1","w")
print("file created")
```

Output:

```
========= RESTART: F:/python6pmaug/ostest2.py =========
F:\python6pmaug
f:\python6pmaug\folder1
file created
```

Example:

```
# write a program to remove directory or folder

import os

dname=input("Directory Name or Folder Name")
try:
    os.rmdir(dname)
    print("Directory is removed")
except OSError:
    print("Directory is not empty")
```

Output:

```
========= RESTART: F:/python6pmaug/ostest3.py =========
Directory Name or Folder Namefolder1
Directory is not empty
```

os.rename(src, dst)

Rename the file or directory src to dst. If dst exists, the operation will fail with an OSError

```
# write a program to rename file
import os
def main():
    old=input("Enter old filename")
    new=input("Enter new file name")
    os.rename(old,new)
    print("File Renamed...")

main()
```

Output:

```
========= RESTART: F:/python6pmaug/ostest4.py =========
Enter old filenamefile1.txt
Enter new file nametemp.txt
File Renamed...
```

os.path.isfile(*path*)
Return True if *path* is an existing regular file

os.path.isdir(*path*)
Return True if *path* is an existing directory

Example:

```
# write a program to find input filename is directory
# or regular file

import os.path
def main():
    fname=input("Enter filename")
    if os.path.isfile(fname):
        print("regular file")
    else:
        print("folder")

main()
```

Output:

```
========= RESTART: F:/python6pmaug/ostest5.py =========
Enter filenametemp.txt
regular file

========= RESTART: F:/python6pmaug/ostest5.py =========
Enter filenamepackage2
Folder
```

os.path.exists(*path*)
Return True if *path* refers to an existing path or an open file descriptor

Example:

```
# write a program to find input filename exists or not

import os.path
def main():
    fname=input("Enter filename")
    if os.path.exists(fname):
        print("File Found")
    else:
        print("File Not Found")

main()
```

Output:

```
========= RESTART: F:/python6pmaug/ostest6.py =========
Enter filenametemp.txt
File Found

========= RESTART: F:/python6pmaug/ostest6.py =========
Enter filenamefile1
File Not Found
```

os.listdir(*path='.'*)
Return a list containing the names of the entries in the directory given by *path*

Example:

```
# write a program list files exists in a given folder

import os
import os.path
def main():
    fname=input("Enter Folder Name")
    if os.path.exists(fname):
        if os.path.isdir(fname):
            l1=os.listdir(fname)
            print(l1)
        else:
            print("Not Folder")
    else:
        print("Folder Does Not Exists")
main()
```

Output:

```
========= RESTART: F:/python6pmaug/ostest7.py =========
Enter Folder Namefolder1
['file1']

========= RESTART: F:/python6pmaug/ostest7.py =========
Enter Folder Namefolder2
Folder Does Not Exists
```

Example:

```
# write a program to count number of files and folders
# in given path
import os
import os.path
def main():
    fname=input("Enter Folder Name")
    if os.path.exists(fname):
        if os.path.isdir(fname):
            l1=os.listdir(fname)
            os.chdir(fname)
            f,d=0,0
            for name in l1:
                if os.path.isfile(name):
                    f=f+1
                else:
                    d=d+1
            print(f'File Count {f}')
            print(f'Folder Count {d}')
        else:
            print("Not Folder")
    else:
        print("Folder Does Not Exists")
main()
```

Output:

```
========= RESTART: F:/python6pmaug/ostest7.py =========
Enter Folder Namefolder1
File Count 1
Folder Count 0

========= RESTART: F:/python6pmaug/ostest7.py =========
Enter Folder Namef:\\python6pmaug
File Count 392
Folder Count 4
```

os.remove(path)
Remove (delete) the file path. If path is a directory, an IsADirectoryError is raised. Use rmdir() to remove directories. If the file does not exist, a FileNotFoundError is raised.

Example:
```
# write a program to delete file

import os
import os.path
def main():
    fname=input("Enter FileName to Delete")
    if os.path.exists(fname):
        if os.path.isfile(fname):
            os.remove(fname)
            print("file deleted...")
        else:
            print("Not Regular File")
    else:
        print("File Not found")

main()
```

Output:
```
Enter FileName to Deletetemp.txt
File Not found
```

Regular Expressions (re module)

“re” module is default module which comes with python software.

What is regular expression?
Regular expression is a special string which defines search pattern.
Regular expression is used for searching pattern within string.

Regular expressions are used in application development,

1. Searching patterns
2. Match patterns
3. Input validations
4. Parsing (compilers/interpreters/parsers)
5. Chabot (ML → Machine Learning)
6. Text Editor
7. Search Engines

Python provides "re" module to work with regular expressions. "re" module provides the following functions to work with patterns.

1. match
2. search
3. findall
4. compile
5. sub

Q: How to create regular expression pattern?
Regular expression is created in two ways.

1. Defining string with prefix r
2. Using compile function of re module

Example:
r'@nareshit','ramesh@nareshit.com'
p=re.compile("@nareshit") → return pattern object, this can be used with one or more than one function

re.match(pattern, string, flags=0)
If zero or more characters at the **beginning of string** match the regular expression pattern, return a corresponding **match object.** Return None if the string does not match the pattern.

```
>>> import re
>>> m=re.match(r'py','python')
>>> print(m)
<re.Match object; span=(0, 2), match='py'>
>>> p=re.compile("py")
>>> print(p)
```

```
re.compile('py')
>>> p.match('python')
<re.Match object; span=(0, 2), match='py'>
>>> m=p.match('python')
>>> print(m)
<re.Match object; span=(0, 2), match='py'>
>> m.group()
'py'
>>> m=re.match(r'py','current python version 3.11')
>>> print(m)
None
```

re.search(pattern, string, flags=0)
Scan through string looking for the first location where the regular expression pattern produces a match, and return a corresponding match object. Return None if no position in the string matches the pattern.

```
>>> m=re.match(r'py','current python version 3.11')
>>> print(m)
None
>>> m=re.search(r'py','current python version 3.11')
>>> print(m)
<re.Match object; span=(8, 10), match='py'>
>>> m=re.search(r'py','current PYTHON version 3.11')
>>> print(m)
None
>>> m=re.search(r'py','current PYTHON version',re.IGNORECASE)
>>> print(m)
<re.Match object; span=(8, 10), match='PY'>
>>> m=re.search(r'py','current PYTHON version',re.I)
>>> print(m)
<re.Match object; span=(8, 10), match='PY'>
>>> m=re.search(r'py','python python python')
>>> print(m)
<re.Match object; span=(0, 2), match='py'>
```

re.findall(pattern, string, flags=0)
Return all non-overlapping matches of pattern in string, as a list of strings or tuples. The string is scanned left-to-right, and matches are returned in the order found. Empty matches are included in the result.

```
>>> m=re.findall(r'py','python python python ironpython')
>>> print(m)
['py', 'py', 'py', 'py']
```

Special characters used in creating patterns

.
(Dot.) In the default mode, this matches any character except a newline. If the DOTALL flag has been specified, this matches any character including a newline.

Example:
```
import re
def main():
    str1="python\nlanguage pypy ironpython jython"
    l=re.findall(r'.',str1)
    print(l)
    l=re.findall(r'.',str1,re.DOTALL)
    print(l)
    l=re.findall(r'..',str1)
    print(l)
    l=re.findall(r'p.',str1)
    print(l)
    l=re.findall(r'.y',str1)
    print(l)
main()
```

Output:
```
['p', 'y', 't', 'h', 'o', 'n', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e', ' ', 'p', 'y', 'p', 'y', ' ', 'i', 'r', 'o',
'n', 'p', 'y', 't', 'h', 'o', 'n', ' ', 'j', 'y', 't', 'h', 'o', 'n']
['p', 'y', 't', 'h', 'o', 'n', '\n', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e', ' ', 'p', 'y', 'p', 'y', ' ', 'i', 'r',
'o', 'n', 'p', 'y', 't', 'h', 'o', 'n', ' ', 'j', 'y', 't', 'h', 'o', 'n']
```

```
['py', 'th', 'on', 'la', 'ng', 'ua', 'ge', ' p', 'yp', 'y ', 'ir', 'on', 'py', 'th', 'on', ' j', 'yt',
'ho']
['py', 'py', 'py', 'py']
['py', 'py', 'py', 'py', 'jy']
```

^

(Caret.) Matches the start of the string, and in MULTILINE mode also matches immediately after each newline.

Example:

```
import re
def main():
    str1='''python is scripting
python is programming
python object oriented'''
    l=re.findall(r'^py',str1)
    print(l)
    l=re.findall(r'^py',str1,re.MULTILINE)
    print(l)
main()
```

Output:

```
['py']
['py', 'py', 'py']
```

$

Matches the end of the string or just before the newline at the end of the string, and in MULTILINE mode also matches before a newline.

Example:

```
import re
def main():
    str1='''python
jython
ironpython'''
    l=re.findall(r'on$',str1)
    print(l)
    l=re.findall(r'on$',str1,re.MULTILINE)
    print(l)
main()
```

Output:
```
========== RESTART: F:/python6pmaug/reex3.py ==========
['on']
['on', 'on', 'on']
```

*****
Causes the resulting RE to match 0 or more repetitions of the preceding RE, as many repetitions as are possible. ab* will match ‘a’, ‘ab’, or ‘a’ followed by any number of ‘b’s.

Example:
```
import re
def main():
    str1="ab abb a b abc ac ad ax"
    l=re.findall(r'ab*',str1)
    print(l)
    l=re.findall(r'a.*',str1)
    print(l)

main()
```

Output:
```
========== RESTART: F:/python6pmaug/regex4.py =========
['ab', 'abb', 'a', 'ab', 'a', 'a', 'a']
['ab abb a b abc ac ad ax']
```

+
Causes the resulting RE to match 1 or more repetitions of the preceding RE. ab+ will match ‘a’ followed by any non-zero number of ‘b’s; it will not match just ‘a’.