Assignment 3

Python Modules

os

- This module implements some useful functions on accessing the filesystem.
- All functions in this module raise **OSError** in the case of invalid or inaccessible file names and paths, or other arguments that have the correct type, but are not accepted by the operating system.

os

os.**getcwd**()

Return a string representing the current working directory

os.**listdir**(*path*)

Return a list containing the names of the entries in the directory given by *path*.

os.**mkdir**(*path*)

Create a directory named *path*

os.**rmdir**(*path*)

Remove (delete) the directory *path.* The directory should be empty.

os

os.**chmod**(*path, mode*)

Change the mode of *path* to the numeric *mode*.

os.**chown**(*path, uid, gid*)

Change the owner and group id of *path* to the numeric *uid* and *gid*. To leave one of the ids unchanged, set it to -1.

os

os.**open**(*file*, *flags*[, *mode*])

Open the file *file* and set various flags according to *flags* and possibly its mode according to *mode*. Return the file descriptor for the newly opened file. Default mode is 0777 (octal).

Some Open() flag constants:

os.**O_RDONLY**

os.**O_WRONLY**

os.**O_RDWR**

os.**O_CREAT**

os

os.**write**(*fd*, *str*)

Write the string *str* to file descriptor *fd*. Return the number of bytes actually written.

os.**access**(*path*, *mode*)

To check if the user is authorized for access to *path.* Return True if access is allowed, False if not.

os.**lstat**(*path*)

Perform the equivalent of an lstat() system call on the given *path*

filecmp

filecmp.**cmp**(*f1*, *f2*[, *shallow*])

Compare the files named *f1* and *f2*, returning **True** if they seem equal, **False** otherwise.

By default, cmp() compares without looking inside the files. The shallow argument tells cmp() whether to look at the contents of the file, as well. - filecmp.**cmp**(*f1*, *f2*, False)

shutil

shutil.**copyfile**(*src*, *dst*)

Copy the contents (no metadata) of the file named *src* to a file named *dst*. If *dst* already exists, it will be replaced. *src* and *dst* are path names given as strings.

shutil.**rmtree**(path)

Delete an entire directory tree.

glob

glob.glob(*pathname*)

For a literal match, wrap the meta-characters in brackets. For example, `'[?]'` matches the character `'?'`.

re

This module provides regular expression matching operations

The functions in this module could check if a particular string matches a given regular expression

Regular expression

specifies a set of strings that matches the pattern

e.g.

- ‘**+**’: indicates *one or more* occurrences of the preceding element.
- `ab+c` matches "abc", "abbc", "abbbc", and so on, but not "ac".

re

re.**compile**(pattern, flags=0)

Compile a regular expression pattern, returning a pattern object.

The expression's behaviour can be modified by specifying a ***flags*** value.

E.g. re.**I**: Ignore case.

combined using bitwise OR (the | operator).

re

re.**search**(pattern, string, flags=0)

- Scan through string looking for the first location where the regular expression pattern produces a match
- Return a corresponding match object
- Return None if no position in the string matches the pattern

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;

re

re.**match**(pattern, string, flags=0)

e.g.

prog = re.compile(pattern)

result = prog.match(string)

is equivalent to

result = re.match(pattern, string)

but using re.compile() and saving the resulting regular expression object for reuse is more efficient when the expression will be used several times in a single program.

os.path

os.path.**abspath**(*path*)

Return a normalized absolutized version of the pathname *path*.

```
>>> os.path.dirname("/usr/local/python3/bin/python")
'/usr/local/python3/bin/python'
```

os.path.**basename**(*path*)

Return the base name of pathname *path*.

```
>>> os.path.dirname("/usr/local/python3/bin/python")
'python'
```

os.path.**dirname**(*path*)

Return the directory name of pathname *path*.

```
>>> os.path.dirname("/usr/local/python3/bin/python")
'/usr/local/python3/bin/'
```

os.path

os.path.**join**(*path*, **paths*)

- Join one or more path components intelligently.
- The return value is the concatenation of *path* and any members of **paths* with exactly one directory separator (os.sep) following each non-empty part except the last, meaning that the result will only end in a separator if the last part is empty.

```
>>>                                          os.path.join("/tmp",file)
/tmp/file
```

- If a component is an absolute path, all previous components are thrown away and joining continues from the absolute path component.

os.path

os.path.**exists**(*path*)

- Return True if *path* refers to an existing path or an open file descriptor.
- Returns False for broken symbolic links.

os.path.**isdir**(*path*)

- Return True if *path* is an existing directory.
- This follows symbolic links, so both islink() and isdir() can be true for the same path.

os.path.**islink**(*path*)

Return True if *path* refers to an existing directory entry that is a symbolic link.