```python
>>> from os import mkdir
>>> mkdir('data')
```
```python
>>> from os import mkdir
>>> mkdir('data')
```
```python
>>> from os import mkdir
>>> mkdir('data')
>>> listdir(getcwd())
['data']
```
```python
>>> from os import mkdir
>>> mkdir('data')
>>> listdir(getcwd())
['data']
>>> listdir('data')
[]
```
```python
>>> from os import mkdir
>>> mkdir('data')
>>> listdir(getcwd())
['data']
>>> listdir('data')
[]
>>> mkdir('data')
```
>>> from os import mkdir
>>> mkdir('data')
>>> listdir(getcwd())
['data']
>>> listdir('data')
[]
>>> mkdir('data')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists: 'data'

Cannot mkdir on an existing directory
```python
>>> mkdir('country/regions/towns')
```
```python
>>> mkdir('country/regions/towns')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 2] No such file or directory: 'country/regions/towns'

mkdir cannot make nested directories
```
>>> mkdir('country/regions/towns')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 2] No such file or directory: 'country/regions/towns'

mkdir cannot make nested directories
but makedirs can

>>> from os import makedirs
>>> makedirs('country/regions/towns')
```python
>>> mkdir('country/regions/towns')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 2] No such file or directory: 'country/regions/towns'

mkdir cannot make nested directories but makedirs can

>>> from os import makedirs
>>> makedirs('country/regions/towns')
```

- **Python**
- **Manipulating Directories and Files**
>>> from os import rmdir

>>> rmdir('country/regions/towns')
```python
>>> from os import rmdir
>>> rmdir('country/regions/towns')
```
>>> from os import rmdir
>>> rmdir('country/regions/towns')
>>> rmdir('country')
```python
>>> from os import rmdir
>>> rmdir('country/regions/towns')
>>> rmdir('country')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists: 'country'
```

`rmdir` cannot remove nested directories
```python
>>> from os import rmdir
>>> rmdir('country/regions/towns')
>>> rmdir('country')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists: 'country'
```

`rmdir` cannot remove nested directories but `removedirs` can:

```python
>>> from os import removedirs
>>> removedirs('country')
```

```python
>>> from os import rmdir
>>> rmdir('country/regions/towns')
>>> rmdir('country')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists: 'country'
```

rmdir cannot remove nested directories but removedirs can

```python
>>> from os import removedirs
>>> removedirs('country')
```
```python
>>> from os import rmdir
>>> rmdir('country/regions/towns')
>>> rmdir('country')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists: 'country'
```

`rmdir` cannot remove nested directories, but `removedirs` can:

```python
>>> from os import removedirs
>>> removedirs('country')
```
>>> removedirs('country')
>>> removedirs('country')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  OSErrro: [Errno 17] File exists:
    'country'
removedirs cannot remove directories with files
>>> removedirs('country')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists: 'country'

removedirs cannot remove directories with files
but rmtree can

>>> from shutil import rmtree
>>> rmtree('country')
```python
>>> removedirs('country')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists: 'country'

removedirs cannot remove directories with files
but rmtree can

>>> from shutil import rmtree
>>> rmtree('country')
```
```python
>>> from os import remove

>>> remove('1.txt')

remove removes individual files
```
>>> from os import remove
>>> remove('1.txt')

remove removes individual files
```python
>>> from os import rename
>>> rename('A', 'B/D')
rename renames directories
```
```python
>>> from os import rename
>>> rename('A', 'B/D')
rename renames directories
```
>>> from os import rename
>>> rename('A', 'B/D')
rename renames directories
The directory must not exist
>>> rename('B/D', 'C')
```python
>>> from os import rename
>>> rename('A', 'B/D')
rename renames directories
The directory must not exist
>>> rename('B/D', 'C')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists
```
```python
>>> from os import rename
>>> rename('A', 'B/D')
rename renames directories
The directory must not exist
>>> rename('B/D', 'C')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists
>>> rename('B/D', 'C/D')
keep the same directory name, it must be provided explicitly
```
```python
>>> from os import rename
>>> rename('A', 'B/D')
rename renames directories
The directory must not exist
>>> rename('B/D', 'C')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists
>>> rename('B/D', 'C/D')
```

- The `rename` function renames directories.
- The directory must not exist.
- Attempting to rename a directory that already exists results in an `OSError`.
```python
>>> from os import rename
>>> rename('A/1.txt', 'B/3.txt')

rename also renames files
```
```python
>>> from os import rename
>>> rename('A/1.txt', 'B/3.txt')

rename also renames files
```
```python
>>> from os import rename
>>> rename('A/1.txt', 'B/3.txt')

rename also renames files

>>> rename('B/3.txt', 'C')

A destination file name needn't be given
```
>>> from os import rename

>>> rename('A/1.txt', 'B/3.txt')

rename also renames files

>>> rename('B/3.txt', 'C')

A destination file name needn’t be given as the source file name will be used
>>> from shutil import move
>>> move('A/1.txt', 'B/3.txt')

move is like rename but more powerful
>>> from shutil import move
>>> move('A/1.txt', 'B/3.txt')

This is like rename but more powerful. It preserves permission bits, group and owner
```python
>>> from shutil import move
>>> move('A/1.txt', 'B/3.txt')
```

`move` is like `rename` but more powerful. It preserves permission bits, group and owner access and modification times.
>>> from shutil import move
>>> move('A/1.txt', 'B/3.txt')

move is like rename but more powerful. It preserves permission bits, group and owner access and modification times for flags
```python
>>> from os import renames
>>> renames('A/1.txt', 'B/D/3.txt')

renames behaves like both
rename
makedirs
```
>>> from os import renames

>>> renames('A/1.txt', 'B/D/3.txt')

renames behaves like both
rename
makedirs

It creates any intermediate directories
```python
>>> from shutil import copytree
>>> copytree('A', 'B')
```

copytree copies directories and files
```python
>>> from shutil import copytree
>>> copytree('A', 'B')
```

`copytree` copies directories and files recursively. It preserves permission bits, group and owner, last access and modification times, and other flags.
```python
>>> from shutil import copyfile
>>> copyfile('A/1.txt', 'B/3.txt')

copyfile copies files
```
>>> from shutil import copyfile
>>> copyfile('A/1.txt', 'B/3.txt')

copyfile copies files
```python
>>> from shutil import copyfile
>>> copyfile('A/1.txt', 'B/3.txt')

copyfile copies files

>>> copyfile('B/3.txt', 'A')
```
>>> from shutil import copyfile
>>> copyfile('A/1.txt', 'B/3.txt')

copyfile copies files
destination file name must always be given

>>> copyfile('B/3.txt', 'A')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/usr/local/lib/python2.4/shutil.py", line 48, in copyfile
    fdst = open(dst, 'wb')
IOError: invalid mode: wb
```python
>>> from shutil import copy
>>> copy('A/1.txt', 'B')
copy also copies files
```
>>> from shutil import copy
>>> copy('A/1.txt', 'B')

copy also copies files
Unlike copyfile, no target file name needs to be given
```python
>>> from shutil import copy
>>> copy('A/1.txt', 'B')
```

Copy also copies files.

Unlike copyfile, no target file name needs to be given.

It also copies existing file permissions.
```python
>>> from shutil import copy
>>> copy('A/1.txt', 'B')
```

You can also copy files using `copy2`

```python
>>> from shutil import copy2
>>> copy2('A/1.txt', 'B')
```

`copy2` also copies files

Unlike `copyfile`, no target file name needs to be given.

It also copies existing file permissions.

`copy2` also copies files.
```python
>>> from shutil import copy
>>> copy('A/1.txt', 'B')
```

Copy also copies files.
Unlike copyfile, no target file name needs to be given.
It also copies existing file permissions.
```python
>>> from shutil import copy2
>>> copy2('A/1.txt', 'B')
```

Copy2 also copies files. It also copies permission bits, group and owner.
Last access and modification times.
Other flags.
Copytree uses copy2.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>os</code></td>
<td>Miscellaneous operating system interfaces</td>
</tr>
<tr>
<td><code>mkdir</code></td>
<td>Make a directory</td>
</tr>
<tr>
<td><code>makedirs</code></td>
<td>Make a directory and any intermediate directories</td>
</tr>
<tr>
<td><code>rmdir</code></td>
<td>Remove an empty directory</td>
</tr>
<tr>
<td><code>removedirs</code></td>
<td>Remove all empty directories in a path</td>
</tr>
<tr>
<td><code>remove</code></td>
<td>Remove a file</td>
</tr>
<tr>
<td><code>rename</code></td>
<td>Rename a file</td>
</tr>
<tr>
<td><code>renames</code></td>
<td>Rename a file, creating any intermediate directories</td>
</tr>
<tr>
<td><code>shutil</code></td>
<td>High-level file operations</td>
</tr>
<tr>
<td><code>rmtree</code></td>
<td>Remove a directory and all its contents</td>
</tr>
<tr>
<td><code>move</code></td>
<td>Move a file or a directory</td>
</tr>
<tr>
<td><code>copytree</code></td>
<td>Copy a directory and all its contents, using copy2.</td>
</tr>
<tr>
<td><code>copyfile</code></td>
<td>Copy a file’s contents</td>
</tr>
<tr>
<td><code>copy</code></td>
<td>Copy a file preserving the file permissions</td>
</tr>
<tr>
<td><code>copy2</code></td>
<td>Copy a file, preserving file permissions, group, owner, last access and modification times and flags</td>
</tr>
</tbody>
</table>