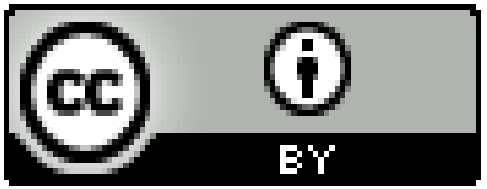
Python

Manipulating Directories and Files



```
>>> from os import mkdir
>>> mkdir('data')
```

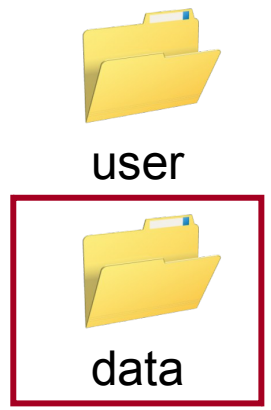
user

```
>>> from os import mkdir
>>> mkdir('data')
```

```
>>> from os import mkdir
>>> mkdir('data')
>>> listdir(getcwd())
['data']
```

```
>>> from os import mkdir
>>> mkdir('data')
>>> listdir(getcwd())
['data']
>>> listdir('data')
[]
```

```
>>> 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

```
>>> mkdir('country/regions/towns')
```

user

```
>>> 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'
```

user

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')
```

```
>>> 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')
```

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

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

user

country

regions

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

```
>>> 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

```
>>> 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

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

```
>>> 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

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

```
>>> 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‘
```

user

rmdir cannot remove nested directories

but removedirs can

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

```
>>> removedirs('country')
```

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

ovedirs cannot remove directories with files

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

ovedirs cannot remove directories with files

rmtree can

```
>>> from shutil import rmtree
>>> rmtree('country')
```

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

ovedirs cannot remove directories with files

rmtree can

```
>>> from shutil import rmtree
>>> rmtree('country')
```

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

remove removes individual files

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

remove removes individual files

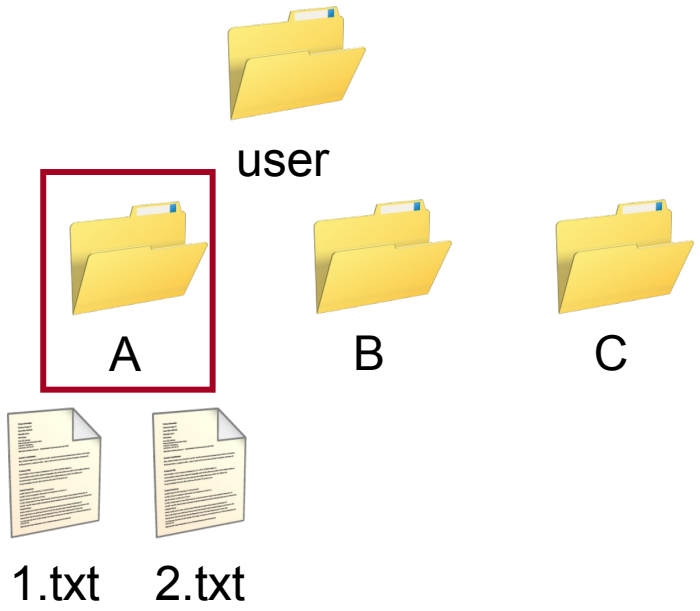
user

data

2.txt

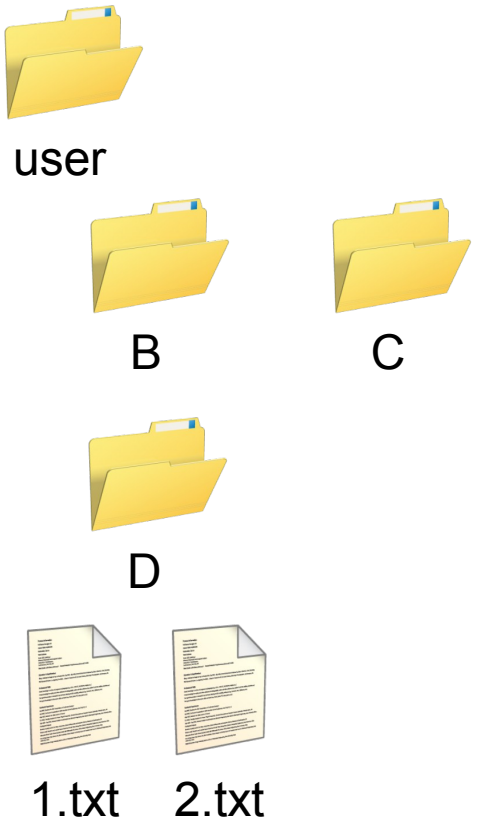
```
>>> from os import rename
>>> rename('A', 'B/D')
```

rename renames directories



```
>>> from os import rename
>>> rename('A', 'B/D')
```

rename renames directories

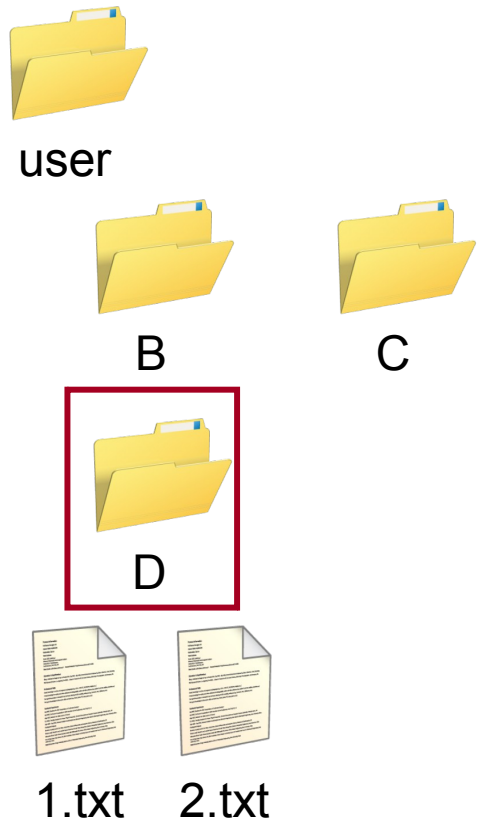


```
>>> from os import rename
>>> rename('A', 'B/D')
```

rename renames directories

The directory must not exist

```
>>> rename('B/D', 'C')
```

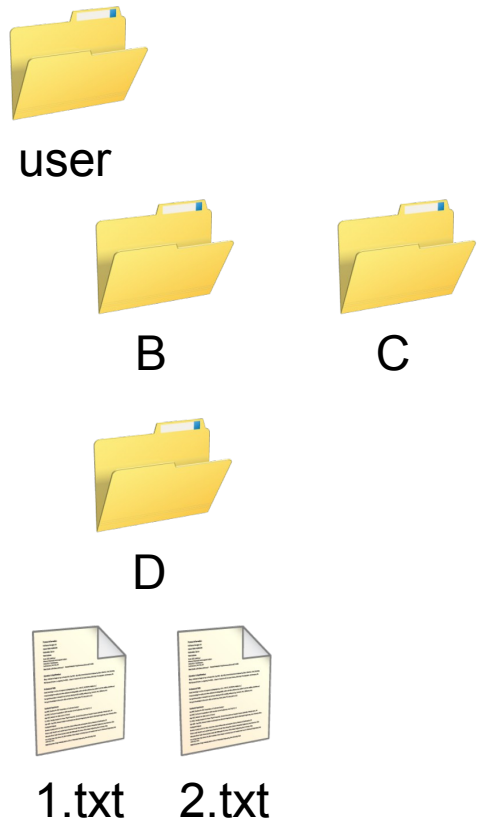


```
>>> from os import rename
>>> rename('A', 'B/D')
```

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
```



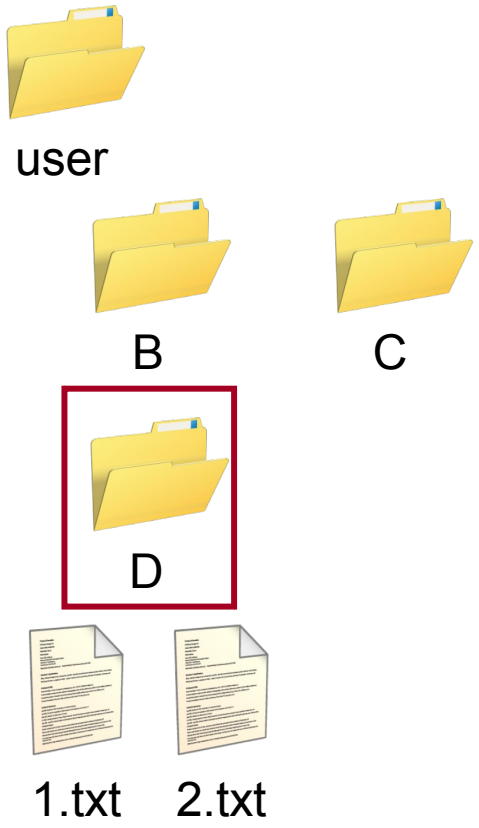
```
>>> from os import rename
>>> rename('A', 'B/D')
```

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 ovided explicitly

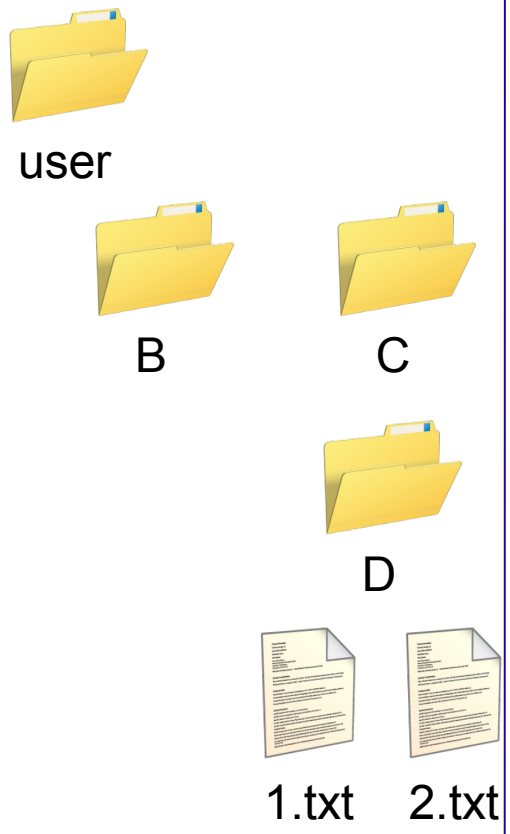


```
>>> from os import rename
>>> rename('A', 'B/D')
```

rename renames directories

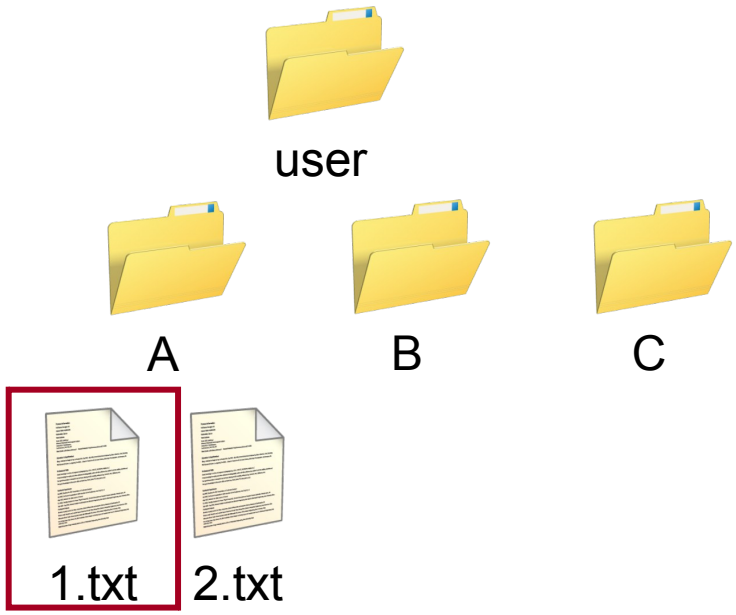
The directory must not exist

```
>>> rename('B/D', 'C')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
OSError: [Errno 17] File exists
>>> rename('B/D', 'C/D')
```



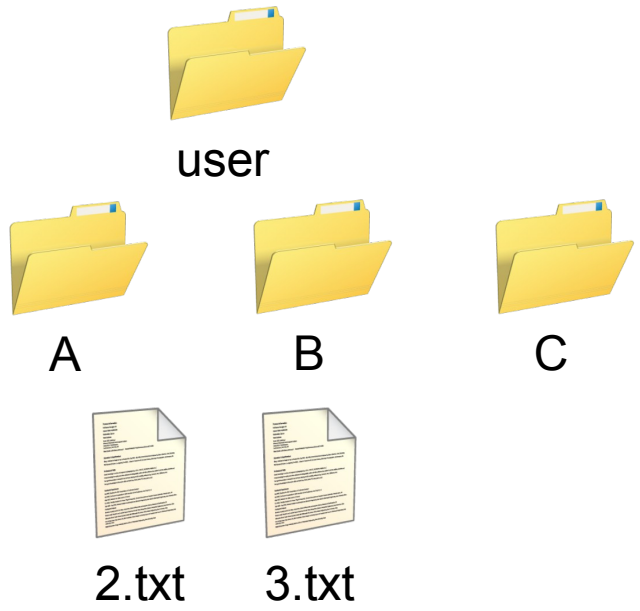
```
>>> from os import rename
>>> rename('A/1.txt', 'B/3.txt')
```

rename also renames files



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

rename also renames files

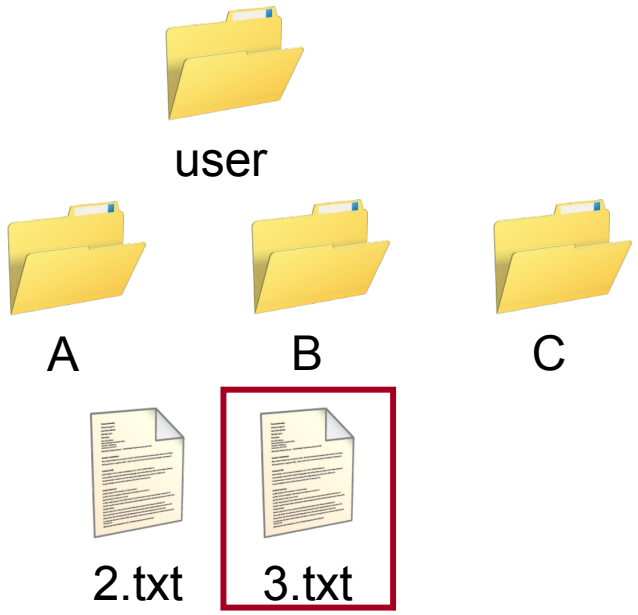


```
>>> from os import rename
>>> rename('A/1.txt', 'B/3.txt')
```

rename also renames files

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

A destination file name needn't be given

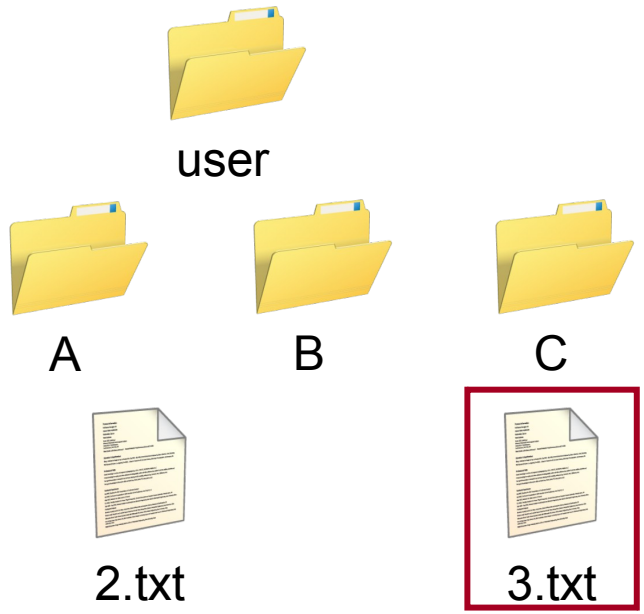


```
>>> from os import rename
>>> rename('A/1.txt', 'B/3.txt')
```

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')
```

is like rename but more powerful. It preserves
ission bits, group and owner

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

is like rename but more powerful. It preserves
ission bits, group and owner
access and modification times

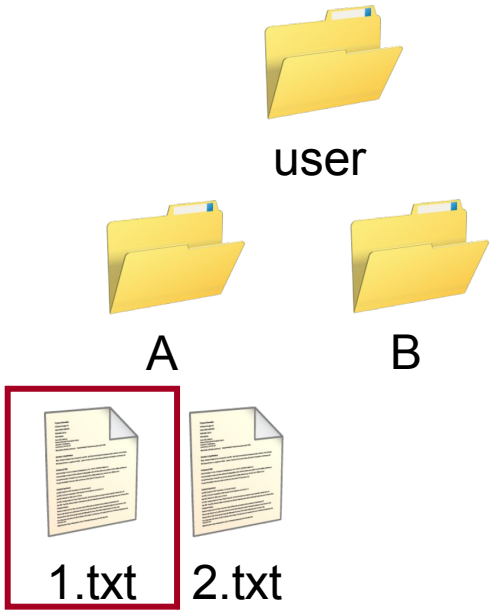
```
>>> from shutil import move
>>> move('A/1.txt', 'B/3.txt')
```

is like rename but more powerful. It preserves
ission bits, group and owner
access and modification times
r flags

```
>>> from os import renames
>>> renames('A/1.txt', 'B/D/3.txt')
```

renames behaves like both

- rename
- makedirs

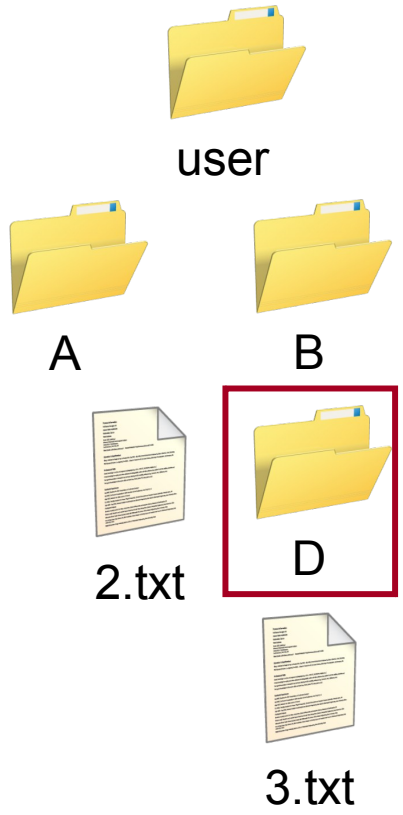


```
>>> from os import renames
>>> renames('A/1.txt', 'B/D/3.txt')
```

renames behaves like both

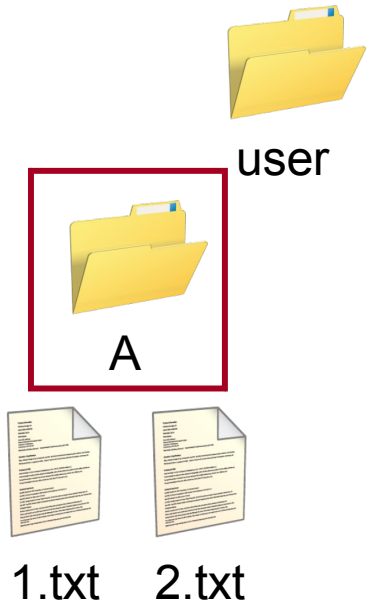
- rename
- makedirs

It creates any intermediate directories



```
>>> from shutil import copytree
>>> copytree('A', 'B')
```

copytree copies directories and files



```
>>> from shutil import copytree
>>> copytree('A', 'B')
```

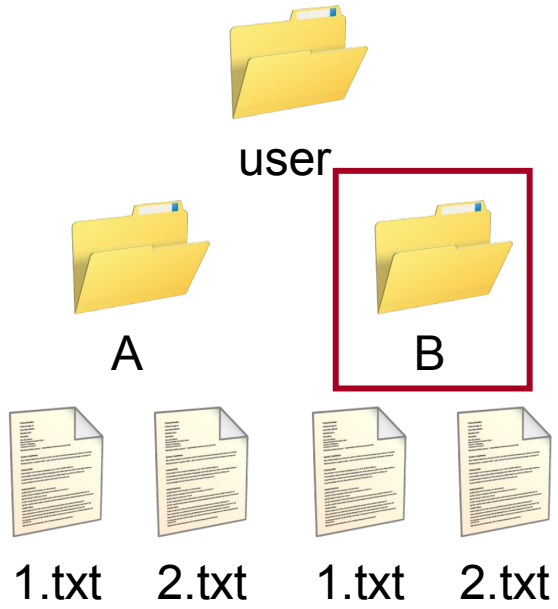
ytree copies directories and files recursively

preserves

ermission bits, group and owner

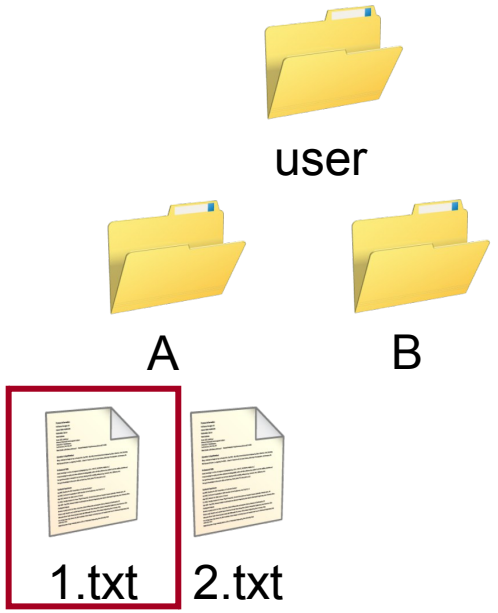
ast access and modification times

ther flags



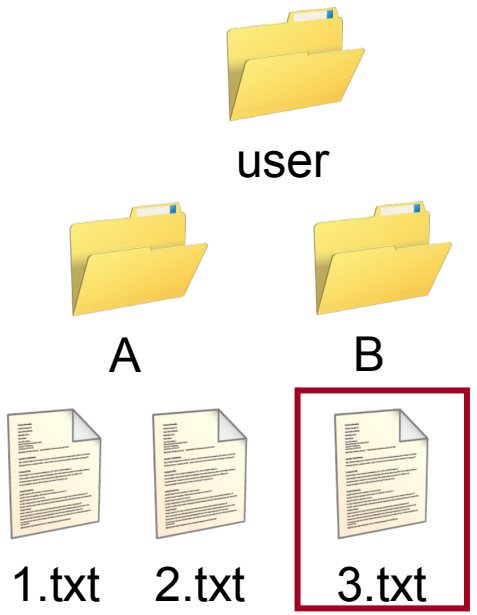
```
>>> from shutil import copyfile
>>> copyfile('A/1.txt', 'B/3.txt')
```

copyfile copies files



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

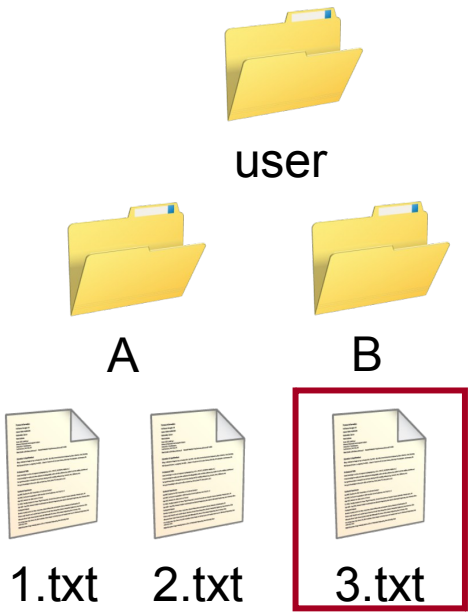
copyfile copies files



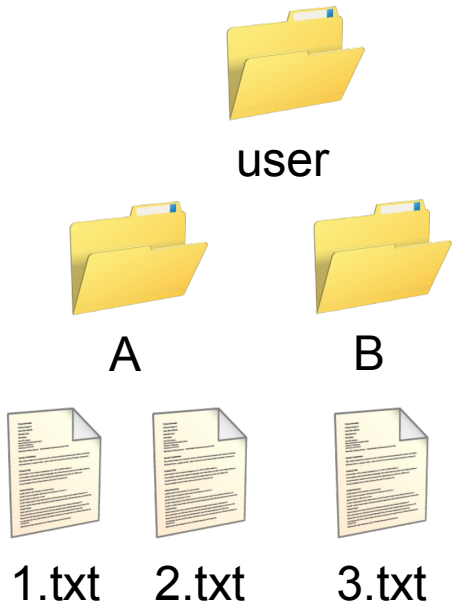
```
>>> from shutil import copyfile
>>> copyfile('A/1.txt', 'B/3.txt')
```

copyfile copies files

```
>>> copyfile('B/3.txt', 'A')
```



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



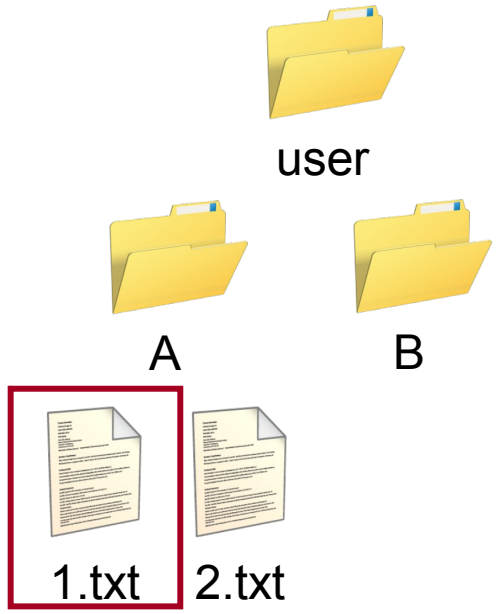
pyfile copies files

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
```

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

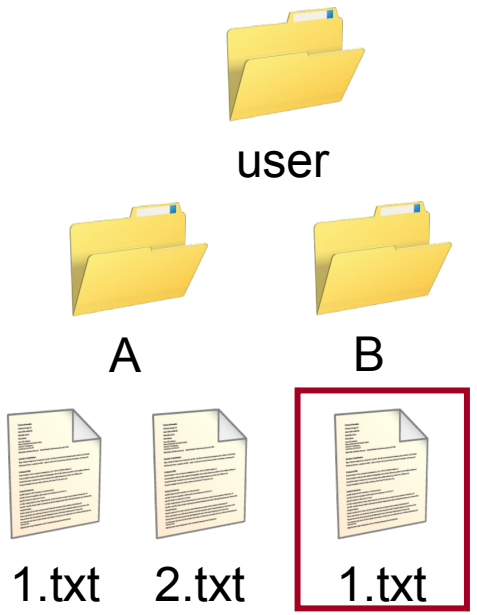
copy also copies files



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

py also copies files

nlike copyfile, no target file name needs to
e given

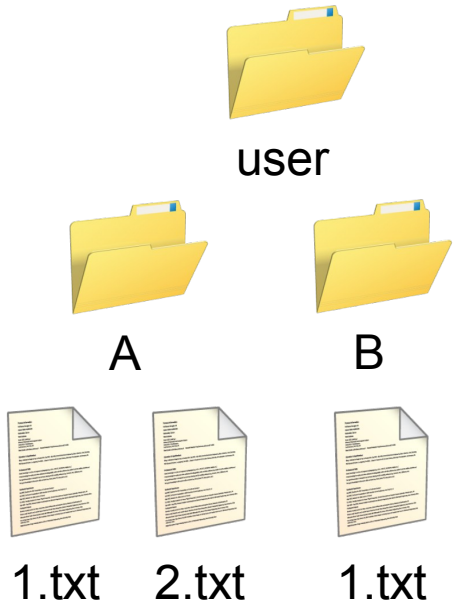


```
>>> from shutil import copy
>>> copy('A/1.txt', 'B')
```

py also copies files

nlike copyfile, no target file name needs to
e given

also copies existing file permissions



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

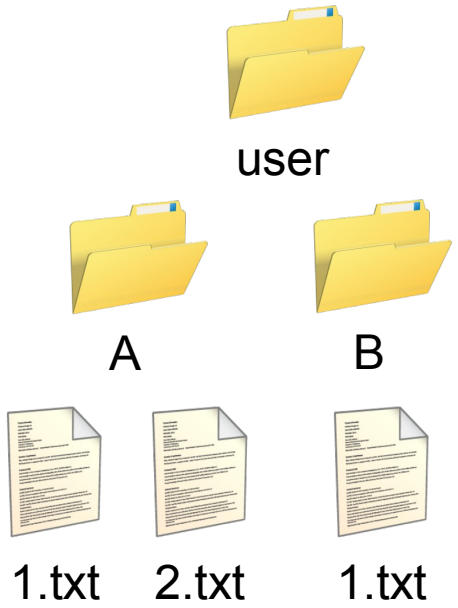
py also copies files

nlike copyfile, no target file name needs to
e given

also copies existing file permissions

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

copy2 also copies files



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

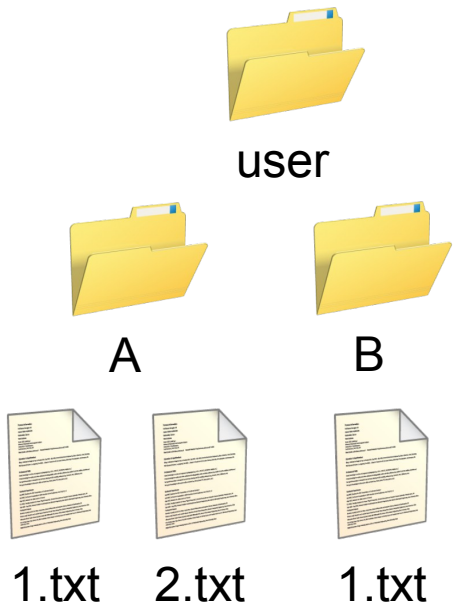
py also copies files

nlike copyfile, no target file name needs to e given

also copies existing file permissions

```
>>> 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



os	Miscellaneous operating system interfaces
mkdir	Make a directory
makedirs	Make a directory and any intermediate directories
rmdir	Remove an empty directory
removedirs	Remove all empty directories in a path
remove	Remove a file
rename	Rename a file
renames	Rename a file, creating any intermediate directories
shutil	High-level file operations
rmtree	Remove a directory and all its contents
move	Move a file or a directory
copytree	Copy a directory and all its contents, using copy2.
copyfile	Copy a file's contents
copy	Copy a file preserving the file permissions
copy2	Copy a file, preserving file permissions, group, owner, last access and modification times and flags

created by

Mike Jackson

May 2011