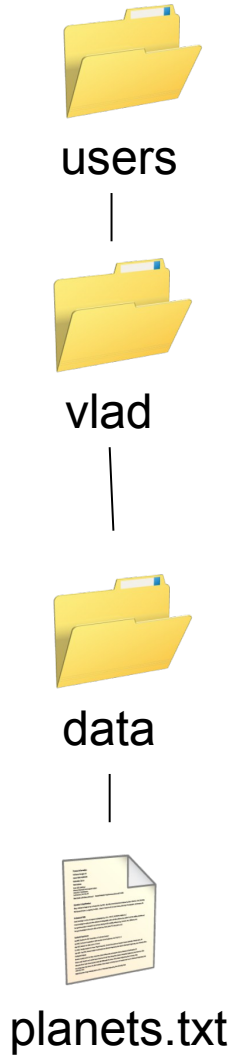
Python

Directory and File Paths

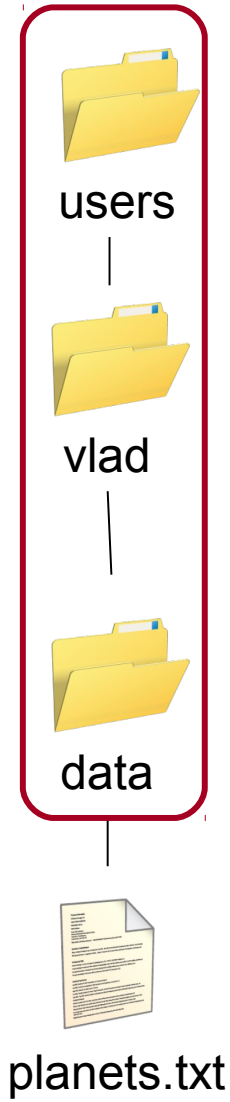
```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
```

We want to build a path



```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = base + '/' + user + '/' + datadir
>>> print path
/users/vlad/data
```

Use string addition



```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = base + '/' + user + '/' + datadir
>>> print path
/users/vlad/data
```

Code assumes Linux/UNIX paths

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = base + '/' + user + '/' + datadir
>>> print path
/users/vlad/data

>>> from os.path import join
```

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = base + '/' + user + '/' + datadir
>>> print path
/users/vlad/data

>>> from os.path import join
>>> path = join(base, user, datadir)
```

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = base + '/' + user + '/' + datadir
>>> print path
/users/vlad/data

>>> from os.path import join
>>> path = join(base, user, datadir)
>>> print path
/users/vlad/data
```

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = base + '/' + user + '/' + datadir
>>> print path
/users/vlad/data

>>> from os.path import join
>>> path = join(base, user, datadir)
>>> print path
/users/vlad/data
```

join picks the file separator based on the current operating system

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = join(base, user, datadir)
>>> print path
/users\\vlad\\data
```

Running under Windows

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = join(base, user, datadir)
>>> print path
/users\\vlad\\data
```

Running under Windows

join chooses the Windows separator

The double \ is only because we're printing them

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = join(base, user, datadir)
>>> print path
/users\\vlad\\data
```

Running under Windows

join chooses the Windows separator

The double \ is only because we're printing them

But it starts with a Linux/UNIX file separator

How do we convert that?

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = join(base, user, datadir)
>>> print path
/users\\vlad\\data

>>> from os.path import normpath
```

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = join(base, user, datadir)
>>> print path
/users\\vlad\\data

>>> from os.path import normpath
>>> nupath = normpath(path)
>>> print nupath
\\users\\vlad\\data
```

```
>>> base = '/users'
>>> user = 'vlad'
>>> datadir = 'data'
>>> path = join(base, user, datadir)
>>> print path
/users\\vlad\\data

>>> from os.path import normpath
>>> nupath = normpath(path)
>>> print nupath
\\users\\vlad\\data
>>> normpath('/some/other/path')
\\some\\other\\path
```

normpath converts path separators

```
>>> path = '/users/vlad//.///data/../..//vlad/./data/..'
```

ormpath not only converts path separators

```
>>> path = '/users/vlad//.///data/../..//vlad/./data/..'
>>> cleanpath = normpath(path)
>>> print cleanpath
/users/vlad
```

ormpath not only converts path separators

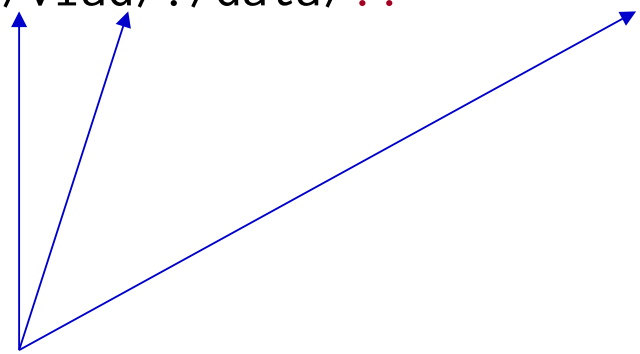
```
>>> path = '/users/vlad//.///data/../..//vlad/./data/..'
>>> cleanpath = normpath(path)
>>> print cleanpath
/users/vlad
```

rmpath not only converts path separators, it
emoves duplicated path separators

```
>>> path = '/users/vlad//.///data/../..//vlad/./data/..'
>>> cleanpath = normpath(path)
>>> print cleanpath
/users/vlad
```

rmpath not only converts path separators, it
emoves duplicated path separators
nd “.” current directory short-hand

```
>>> path = '/users/vlad//.///data/../..//vlad/./data/..'
>>> cleanpath = normpath(path)
>>> print cleanpath
/users/vlad
```



rmpath not only converts path separators, it

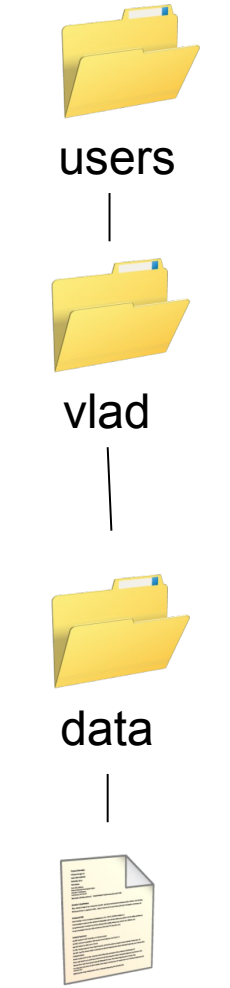
emoves duplicated path separators

emoves "." current directory short-hand

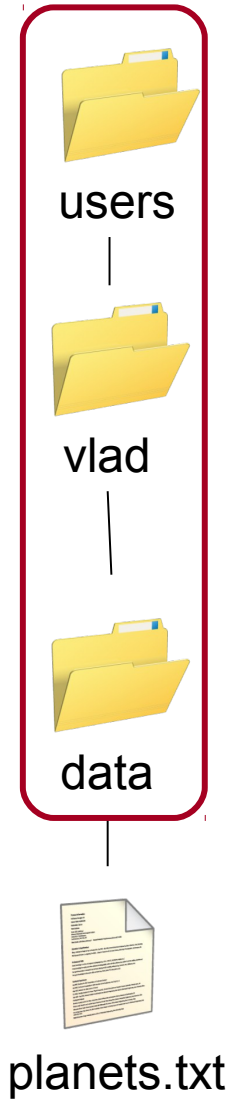
ies to resolve ".." parent directory short-hand

```
>>> from os.path import dirname, basename
```

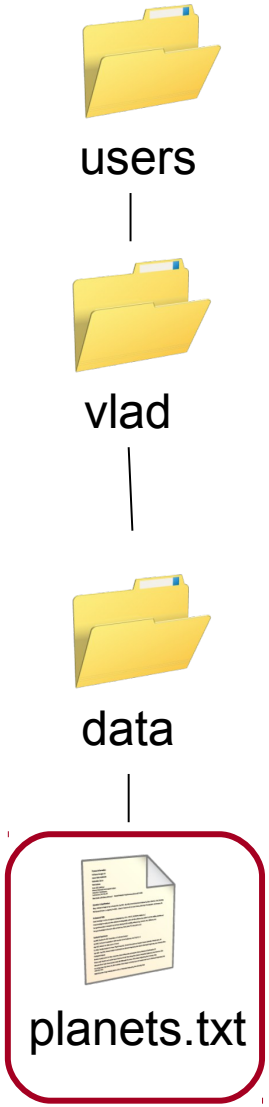
```
>>> from os.path import dirname, basename
>>> path = '/users/vlad/data/planets.txt'
```



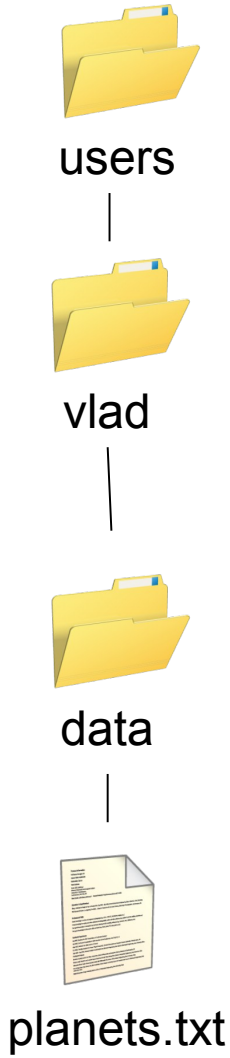
```
>>> from os.path import dirname, basename
>>> path = '/users/vlad/data/planets.txt'
>>> dirname(path)
/users/vlad/data
```



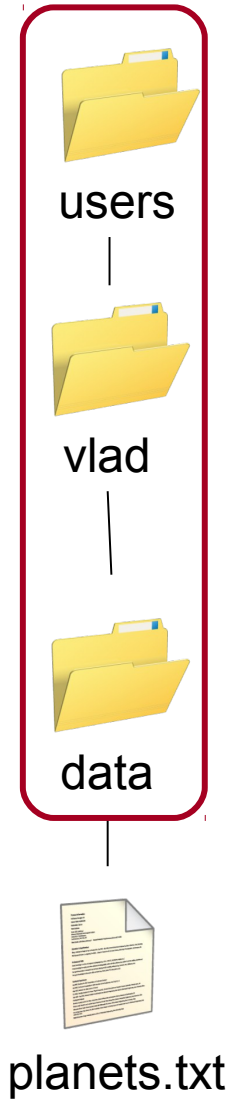
```
>>> from os.path import dirname, basename
>>> path = '/users/vlad/data/planets.txt'
>>> dirname(path)
/users/vlad/data
>>> basename(path)
planets.txt
```



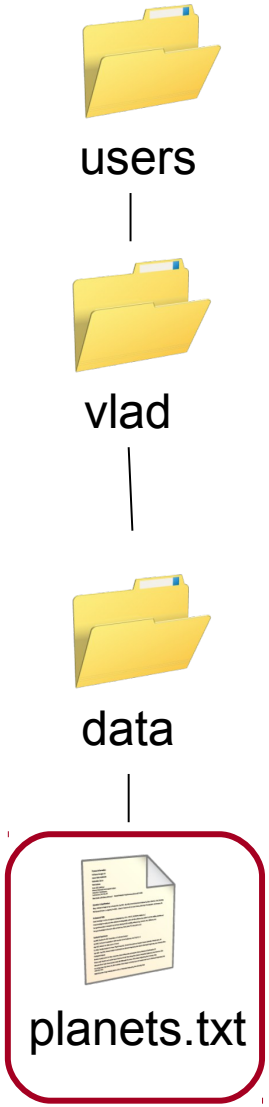
```
>>> from os.path import dirname, basename
>>> path = '/users/vlad/data/planets.txt'
>>> dirname(path)
/users/vlad/data
>>> basename(path)
planets.txt
>>> from os.path import split
>>> (head, tail) = split(path)
```



```
>>> from os.path import dirname, basename
>>> path = '/users/vlad/data/planets.txt'
>>> dirname(path)
/users/vlad/data
>>> basename(path)
planets.txt
>>> from os.path import split
>>> (head, tail) = split(path)
>>> print head
/users/vlad/data
```



```
>>> from os.path import dirname, basename
>>> path = '/users/vlad/data/planets.txt'
>>> dirname(path)
/users/vlad/data
>>> basename(path)
planets.txt
>>> from os.path import split
>>> (head, tail) = split(path)
>>> print head
/users/vlad/data
>>> print tail
planets.txt
```



```
>>> from os.path import splitext
>>> path = '/users/vlad/data/planets.txt'
```

```
>>> from os.path import splitext
>>> path = '/users/vlad/data/planets.txt'
>>> (root, ext) = splitext(path)
```

```
>>> from os.path import splitext
>>> path = '/users/vlad/data/planets.txt'
>>> (root, ext) = splitext(path)
>>> print root
/users/vlad/data/planets
```

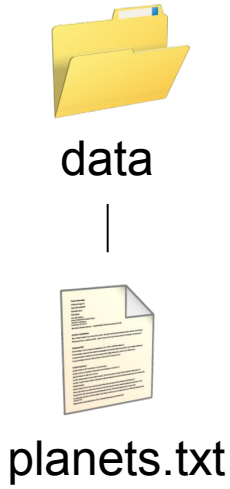
```
>>> from os.path import splitext
>>> path = '/users/vlad/data/planets.txt'
>>> (root, ext) = splitext(path)
>>> print root
/users/vlad/data/planets
>>> print ext
.txt
```

```
>>> from os.path import splitdrive
>>> path = 'C:\\users\\vlad\\data\\planets.txt'
>>> (drive, tail) = splitdrive(path)
```

```
>>> from os.path import splitdrive
>>> path = 'C:\\users\\vlad\\data\\planets.txt'
>>> (drive, tail) = splitdrive(path)
>>> print drive
C:
```

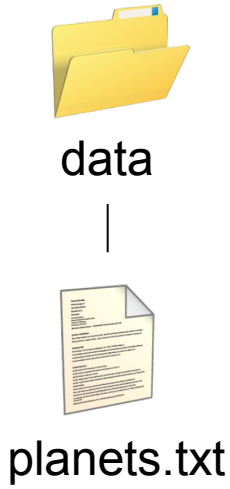
```
>>> from os.path import splitdrive
>>> path = 'C:\\users\\vlad\\data\\planets.txt'
>>> (drive, tail) = splitdrive(path)
>>> print drive
C:
>>> print tail
\\users\\vlad\\data\\planets.txt
```

```
>>> path = 'data/planets.txt'
```



```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
```

isabs checks is the path is absolute



```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
```

is the path is absolute

: does it start with / ?

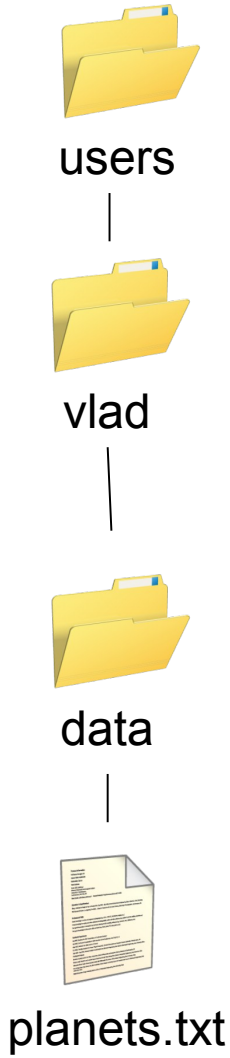
oes it start with \ after the drive's been removed?

data

planets.txt

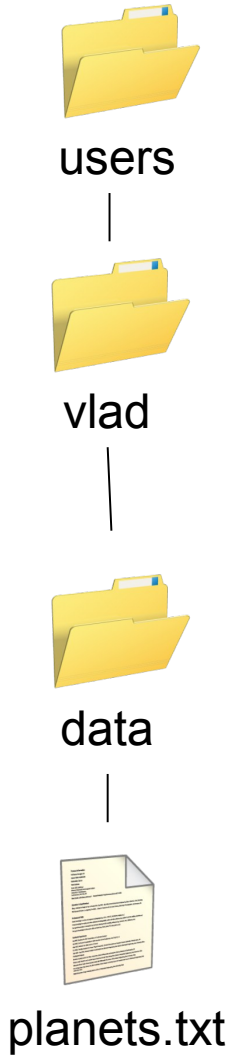
```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
>>> from os.path import abspath
>>> nupath = abspath(path)
```

abspath makes relative paths absolute



```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
>>> from os.path import abspath
>>> nupath = abspath(path)
>>> print abspath
/users/vlad/data/planets.txt
```

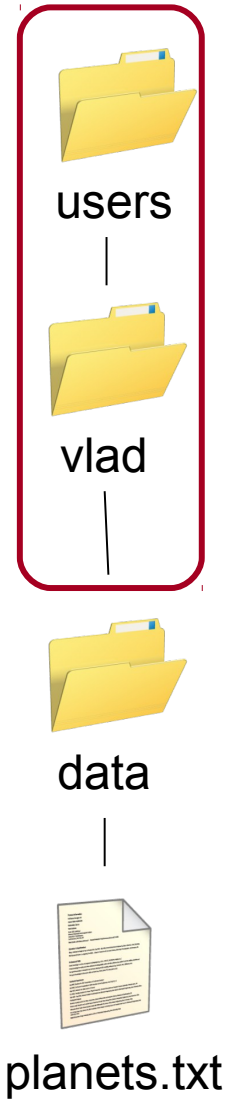
abspath makes relative paths absolute



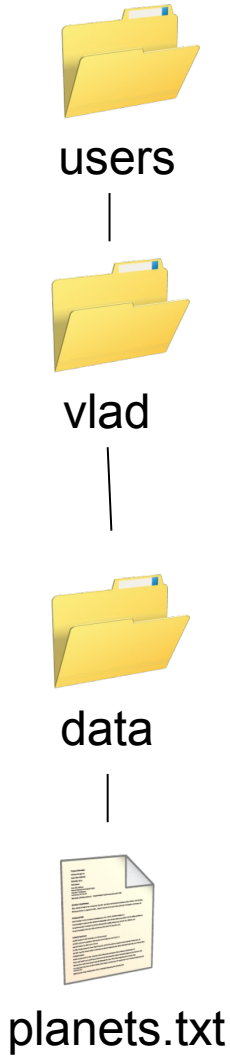
```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
>>> from os.path import abspath
>>> nupath = abspath(path)
>>> print abspath
/users/vlad/data/planets.txt
```

abspath makes relative paths absolute

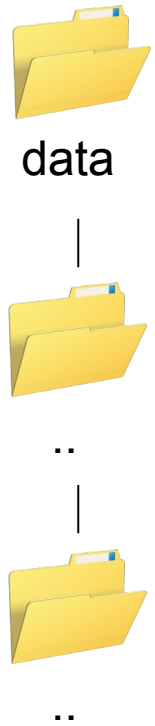
It uses getcwd()



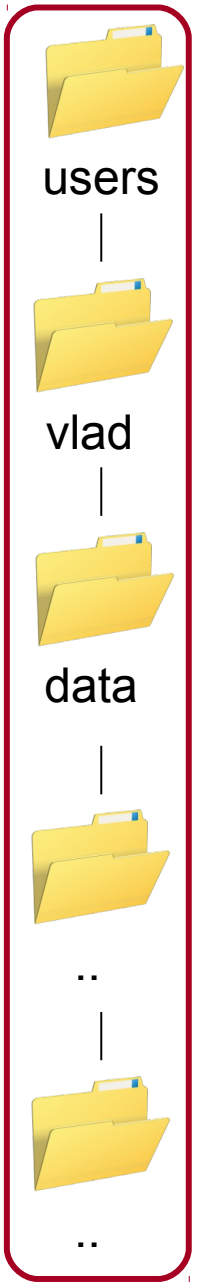
```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
>>> from os.path import abspath
>>> nupath = abspath(path)
>>> print abspath
/users/vlad/data/planets.txt
>>> isabs(nupath)
True
```



```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
>>> from os.path import abspath
>>> nupath = abspath(path)
>>> print abspath
/users/vlad/data/planets.txt
>>> isabs(nupath)
True
>>> abspath('data/../..')
```

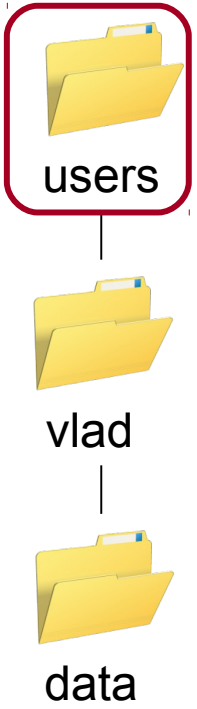


```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
>>> from os.path import abspath
>>> nupath = abspath(path)
>>> print abspath
/users/vlad/data/planets.txt
>>> isabs(nupath)
True
>>> abspath('data/../..')
/users
```



```
>>> path = 'data/planets.txt'
>>> from os.path import isabs
>>> isabs(path)
False
>>> from os.path import abspath
>>> nupath = abspath(path)
>>> print abspath
/users/vlad/data/planets.txt
>>> isabs(nupath)
True
>>> abspath('data/../..')
/users
```

abspath also normalizes the path



Beware!

None of these operations check whether the directories or files exist

Beware!

None of these operations check whether the directories or files exist

Remember `os.path exists` function

`os.path`	Common pathname manipulations
`join`	Join relative paths together using operating system-specific separators
`normpath`	Clean up a path and convert separators to be consistent with the current operating system
`dirname`	Get the path up to the final directory/file in the path
`basename`	Get the final directory/file in the path
`split`	Split a path into directory and file name
`splitext`	Split a path to get a file extension
`splitdrive`	Split a path to get a drive name
`isabs`	Is a path relative or absolute?
`abspath`	Convert a path to an absolute path, using getcwd()

created by

Mike Jackson and Greg Wilson

May 2011