如果我们要操作文件、目录，可以在命令行线面输入操作系统提供的各种命令来完成。比如dir、cp等命令。 如果要在Python程序中执行这些目录和文件的操作怎么办?起始操作系统提供的命令只是简单地调用了操作系统提供的接口函数，Python内置的os模块也可以直接调用操作系统提供的接口函数。 打开Python交互式命令行，我们来看看如何使用os模块的基本功能：

```
>>> import os
>>> os.name
'nt'
```

环境变量

在操作系统中定义的环境变量，全部保留在os.environ这个变量中，可以直接查看：

```
os.environ
environ({'ALLUSERSPROFILE': 'C:\\ProgramData', 'APPDATA': 'C:\\Users\\dell\\AppData\\Roaming', 'CM2014DIR':...})
```

要获取某个环境变量的值，可以调用os.environ.get('key'):

```
>>> os.environ.get('Path')
'D:\\coding\\JAVA\\jdk\\bin;C:\\ProgramData\\Oracle\\Java\\javapath;C:\\Python27\\Scripts;C:\\Program Files\\Common Files\\Siemens\\Autom
```

操作文件和目录

操作文件和目录的函数一部分放在os模块中，一部分放在os,path模块中，这一点要注意一下。查看、创建和删除目录可以这么调用：

```
# 查看当前目录的绝对路径
>>> os.path.abspath('.')
'C:\\Users\\dell\\AppData\\Local\\Programs\\Python\\Python36-32'
# 在某个目录下创建一个新目录，首先把新目录的完整路径表示出来:
>>> os.path.join('D:/编程','testdir')
'D:/编程\\testdir'
# 然后创建一个目录:
>>> os.mkdir('D:/编程/testdir')
# 删掉一个目录
>>> os.rmdir('D:/编程/testdir')
```

把两个路径合成一个，不要直接拼字符串，而要通过os.path.join()函数。同样的道理要拆分路径时，也不要直接去拆字符串，而要通过os.path.split()函数，这样可以把一个路径拆分为两部分，后一部分是最后级别的目录或文件名：

```
# 创建文件
>>> f = open('D:/编程/testdir/nihao.txt','w')
# 分离出目录
>>> os.path.split('D:/编程/testdir/nihao.txt')
('D:/编程/testdir', 'nihao.txt')
# 分离出文件类型
>>> os.path.splitext('D:/编程/testdir/nihao.txt')
('D:/编程/testdir/nihao', '.txt')
# 进入目标目录
>>> os.chdir('D:/编程/testdir')
# 改变文件名
>>> os.rename('test.txt','test.py')
# 获取当前目录
>>> os.getcwd()
'D:\\编程\\testdir'
# 复制文件
>>> import shutil
>>> shutil.copyfile('D:/编程/testdir/nihao.txt','D:/编程/testdir/nihao1.txt')
'D:/编程/testdir/nihao1.txt'
#筛选当前目录下面的文件
>>> [x for x in os.listdir('.') if os.path.isdir(x)]
['test.py']
```

```
>>> [x for x in os.listdir('.') if os.path.isfile(x) and os.path.splitext(x)[1]=='.py']
[]
>>> [x for x in os.listdir('.') if os.path.isfile(x) and os.path.splitext(x)[1]=='.txt']
['nihao.txt', 'nihao1.txt']
```