路径操作

路径操作模块

3.4版本之前

os.path模块

```
from os import path

p = path.join('/etc','sysconfig','network')
print(type(p), p)
print(path.exists(p))
print(path.split(p)) # (head,tail)
print(path.abspath('.'))
p = path.join('o:/', p, 'test.txt')
print(path.dirname(p))
print(path.basename(p))
print(path.splitdrive(p))

p1 = path.abspath(__file__)
print(p1, path.basename(p1))
while p1 != path.dirname(p1):
    p1 = path.dirname(p1)
    print(p1, path.basename(p1))
```

3.4版本开始

建议使用pathlib模块，提供Path对象来操作。包括目录和文件。

pathlib模块

from pathlib import Path

目录操作

初始化

```
p = Path() # 当前目录
p = Path('a','b','c/d') # 当前目录下的a/b/c/d
p = Path('/etc') # 根下的etc目录
```

路径拼接和分解

操作符/

Path对象 / Path对象

Path对象 / 字符串 或者 字符串 / Path对象

分解

parts属性，可以返回路径中的每一个部分

joinpath

joinpath(*other) 连接多个字符串到Path对象中

```
p = Path()
p = p / 'a'
p1 = 'b' / p
p2 = Path('c')
p3 = p2 / p1
print(p3.parts)
p3.joinpath('etc','init.d',Path('httpd'))
```

获取路径

str 获取路径字符串

bytes 获取路径字符串的bytes

```
p = Path('/etc')
print(str(p), bytes(p))
```

父目录

parent 目录的逻辑父目录

parents 父目录序列，索引0是直接的父

```
p = Path('/a/b/c/d')
print(p.parent.parent)
for x in p.parents:
    print(x)
```

name、stem、suffix、suffixes、with_suffix(suffix)、with_name(name)

name 目录的最后一个部分

suffix 目录中最后一个部分的扩展名
stem 目录最后一个部分，没有后缀
suffixes 返回多个扩展名列表
with_suffix(suffix) 补充扩展名到路径尾部，返回新的路径，扩展名存在则无效
with_name(name) 替换目录最后一个部分并返回一个新的路径

```
p = Path('/magedu/mysqlinstall/mysql.tar.gz')
print(p.name)
print(p.suffix)
print(p.suffixes)
print(p.stem)
print(p.with_name('mysql-5.tgz'))
p = Path('README')
print(p.with_suffix('.txt'))
```

cwd() 返回当前工作目录
home() 返回当前家目录

is_dir() 是否是目录，目录存在返回True
is_file() 是否是普通文件，文件存在返回True
is_symlink() 是否是软链接
is_socket() 是否是socket文件
is_block_device() 是否是块设备
is_char_device() 是否是字符设备
is_absolute() 是否是绝对路径

resolve() 返回一个新的路径，这个新路径就是当前Path对象的绝对路径，如果是软链接则直接被解析
absolute() 也可以获取绝对路径，但是推荐使用resolve()

exists() 目录或文件是否存在
rmdir() 删除空目录。没有提供判断目录为空的方法
touch(mode=0o666, exist_ok=True) 创建一个文件
as_uri() 将路径返回成URI，例如'file:///etc/passwd'

mkdir(mode=0o777, parents=False, exist_ok=False)
parents，是否创建父目录，True等同于mkdir -p；False时，父目录不存在，则抛出FileNotFoundError
exist_ok参数，在3.5版本加入。False时，路径存在，抛出FileExistsError；True时，FileExistsError被忽略

iterdir()

迭代当前目录

```
p = Path()
p /= 'a/b/c/d'
p.exists() # True

# 创建目录
p.mkdir() # FileNotFoundError
p.mkdir(parents=True)
p.exists() # True
p.mkdir(parents=True)
p.mkdir(parents=True,exist_ok=True)
p /= 'readme.txt'
p.parent.rmdir() #
p.parent.exists() # False '/a/b/c'
p.mkdir() # FileNotFoundError
p.mkdir(parents=True) # 成功

# 遍历，并判断文件类型，如果是目录是否可以判断其是否为空？
for x in p.parents[len(p.parents)-1].iterdir():
    print(x, end='\t')
    if x.is_dir():
        flag = False
        for _ in x.iterdir():
            flag = True
            break
        # for 循环是否可以使用else子句
        print('dir','Not Empty' if flag else 'Empyt',sep='\t')
    elif x.is_file():
        print('file')
    else:
        print('other file')
```

通配符
glob(pattern) 通配给定的模式
rglob(pattern) 通配给定的模式，递归目录
返回一个生成器

```
list(p.glob('test*')) # 返回当前目录对象下的test开头的文件
list(p.glob('**/*.py')) # 递归所有目录，等同rglob
```

```
g = p.rglob('*.py') # 生成器
next(g)
```

匹配
match(pattern)
模式匹配，成功返回True

```
Path('a/b.py').match('*.py') # True
Path('/a/b/c.py').match('b/*.py') # True
Path('/a/b/c.py').match('a/*.py') # False
Path('/a/b/c.py').match('a/*/*.py') # True
Path('/a/b/c.py').match('a/**/*.py') # True
Path('/a/b/c.py').match('**/*.py') # True
```

stat() 相当于stat命令
lstat() 同stat()，但如果是符号链接，则显示符号链接本身的文件信息

```
$ ln -s test t
from pathlib import Path
p = Path('test')
p.stat()
p1 = Path('t')
p1.stat()
p1.lstat()
```


文件操作

open(mode='r', buffering=-1, encoding=None, errors=None, newline=None)
使用方法类似内建函数open。返回一个文件对象

3.5增加的新函数
read_bytes()
以'rb'读取路径对应文件，并返回二进制流。看源码

read_text(encoding=None, errors=None)
以'rt'方式读取路径对应文件，返回文本。

Path.write_bytes(data)
以'wb'方式写入数据到路径对应文件。

write_text(data, encoding=None, errors=None)

以'wt'方式写入字符串到路径对应文件。

```
p = Path('my_binary_file')
p.write_bytes(b'Binary file contents')
p.read_bytes() # b'Binary file contents'

p = Path('my_text_file')
p.write_text('Text file contents')
p.read_text() # 'Text file contents'


from pathlib import Path
p = Path('o:/test.py')
p.write_text('hello python')
print(p.read_text())
with p.open() as f:
    print(f.read(5))
```