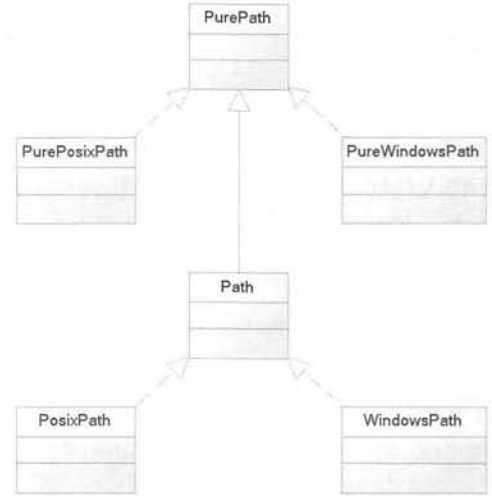
# pathlib模块下的类



PurePath：代表并不访问实际文件系统的”纯路径”。

简单的说，PurePath只负责对路径字符串的操作，置于该字符串是否对应实际的路径，它不关心。

## purepath的基本功能

from pathlib import \*

pp = PurePath('test.py')

print(type(pp))

pp = PurePath('crazyit', 'some/path', 'info')

print(pp)

pp = PurePath(Path('crazyit'), Path('info'))

print(pp)

pp = PurePosixPath('crazyit', 'some/path', 'info')

print(pp)

pp = PureWindowsPath('crazyit', 'some/path', 'info')

print(pp)

## Purepath的属性和方法

from pathlib import \*

print(PurePosixPath('/etc').drive)

print(PurePosixPath('/etc').root)

print(PurePosixPath('/etc').anchor)

print("\n")

pp = PurePath('abc/xyz/wawa/haha')

print(pp.parents[0])

print(pp.parents[1])

print(pp.parents[2])

print(pp.parents[3])

print(pp.parent)

print("\n")

pp = PurePath('abc/wawa/bb.txt')

print(pp.name)

pp = PurePath('abc/wawa/bb.txt.tar.gz')

print(pp.suffixes[0])

print(pp.suffixes[1])

print(pp.suffixes[2])

print(pp.suffix)

print(pp.stem) #返回当前路径重的主文件名

print("=================")

pp=PurePath('/abc', 'xyz', 'wawa', 'haha')

print(pp.as\_posix())

print(pp.as\_uri())

print("\n====test relateive\_to===========")

pp=PurePosixPath('/home/test/flychengwen/aaa.txt')

print(pp.relative\_to('/'))

print(pp.relative\_to('/home'))

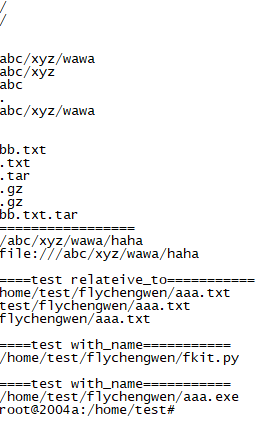
print(pp.relative\_to('/home/test'))

print("\n====test with\_name===========")

print(pp.with\_name('fkit.py'))

print("\n====test with\_name===========")

print(pp.with\_suffix('.exe'))



## path的功能和用法

from pathlib import \*

p = Path('.')

for x in p.iterdir():

print(x)

p = Path('../')

for x in p.glob('\*\*/\*.py'):

print(x)

break

from pathlib import \*

p = Path('test.txt')

res = p.write\_text('''beatifull world''', encoding='GBK')

print(res)

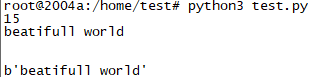
content=p.read\_text(encoding='GBK')

print(content)

print("\n")

bb=p.read\_bytes()

print(bb)



# os.path

import os

import time

print(os.path.abspath("abc.txt"))

print(os.path.commonprefix(['/usr/lib', '/usr/local/lib']))

print(os.path.commonpath(['/usr/lib', '/usr/local/lib']))

print(os.path.dirname('/home/test'))

print(os.path.exists('/home/test/test.py'))

print(time.ctime(os.path.getatime('test.py')))

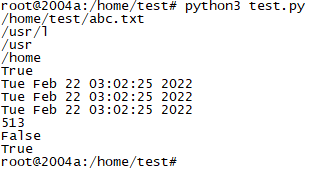
print(time.ctime(os.path.getmtime('test.py')))

print(time.ctime(os.path.getctime('test.py')))

print(os.path.getsize('test.py'))

print(os.path.isfile('/home/chen'))

print(os.path.samefile('test.py', 'test.py'))



# fnmatch

from pathlib import \*

import fnmatch

for file in Path('.').iterdir():

if fnmatch.fnmatch(file, 'test\*'):

print(file)

names=['a.py', 'b.py', 'c.py', 'd.py']

sub = fnmatch.filter(names, '[ac].py')

print(sub)

# open

f=open("111.txt")

print(f.encoding)

print(f.mode)

print(f.closed)

print(f.name)

f.close()

root@2004a:/home/test# python3 test.py

UTF-8

r

False

111.txt

root@2004a:/home/test#

# 读文件

|  |  |
| --- | --- |
| f=open("test.txt", "r", True)  while True:  ch = f.read(1)  if not ch: break  print(ch, end=',')  f.close() | f=open("test.txt", "r", True)  try:  while True:  ch = f.read(1)  if not ch: break  print(ch, end=',')  finally:  f.close() |

使用codecs模块打开文件，允许打开时指定字符集

#按行读取

import codecs

f=codecs.open('test.py', 'r', 'utf-8', buffering=True)

while True:

line = f.readline()

if not line: break

print(line, end='')

f.close()

#使用fileinput读取多个输入流

import fileinput

for line in fileinput.input(files=('info.txt', 'test.txt')):

print(fileinput.filename(), fileinput.filelineno(), line, end=' ')

fileinput.close()

root@2004a:/home/test# python3 test.py

info.txt 1 aaa

info.txt 2 bbb

test.txt 1 111

test.txt 2 222

test.txt 3 333

test.txt 4

root@2004a:/home/test#

#文件迭代器

import codecs

f=codecs.open('test.py', 'r', 'utf-8', buffering=True)

for line in f:

print(line, end='')

f.close()

## with语句

使用with语句的好处是：程序主动关闭文件

|  |  |
| --- | --- |
| import fileinput  with fileinput.input(files=('info.txt', 'test.txt')) as f:  for line in f:  print(line, end='') | import codecs  with codecs.open('test.txt', 'r', 'utf-8', buffering=True) as f:  for line in f:  print(line, end='') |

使用with语句管理的资源必须是一个实现上下文管理协议(context manage protocol)的类

这个类的对象可被称为上下文管理器，要实现上下文管理协议，必须实现如下两个方法：

class FkResource:

def \_\_init\_\_(self, tag):

self.tag=tag

print("constructor: %s" % tag)

def \_\_enter\_\_(self):

print("enter: %s" % self.tag)

return 'fkit'

def \_\_exit\_\_(self, exc\_type, exc\_value, exc\_traceback):

print("leave: %s" % self.tag)

if exc\_traceback is None:

print('close normally')

else:

print('close abnormally')

return False

with FkResource('sunyang') as dr:

print(dr)

print('[with code block] normally')

print('-----------------------------')

with FkResource('zcw') as dr:

print('[with code block] before exception')

raise Exception

print('[with code block] after exception')

## linecache

下面程序示范了使用linecache模块来随机读取指定行

import linecache

import random

print(linecache.getline(random.\_\_file\_\_, 3)) #读取python的random模块的源代码

print(linecache.getline("test.txt", 2))

print(linecache.getline("test.py", 3))

# 写文件

## seek

seek的第二个参数：

0：从文件开始位置

1：从当前位置

2：从文件结束位置，第一个参数应使用负值

root@2004a:/home/test# cat test.py

f=open('test.py', 'rb')

print(f.tell())

f.seek(3)

print(f.tell())

print(f.read(1))

print(f.tell())

f.seek(5)

print(f.tell())

f.seek(5,1)

print(f.tell())

f.seek(-10, 2)

print(f.tell())

print(f.read(1))

root@2004a:/home/test# python3 test.py

0

3

b'p'

4

5

10

192

b'r'

## write和writelines

import os

f = open('test.txt', 'w+')

f.write('I love python ' + os.linesep)

f.writelines(('hello ' + os.linesep,

'welcome ' + os.linesep,

'to ' + os.linesep,

'China ' + os.linesep))

f.close()

# 文件和目录函数

import os

print(os.getcwd())

os.chdir('/home/zcw')

print(os.getcwd())

print('------------------')

os.mkdir('/home/test/my\_dir', 0o755)

os.makedirs('/home/test/aa/bb/cc', 0o755) #递归创建目录

os.rmdir('/home/test/my\_dir')

os.removedirs('/home/test/aa/bb/cc') #aa/bb/cc都会被删除

import os,sys

ret = os.access('.', os.F\_OK | os.R\_OK | os.W\_OK | os.X\_OK)

print("retcode: ", ret)

ret = os.access('test.py', os.F\_OK | os.R\_OK | os.W\_OK | os.X\_OK)

print("retcode: ", ret)

os.chmod('test.txt', stat.S\_IREAD)

ret = os.access('test.py', os.W\_OK)

print("os.W\_OK: retcode=", ret)

重命名函数：

rename

renames

# 文件描述符

import os

f = open('test.txt', "w+")

print(type(f))

f.close()

f = os.open('test.txt', os.O\_RDWR)

print(type(f))

os.close(f)

root@2004a:/home/test# python3 test.py

<class '\_io.TextIOWrapper'>

<class 'int'>

import os

f = os.open('test.txt', os.O\_RDWR)

len1=os.write(f, "hello123\n".encode('utf-8'))

len2=os.write(f, "china456\n".encode('utf-8'))

os.lseek(f, 0, os.SEEK\_SET)

data = os.read(f, len1+len2)

print(data)

print(data.decode('utf-8'))

os.close(f)

root@2004a:/home/test# python3 test.py

b'hello123\nchina456\n'

hello123

china456

# 使用tempfile模块生成临时文件和临时目录

root@2004a:/home/test# cat test.py

import tempfile

fp = tempfile.TemporaryFile()

print(fp.name)

fp.write('abcdefg'.encode('utf-8'))

fp.write('1234567'.encode('utf-8'))

fp.seek(0)

print(fp.read().decode('utf-8'))

fp.close()

print("======================")

with tempfile.TemporaryFile() as fp:

fp.write(b'I love python!')

fp.seek(0)

print(fp.read())

with tempfile.TemporaryDirectory() as tmpdirname:

print("创建临时目录", tmpdirname)

root@2004a:/home/test#

root@2004a:/home/test# python3 test.py

3

abcdefg1234567

======================

b'I love python!'

创建临时目录 /tmp/tmpqze8k970

root@2004a:/home/test#