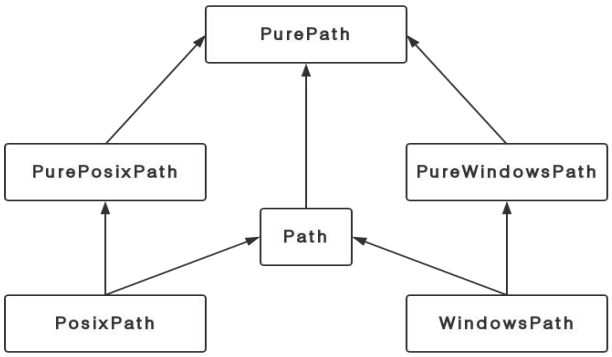
pathlib 常用操作

该模块提供表示文件系统路径的类，其语义适用于不同的操作系统。路径类在纯路径之间划分，纯路径提供纯粹的计算操作，没有 I / O，以及具体路径，它继承纯路径但也提供 I / O 操作。下面是它的 UML 类图：



一般而言主要用的是 Path 类，顶天了再用到 PurePath，下面仅介绍着 2 个类。

一、PurePath 类常用操作

（一）创建对象

语法：**PurePath(*args:str)**

```
# 创建当前目录的路径有以下三种方式
p = PurePath()                                  # PureWindowsPath('.')
p = PurePath(".")                               # PureWindowsPath('.')
p = PurePath("")                                # PureWindowsPath('.')

# 入参只有一个 pathlike 字符时，就创建这个 path 路径
p = PurePath("./d/a.json")                      # PureWindowsPath('d/a.json')

# 入参有多个参数时
# ①当有多个盘符时，最后一个覆盖前面所有盘符，也就是取最后一个盘符作为结果
p = PurePath("c:","d:")                         # PureWindowsPath('d:')
# ②当有多个绝对路径时，同样取最后一个作为结果，
p = PurePath("/abc/sdf/we","/ddd/asd/ert")      # PureWindowsPath('/ddd/asd/ert')
# ③取①和②的综合示例
p = PurePath("c:/a/b/c/d.txt","/f/g/h/r.txt")   # PureWindowsPath('c:/f/g/h/r.txt')
# ④当有多个相对路径时，会用路径分隔符从左到右依次拼接出一个目录
p = PurePath("./d","a.json")                    # PureWindowsPath('d/a.json')
# ⑤绝对路径和相对路径混合出现时，绝对路径也会覆盖之前的相对路径
p = PurePath("a/b/c/y.txt","d:","/f/g/h/r")     # PureWindowsPath('d:/f/g/h/r')

# 综合示例
p = PurePath("c:","a/b/c","/d/e","d:","/root/localhome","workspace","../demo","./test.py")
# 输出：PureWindowsPath('d:/root/localhome/workspace/../demo/test.py')
```

【注】

①“.”表示当前目录，但创建时会过滤掉，不识别

②“..”表示父目录，会识别，拼接到路径中

③路径分隔符可以是“/”，“\”或者“\\”

（二）属性

1. anchor

功能：返回驱动器和根的串联

```
PurePath("c:\d/ff").anchor                    # 'c:\\'
PurePath("\d/ff").anchor                      # '\\'
PurePath("/d/ff").anchor                      # '\\'
PurePath("d:").anchor                         # 'd:'
```

2. drive

功能：返回盘符

```
PurePath("d://").drive                        # 'd:'
PurePath("./da/sd").drive                     # ''
```

3. root

功能：返回根

```
from pathlib import PureWindowsPath, PurePosixPath
PureWindowsPath('c:/Program Files/').root     # '\\'
PureWindowsPath('c:Program Files/').root      # ''
PurePosixPath('/etc').root                    # '/'
```

4. name

功能：返回最终路径组件的字符串，不包括驱动器和根目录（如果有）

```
PurePath("d:/root/localhost").name            # 'localhost'
PurePath("d:/root/localhost/test.py").name    # 'test.py'
PurePath().name                               # ''
```

5. parent

功能：返回路径的逻辑父级

```
PurePath("c:/root/pop/demo.py").parent        # PureWindowsPath('c:/root/pop')
PurePath("c:/root/pop/demo").parent           # PureWindowsPath('c:/root/pop')
```

6. parents

功能：返回给定路径的每层目录，类型为 tuple

```
list(PurePath("c:/root/me/demo").parents)
# 输出：[PureWindowsPath('c:/root/me), PureWindowsPath('c:/root'), PureWindowsPath('c:/')]
list(PurePath("./me/demo").parents)
# 输出：[PureWindowsPath(me), PureWindowsPath('.')]
list(PurePath().parents)
# 输出：[]
list(PurePath("root/pop/demo").parents)
# 输出：[PureWindowsPath('root/pop'), PureWindowsPath('root'), PureWindowsPath('.')]
```

7. parts

功能：返回路径的各个层级，类型为 tuple

```
list(PurePath("c:/root/pop/demo.py").parts)   # ['c:\\', 'root', 'pop', 'demo.py']
list(PurePath("root/pop/demo.py").parts)      # ['root', 'pop', 'demo.py']
```

8. suffix

功能：返回文件的最后一层后缀（如果有）

```
PurePath("data.tar.7z").suffix                    # .7z
```

9. suffixes

功能：返回文件的所有后缀 tuple

```
PurePath("data.tar.7z").suffixes                  # ['.tar', '.7z']
```

（三）方法

1. is_absolute()方法

功能：判断给定路径是否是绝对路径

```
PurePath("root/pop/demo.py").is_absolute()        # False
PurePath("c:/root/pop/demo.py").is_absolute()     # True
```

2. is_reserved()方法

功能：判断路径是否是 Windows 下的保留路径，是返回 True

3. match()方法

功能：匹配路径是否存在

```
PurePath("E:\\",'myStudy',"python","abc.py").match("*.py")        # True
PurePath("E:\\",'myStudy',"python","abc.py").match("./*/*.py")    # True
```

4. as_posix()方法

功能：将所有目录层级分隔符转为“/”

```
PurePath("c:\\root\local\demo\\test.py").as_posix()        # 'c:/root/local/demo/test.py'
```

5. as_uri()方法

功能：路径转换成 URI

```
PurePath("c:\\root\local\demo\\test.py").as_uri()          # 'file:///c:/root/local/demo/test.py'
```

6. joinpath()方法

功能：连接 2 个 path，等价于“/”

```
PurePath("c:/root/local").joinpath("test.py")     #PureWindowsPath('c:/root/local/test.py')
PurePath("c:/root/local") / "test.py"             # PureWindowsPath('c:/root/local/test.py')
```

7. relative_to()方法

功能：返回相对于给定 path 的相对路径

```
PurePath("c:/root/test.py").relative_to("c:/root")     # PureWindowsPath('test.py')
```

8. with_name()方法

功能：更改最后一层文件名，用它创建新的纯路径对象返回，不改变原路径

```
PurePath("c:/root/test.py").with_name("demo.py")   # PureWindowsPath('c:/root/demo.py')
```

9. with_suffix()方法

功能：更改文件的最后一层后缀，用它创建新的纯路径，不改变原路径

```
PurePath("c:/root/test.py.dat").with_suffix(".txt")    # PureWindowsPath('c:/root/test.py.txt')
```

二、Path 类常用操作

（一）创建对象

语法：**Path(*args:str)**

用法和 PurePath 一样，返回的是 PosixPath 或者 WindowsPath

（二）方法

1. cwd()方法

语法：**Path.cwd()**

功能：返回表示当前目录的新路径对象

```
Path.cwd()                                  # WindowsPath('E:/python_workspace')
```

2. home()方法

语法：**Path.home()**

功能：返回代表用户的主目录

```
Path.home()                                 # WindowsPath('C:/Users/Administrator')
```

3. stat()方法

语法：**PathObject.stat()**

功能：返回给定路径的文件或文件夹的信息

```
p =Path("d/a.json")
p.stat()
"""
os.stat_result(st_mode=33206, st_ino=2533274790397213, st_dev=3894667492, st_nlink=1,
               st_uid=0, st_gid=0, st_size=91, st_atime=1558685728, st_mtime=1558685728,
           st_ctime=1558658953)
"""
```

4. chmod()方法

语法：**PathObject.chmod(args)**

功能：更改文件模式和权限

```
p = Path("C3.py")
p.stat().st_mode                            # 33206
p.chmod(0o444)
p.stat().st_mode                            # 33060
```

5. exists()方法

语法：**PathObject.exists()**

功能：判断给定路径是否存在

```
p = Path("er.txt")
p.exists()                                  # False
```

6. expanduser()方法

语法：**PathObject.expanduser()**

功能：将~和~user 替换主目录，Linux 没试过，Windows 就这样

```
Path("~/C3.py").expanduser()                # WindowsPath('C:/Users/Administrator/C3.py')
Path("~user/C3.py").expanduser()            # WindowsPath('C:/Users/user/C3.py')
```

7. glob()方法

语法：**PathObject.glob(pattern)**

功能：搜索符合 pattern 的文件（从右向左匹配），返回一个包含这些文件的生成器

```
p = Path()
list(p.glob("*.py"))
# 输出：[WindowsPath('C3.py'), WindowsPath('test.py'), WindowsPath('TestPath.py')]
list(p.glob("*/*.py"))
# 输出：[WindowsPath('d/a.py')]
list(p.glob("./*.py"))
# 输出：[WindowsPath('C3.py'), WindowsPath('test.py'), WindowsPath('TestPath.py')]
list(p.glob("**/*.py"))
'''输出：
[WindowsPath('C3.py'), WindowsPath('test.py'), WindowsPath('TestPath.py'), WindowsPath('d/a.py')]
'''
```

【注】

①“*”表示任意通配符

②“**”也表示通配符，但是会递归搜索其后所有文件夹的文件，①中的通配符只会搜索给定 pattern 目录的文件

③“.”表示当前目录，“..”表示上层目录

8. is_dir()方法

语法：**PathObject.is_dir()**

功能：判断给定路径是否是目录

9. is_file()方法

语法：**PathObject.is_file()**

功能：判断给定路径是否是文件

10. iterdir()方法

语法：**PathObject.iterdir()**

功能：返回一个生成器，迭代指定路径这下所有的文件和文件夹

```
p = Path()
list(p.iterdir())
'''
[WindowsPath('.idea'), WindowsPath('aa.txt'), WindowsPath('b.json'), WindowsPath('C3.py'),
 WindowsPath('d'), WindowsPath('out.log'), WindowsPath('test.py'),
WindowsPath('TestPath.py'), WindowsPath('__pycache__')]
'''
```

11. mkdir()方法

语法：**PathObject.mkdir(mode=0o777, parents=False, exist_ok=False)**

功能：如果给定路径不存在，则会创建该路径所指向的目录

```
p = Path("local")
p.mkdir()
p.mkdir()                                    # 第二次创建会报错，因为已经存在
```

12. open()函数

语法：**Pathobj.open(mode='r',buffering=-1,encoding=None,errors=None, newline=None)**

实际上一般只用 **PathObject.open(mode='r')**

功能：按照给定模式打开文件，不存在则报错，类似 open()内置函数

```
with p.open("r",encoding="utf-8") as f:
    str1 = f.read()
```

13. read_bytes()方法

语法：**PathObject.read_bytes()**

功能：读取路径指向的文件，返回二进制表示

```
p = Path("C3.py")
p.read_bytes().decode("utf-8")
```

14. read_text()方法

语法：**read_text(encoding = None，errors = None)**

功能：将指向文件的已解码内容作为字符串返回

```
p = Path("C3.py")
p.read_text("utf-8")
```

15. rename()方法

语法：**PathObject.rename(target)**

功能：将此文件或目录重命名为给定目标

```
p = Path("d/abc.txt")
p.exists()                                  # True
p.rename("d/demo.py")
p.exists()                                  # False

p = Path("d/abc.txt")
p.exists()                                  # True
p.rename("d/demo.py")
p.exists()                                  # False
```

16. replace()方法

用法和 rename()一样

17. resolve()方法

语法：PathObject.resolve(strict=False)

功能：使路径绝对，解析任何符号链接。返回一个新的路径对象，

```
p = Path("d/xyz.xml")
p.resolve()                         # WindowsPath('E:/python_workspace/d/xyz.xml')
p = Path("go/rubby/../kt/d/a.json")
p.resolve()                         # WindowsPath('go/kt/d/a.json')
p = Path("rubby/../kt/d/a.json")
p.resolve()                         # WindowsPath('E:/python_workspace/kt/d/a.json')
```

【注】方法会删除..和其之前的一个层级，然后解析绝对路径

18. rmdir()方法

语法：**PathObject.rmdir()**

功能：删除目录，该目录必须为空

19. write_bytes()方法

语法：**PathObject.write_bytes(data)**

功能：向文件写入数据

```
p = Path("d/a.json")
p.write_bytes(b'{"name":"yee","age":18,"sex":"male"}')
```

20. write_text()方法

语法：**PathObject.write_text(data, encoding=None, errors=None)**

功能：向文件写入数据

```
p.write_text('{"name":"yee","age":18,"sex":"male"}')
```