

Backslash on Windows and Forward Slash on macOS and Linux

```
>>> from pathlib import Path
>>> Path('spam', 'bacon', 'eggs')

WindowsPath('spam/bacon/eggs')
>>> str(Path('spam', 'bacon', 'eggs'))
'spam\\bacon\\eggs'

>>> from pathlib import Path
>>> myFiles = ['accounts.txt', 'details.csv', 'invite.docx']
>>> for filename in myFiles:
    print(Path(r'C:\Users\Al', filename))
C:\Users\Al\accounts.txt
C:\Users\Al\details.csv
C:\Users\Al\invite.docx
```

Using the / Operator to Join Paths

```
>>> from pathlib import Path
>>> Path('spam') / 'bacon' / 'eggs'
WindowsPath('spam/bacon/eggs')
>>> Path('spam') / Path('bacon/eggs')
WindowsPath('spam/bacon/eggs')
>>> Path('spam') / Path('bacon', 'eggs')
WindowsPath('spam/bacon/eggs')

>>> homeFolder = r'C:\Users\Al'
>>> subFolder = 'spam'
>>> homeFolder + '\\' + subFolder
'C:\\Users\\Al\\spam'
>>> '\\'.join([homeFolder, subFolder])
'C:\\Users\\Al\\spam'

>>> homeFolder = Path('C:/Users/Al')
>>> subFolder = Path('spam')
>>> homeFolder / subFolder
WindowsPath('C:/Users/Al/spam')
>>> str(homeFolder / subFolder)
'C:\\Users\\Al\\spam'

>>> 'spam' / 'bacon' / 'eggs'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

The Current Working Directory

```
>>> from pathlib import Path
>>> import os
```

```
>>> Path.cwd()
WindowsPath('C:/Users/Al/AppData/Local/Programs/Python/Python37')
>>> os.chdir('C:\\Windows\\System32')
>>> Path.cwd()
WindowsPath('C:/Windows/System32')

>>> os.chdir('C:/ThisFolderDoesNotExist')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [WinError 2] The system cannot find the file specified:
'C:/ThisFolderDoesNotExist'
```

The Home Directory

```
>>> Path.home()
WindowsPath('C:/Users/Al')
```

Absolute vs. Relative Paths

Creating New Folders Using the os.makedirs() Function

```
>>> import os
>>> os.makedirs('C:\\delicious\\walnut\\waffles')

>>> from pathlib import Path
>>> Path(r'C:\Users\Al\spam').mkdir()
```

Handling Absolute and Relative Paths

```
>>> Path.cwd()
WindowsPath('C:/Users/Al/AppData/Local/Programs/Python/Python37')
>>> Path.cwd().is_absolute()
True
>>> Path('spam/bacon/eggs').is_absolute()
False

>>> Path('my/relative/path')
WindowsPath('my/relative/path')
>>> Path.cwd() / Path('my/relative/path')
WindowsPath('C:/Users/Al/AppData/Local/Programs/Python/Python37/my/relative/
path')

>>> Path('my/relative/path')
WindowsPath('my/relative/path')
>>> Path.home() / Path('my/relative/path')
WindowsPath('C:/Users/Al/my/relative/path')

>>> os.path.abspath('.')
'C:\\Users\\Al\\AppData\\Local\\Programs\\Python\\Python37'
```

```

>>> os.path.abspath('..\Scripts')
'C:\\Users\\Al\\AppData\\Local\\Programs\\Python\\Python37\\Scripts'
>>> os.path.isabs('.')
False
>>> os.path.isabs(os.path.abspath('.'))
True

>>> os.path.relpath('C:\\Windows', 'C:\\')
'Windows'
>>> os.path.relpath('C:\\Windows', 'C:\\spam\\eggs')
'..\\..\\Windows'

```

Getting the Parts of a File Path

```

>>> p = Path('C:/Users/Al/spam.txt')
>>> p.anchor
'C:\\'
>>> p.parent # This is a Path object, not a string.
WindowsPath('C:/Users/Al')
>>> p.name
'spam.txt'
>>> p.stem
'spam'
>>> p.suffix
'.txt'
>>> p.drive
'C:'

>>> Path.cwd()
WindowsPath('C:/Users/Al/AppData/Local/Programs/Python/Python37')
>>> Path.cwd().parents[0]
WindowsPath('C:/Users/Al/AppData/Local/Programs/Python')
>>> Path.cwd().parents[1]
WindowsPath('C:/Users/Al/AppData/Local/Programs')
>>> Path.cwd().parents[2]
WindowsPath('C:/Users/Al/AppData/Local')
>>> Path.cwd().parents[3]
WindowsPath('C:/Users/Al/AppData')
>>> Path.cwd().parents[4]
WindowsPath('C:/Users/Al')
>>> Path.cwd().parents[5]
WindowsPath('C:/Users')
>>> Path.cwd().parents[6]
WindowsPath('C:/')

>>> calcFilePath = 'C:\\Windows\\System32\\calc.exe'
>>> os.path.basename(calcFilePath)
'calc.exe'
>>> os.path.dirname(calcFilePath)
'C:\\Windows\\System32'

```

```
>>> calcFilePath = 'C:\\Windows\\System32\\calc.exe'
>>> os.path.split(calcFilePath)
('C:\\Windows\\System32', 'calc.exe')

>>> (os.path.dirname(calcFilePath), os.path.basename(calcFilePath))
('C:\\Windows\\System32', 'calc.exe')

>>> calcFilePath.split(os.sep)
['C:', 'Windows', 'System32', 'calc.exe']

>>> '/usr/bin'.split(os.sep)
['', 'usr', 'bin']
```

Finding File Sizes and Folder Contents

```
>>> os.path.getsize('C:\\Windows\\System32\\calc.exe')
27648
>>> os.listdir('C:\\Windows\\System32')
['0409', '12520437.cpx', '12520850.cpx', '5U877.ax', 'aaclient.dll',
--snip--
'xwtpdud.dll', 'xwtpw32.dll', 'zh-CN', 'zh-HK', 'zh-TW', 'zipfldr.dll']

>>> totalSize = 0
>>> for filename in os.listdir('C:\\Windows\\System32'):
    totalSize = totalSize + os.path.getsize(os.path.join('C:\\Windows\\
\\System32', filename))
>>> print(totalSize)
2559970473
```

Modifying a List of Files Using Glob Patterns

```
>>> p = Path('C:/Users/Al/Desktop')
>>> p.glob('*')
<generator object Path.glob at 0x000002A6E389DED0>
>>> list(p.glob('*')) # Make a list from the generator.
[WindowsPath('C:/Users/Al/Desktop/1.png'), WindowsPath('C:/Users/Al/
Desktop/22-ap.pdf'), WindowsPath('C:/Users/Al/Desktop/cat.jpg'),
--snip--
WindowsPath('C:/Users/Al/Desktop/zzz.txt')]

>>> list(p.glob('*.txt')) # Lists all text files.
[WindowsPath('C:/Users/Al/Desktop/foo.txt'),
--snip--
WindowsPath('C:/Users/Al/Desktop/zzz.txt')]

>>> list(p.glob('project?.docx'))
[WindowsPath('C:/Users/Al/Desktop/project1.docx'), WindowsPath('C:/Users/
Al/
Desktop/project2.docx'),
--snip--
WindowsPath('C:/Users/Al/Desktop/project9.docx')]
```

```
>>> list(p.glob('*.?x?'))
[WindowsPath('C:/Users/Al/Desktop/calc.exe'), WindowsPath('C:/Users/Al/
Desktop/foo.txt'),
  --snip--
WindowsPath('C:/Users/Al/Desktop/zzz.txt')]

>>> p = Path('C:/Users/Al/Desktop')
>>> for textFilePathObj in p.glob('*.txt'):
...     print(textFilePathObj) # Prints the Path object as a string.
...     # Do something with the text file.
...
C:\Users\Al\Desktop\foo.txt
C:\Users\Al\Desktop\spam.txt
C:\Users\Al\Desktop\zzz.txt
```

Checking Path Validity

```
>>> winDir = Path('C:/Windows')
>>> notExistsDir = Path('C:/This/Folder/Does/Not/Exist')
>>> calcFile = Path('C:/Windows
/System32/calc.exe')
>>> winDir.exists()
True
>>> winDir.is_dir()
True
>>> notExistsDir.exists()
False
>>> calcFile.is_file()
True
>>> calcFile.is_dir()
False

>>> dDrive = Path('D:/')
>>> dDrive.exists()
False
```

The File Reading/Writing Process

```
>>> from pathlib import Path
>>> p = Path('spam.txt')
>>> p.write_text('Hello, world!')
13
>>> p.read_text()
'Hello, world!'
```

Opening Files with the open() Function

```
>>> helloFile = open(Path.home() / 'hello.txt')

>>> helloFile = open('C:\\Users\\your_home_folder\\hello.txt')
```

```
>>> helloFile = open('/Users/your_home_folder/hello.txt')
```

Reading the Contents of Files

```
>>> helloContent = helloFile.read()
>>> helloContent
'Hello, world!'
```

When, in disgrace with fortune and men's eyes,
I all alone beweeep my outcast state,
And trouble deaf heaven with my bootless cries,
And look upon myself and curse my fate,

```
>>> sonnetFile = open(Path.home() / 'sonnet29.txt')
>>> sonnetFile.readlines()
[When, in disgrace with fortune and men's eyes,\n', ' I all alone beweeep my
outcast state,\n', And trouble deaf heaven with my bootless cries,\n', And
look upon myself and curse my fate,']
```

Writing to Files

```
>>> baconFile = open('bacon.txt', 'w')
>>> baconFile.write('Hello, world!\n')
13
>>> baconFile.close()
>>> baconFile = open('bacon.txt', 'a')
>>> baconFile.write('Bacon is not a vegetable.')
25
>>> baconFile.close()
>>> baconFile = open('bacon.txt')
>>> content = baconFile.read()
>>> baconFile.close()
>>> print(content)
Hello, world!
Bacon is not a vegetable.
```

Saving Variables with the shelve Module

```
>>> import shelve
>>> shelfFile = shelve.open('mydata')
>>> cats = ['Zophie', 'Pooka', 'Simon']
>>> shelfFile['cats'] = cats
>>> shelfFile.close()

>>> shelfFile = shelve.open('mydata')
>>> type(shelfFile)
<class 'shelve.DbfilenameShelf'>
>>> shelfFile['cats']
['Zophie', 'Pooka', 'Simon']
>>> shelfFile.close()
```

```
>>> shelfFile = shelve.open('mydata')
>>> list(shelfFile.keys())
['cats']
>>> list(shelfFile.values())
[['Zophie', 'Pooka', 'Simon']]
>>> shelfFile.close()
```

Saving Variables with the pprint.pformat() Function

```
>>> import pprint
>>> cats = [{'name': 'Zophie', 'desc': 'chubby'}, {'name': 'Pooka', 'desc':
'fluffy'}]
>>> pprint.pformat(cats)
"[{'desc': 'chubby', 'name': 'Zophie'}, {'desc': 'fluffy', 'name': 'Pooka'}]"
>>> fileObj = open('myCats.py', 'w')
>>> fileObj.write('cats = ' + pprint.pformat(cats) + '\n')
83
>>> fileObj.close()

>>> import myCats
>>> myCats.cats
[{'name': 'Zophie', 'desc': 'chubby'}, {'name': 'Pooka', 'desc': 'fluffy'}]
>>> myCats.cats[0]
{'name': 'Zophie', 'desc': 'chubby'}
>>> myCats.cats[0]['name']
'Zophie'
```