ERROR: Could not install packages due to an OSError: [Errno 2] No such file or directory: 'C:\\Users\\chuck\\OneDrive\\DOCUMENTS\\CERTIFICATIONS\\PYTHON AND DJANGO\\Django 3 - Full Stack Websites with Python Web Development {udemy}\\password\_generator\_project\\.venv\\Lib\\site-packages\\django\\contrib\\contenttypes\\management\\commands\\remove\_stale\_contenttypes.py'

HINT: This error might have occurred since this system does not have Windows Long Path support enabled. You can find information on how to enable this at <https://pip.pypa.io/warnings/enable-long-paths>

**Maximum Path Length Limitation**

* Article
* 04/22/2021
* 4 minutes to read
* 8 contributors

In the Windows API (with some exceptions discussed in the following paragraphs), the maximum length for a path is **MAX\_PATH**, which is defined as 260 characters. A local path is structured in the following order: drive letter, colon, backslash, name components separated by backslashes, and a terminating null character. For example, the maximum path on drive D is "D:\*some 256-character path string*<NUL>" where "<NUL>" represents the invisible terminating null character for the current system codepage. (The characters < > are used here for visual clarity and cannot be part of a valid path string.)

For example, you may hit this limitation if you are cloning a git repo that has long file names into a folder that itself has a long name.

**Note**

File I/O functions in the Windows API convert "/" to "\" as part of converting the name to an NT-style name, except when using the "\\?\" prefix as detailed in the following sections.

The Windows API has many functions that also have Unicode versions to permit an extended-length path for a maximum total path length of 32,767 characters. This type of path is composed of components separated by backslashes, each up to the value returned in the *lpMaximumComponentLength* parameter of the [**GetVolumeInformation**](https://docs.microsoft.com/en-us/windows/desktop/api/FileAPI/nf-fileapi-getvolumeinformationa) function (this value is commonly 255 characters). To specify an extended-length path, use the "\\?\" prefix. For example, "\\?\D:\*very long path*".

**Note**

The maximum path of 32,767 characters is approximate, because the "\\?\" prefix may be expanded to a longer string by the system at run time, and this expansion applies to the total length.

The "\\?\" prefix can also be used with paths constructed according to the universal naming convention (UNC). To specify such a path using UNC, use the "\\?\UNC\" prefix. For example, "\\?\UNC\server\share", where "server" is the name of the computer and "share" is the name of the shared folder. These prefixes are not used as part of the path itself. They indicate that the path should be passed to the system with minimal modification, which means that you cannot use forward slashes to represent path separators, or a period to represent the current directory, or double dots to represent the parent directory. Because you cannot use the "\\?\" prefix with a relative path, relative paths are always limited to a total of **MAX\_PATH** characters.

There is no need to perform any Unicode normalization on path and file name strings for use by the Windows file I/O API functions because the file system treats path and file names as an opaque sequence of **WCHAR**s. Any normalization that your application requires should be performed with this in mind, external of any calls to related Windows file I/O API functions.

When using an API to create a directory, the specified path cannot be so long that you cannot append an 8.3 file name (that is, the directory name cannot exceed **MAX\_PATH** minus 12).

The shell and the file system have different requirements. It is possible to create a path with the Windows API that the shell user interface is not able to interpret properly.

**Enable Long Paths in Windows 10, Version 1607, and Later**

Starting in Windows 10, version 1607, **MAX\_PATH** limitations have been removed from common Win32 file and directory functions. However, you must opt-in to the new behavior.

To enable the new long path behavior, both of the following conditions must be met:

* The registry key Computer\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem\LongPathsEnabled (Type: REG\_DWORD) must exist and be set to 1. The key's value will be cached by the system (per process) after the first call to an affected Win32 file or directory function (see below for the list of functions). The registry key will not be reloaded during the lifetime of the process. In order for all apps on the system to recognize the value of the key, a reboot might be required because some processes may have started before the key was set.

You can also copy this code to a .reg file which can set this for you, or use the PowerShell command from a terminal window with elevated privileges:

* [cmd](https://docs.microsoft.com/en-us/windows/win32/fileio/maximum-file-path-limitation?tabs=powershell#tabpanel_1_cmd)
* [PowerShell](https://docs.microsoft.com/en-us/windows/win32/fileio/maximum-file-path-limitation?tabs=powershell#tabpanel_1_powershell)

PowerShellCopy

New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem" `

-Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force

**Note**

This registry key can also be controlled via Group Policy at Computer Configuration > Administrative Templates > System > Filesystem > Enable Win32 long paths.

* The [application manifest](https://docs.microsoft.com/en-us/windows/win32/sbscs/application-manifests) must also include the longPathAware element.

XMLCopy

<application xmlns="urn:schemas-microsoft-com:asm.v3">

<windowsSettings xmlns:ws2="http://schemas.microsoft.com/SMI/2016/WindowsSettings">

<ws2:longPathAware>true</ws2:longPathAware>

</windowsSettings>

</application>

These are the directory management functions that no longer have **MAX\_PATH** restrictions if you opt-in to long path behavior: CreateDirectoryW, CreateDirectoryExW GetCurrentDirectoryW RemoveDirectoryW SetCurrentDirectoryW.

These are the file management functions that no longer have **MAX\_PATH** restrictions if you opt-in to long path behavior: CopyFileW, CopyFile2, CopyFileExW, CreateFileW, CreateFile2, CreateHardLinkW, CreateSymbolicLinkW, DeleteFileW, FindFirstFileW, FindFirstFileExW, FindNextFileW, GetFileAttributesW, GetFileAttributesExW, SetFileAttributesW, GetFullPathNameW, GetLongPathNameW, MoveFileW, MoveFileExW, MoveFileWithProgressW, ReplaceFileW, SearchPathW, FindFirstFileNameW, FindNextFileNameW, FindFirstStreamW, FindNextStreamW, GetCompressedFileSizeW, GetFinalPathNameByHandleW.