Skip to content

File Paths & String Encoding

Relative File Paths

Blender's relative file paths are not compatible with standard Python modules such as `sys` and `os`. Built-in Python functions don't understand Blender's `//` prefix which denotes the blend-file path.

A common case where you would run into this problem is when exporting a material with associated image paths:

```
>>> bpy.path.abspath(image.filepath)
```

When using Blender data from linked libraries there is an unfortunate complication since the path will be relative to the library rather than the open blend-file. When the data block may be from an external blend-file pass the library argument from the `bpy.types.ID`.

```
>>> bpy.path.abspath(image.filepath, library=image.library)
```

These returns the absolute path which can be used with native Python modules.

Unicode Problems

Python supports many different encodings so there is nothing stopping you from writing a script in `latin1` or `iso-8859-15`. See PEP 263.

However, this complicates matters for Blender's Python API because `.blend` files don't have an explicit encoding. To avoid the problem for Python integration and script authors we have decided that all strings in blend-files **must** be `UTF-8`, `ASCII` compatible. This means assigning strings with different encodings to an object name, for instance, will raise an error.

Paths are an exception to this rule since the existence of non-UTF-8 paths on the user's file system cannot be ignored. This means seemingly harmless expressions can raise errors, e.g:

```
>>> print(bpy.data.filepath)
UnicodeEncodeError: 'ascii' codec can't encode characters in position 10-21: ordinal not i
```

```
>>> bpy.context.object.name = bpy.data.filepath
Traceback (most recent call last):
  File "<blender_console>", line 1, in <module>
TypeError: bpy_struct: item.attr= val: Object.name expected a string type, not str
```

Here are two ways around file-system encoding issues:

```
>>> print(repr(bpy.data.filepath))
```

```
>>> import os
>>> filepath_bytes = os.fsencode(bpy.data.filepath)
>>> filepath_utf8 = filepath_bytes.decode('utf-8', "replace")
>>> bpy.context.object.name = filepath_utf8
```

Unicode encoding/decoding is a big topic with comprehensive Python documentation, to keep it short about encoding problems – here are some suggestions:

- Always use UTF-8 encoding or convert to UTF-8 where the input is unknown.
- Avoid manipulating file paths as strings directly, use `os.path` functions instead.
- Use `os.fsencode()` or `os.fsdecode()` instead of built-in string decoding functions when operating on paths.
- To print paths or to include them in the user interface use `repr(path)` first or `"%r" % path` with string formatting.

Note

Sometimes it’s preferable to avoid string encoding issues by using bytes instead of Python strings, when reading some input it’s less trouble to read it as binary data though you will still need to decide how to treat any strings you want to use with Blender, some importers do this.