5.15 打印不合法的文件名¶

问题¶

你的程序获取了一个目录中的文件名列表，但是当它试着去打印文件名的时候程序崩溃， 出现了 `UnicodeEncodeError` 异常和一条奇怪的消息——`surrogates not allowed` 。

解决方案¶

当打印未知的文件名时，使用下面的方法可以避免这样的错误：

```
def bad_filename(filename):
    return repr(filename)[1:-1]

try:
    print(filename)
except UnicodeEncodeError:
    print(bad_filename(filename))
```

讨论¶

这一小节讨论的是在编写必须处理文件系统的程序时一个不太常见但又很棘手的问题。 默认情况下，Python假定所有文件名都已经根据 `sys.getfilesystemencoding()` 的值编码过了。 但是，有一些文件系统并没有强制要求这样做，因此允许创建文件名没有正确编码的文件。 这种情况不太常见，但是总会有些用户冒险这样做或者是无意之中这样做了(可能是在一个有缺陷的代码中给 `open()` 函数传递了一个不合规范的文件名)。

当执行类似 `os.listdir()` 这样的函数时，这些不合规范的文件名就会让Python陷入困境。 一方面，它不能仅仅只是丢弃这些不合格的名字。而另一方面，它又不能将这些文件名转换为正确的文本字符串。 Python对这个问题的解决方案是从文件名中获取未解码的字节值比如 `\xhh` 并将它映射成Unicode字符 `\udchh` 表示的所谓的”代理编码”。 下面一个例子演示了当一个不合格目录列表中含有一个文件名为bäd.txt(使用Latin-1而不是UTF-8编码)时的样子：

```
>>> import os
>>> files = os.listdir('.')
>>> files
['spam.py', 'b\udce4d.txt', 'foo.txt']
>>>
```

如果你有代码需要操作文件名或者将文件名传递给 `open()` 这样的函数，一切都能正常工作。 只有当你想要输出文件名时才会碰到些麻烦(比如打印输出到屏幕或日志文件等)。 特别的，当你想打印上面的文件名列表时，你的程序就会崩溃：

```
>>> for name in files:
...     print(name)
...
spam.py
Traceback (most recent call last):
    File "<stdin>", line 2, in <module>
UnicodeEncodeError: 'utf-8' codec can't encode character '\udce4' in
position 1: surrogates not allowed
>>>
```

程序崩溃的原因就是字符 `\udce4` 是一个非法的Unicode字符。 它其实是一个被称为代理字符对的双字符组合的后半部分。 由于缺少了前半部分，因此它是个非法的Unicode。 所以，唯一能成功输出的方法就是当遇到不合法文件名时采取相应的补救措施。 比如可以将上述代码修改如下：

```
>>> for name in files:
... try:
...     print(name)
... except UnicodeEncodeError:
...     print(bad_filename(name))
...
spam.py
b\udce4d.txt
foo.txt
>>>
```

在 `bad_filename()` 函数中怎样处置取决于你自己。 另外一个选择就是通过某种方式重新编码，示例如下：

```
def bad_filename(filename):
   temp = filename.encode(sys.getfilesystemencoding(), errors='surrogateescape')
   return temp.decode('latin-1')
```

译者注:

surrogateescape:
这种是Python在绝大部分面向OS的API中所使用的错误处理器，
它能以一种优雅的方式处理由操作系统提供的数据的编码问题。
在解码出错时会将出错字节存储到一个很少被使用到的Unicode编码范围内。
在编码时将那些隐藏值又还原回原先解码失败的字节序列。
它不仅对于OS API非常有用，也能很容易的处理其他情况下的编码错误。

使用这个版本产生的输出如下：

```
>>> for name in files:
...     try:
...         print(name)
...     except UnicodeEncodeError:
...         print(bad_filename(name))
...
spam.py
bäd.txt
foo.txt
>>>
```

这一小节主题可能会被大部分读者所忽略。但是如果你在编写依赖文件名和文件系统的关键任务程序时， 就必须得考虑到这个。否则你可能会在某个周末被叫到办公室去调试一些令人费解的错误。