```
import unicodedata

def compare_caseless(s1, s2):
    def NFD(s):
        return unicodedata.normalize('NFD', s)

    return NFD(NFD(s1).casefold()) == NFD(NFD(s2).casefold())

# Example usage
single_char = 'ê'
multiple_chars = '\N{LATIN CAPITAL LETTER E}\N{COMBINING CIRCUMFLEX ACCENT}'

print(compare_caseless(single_char, multiple_chars))
```

This will print `True`. (Why is `NFD()` invoked twice? Because there are a few characters that make `casefold()` return a non-normalized string, so the result needs to be normalized again. See section 3.13 of the Unicode Standard for a discussion and an example.)

## 2.6 Unicode Regular Expressions

The regular expressions supported by the `re` module can be provided either as bytes or strings. Some of the special character sequences such as `\d` and `\w` have different meanings depending on whether the pattern is supplied as bytes or a string. For example, `\d` will match the characters `[0-9]` in bytes but in strings will match any character that's in the `'Nd'` category.

The string in this example has the number 57 written in both Thai and Arabic numerals:

```
import re
p = re.compile(r'\d+')

s = "Over \u0e55\u0e57 57 flavours"
m = p.search(s)
print(repr(m.group()))
```

When executed, `\d+` will match the Thai numerals and print them out. If you supply the `re.ASCII` flag to `compile()`, `\d+` will match the substring "57" instead.

Similarly, `\w` matches a wide variety of Unicode characters but only `[a-zA-Z0-9_]` in bytes or if `re.ASCII` is supplied, and `\s` will match either Unicode whitespace characters or `[ \t\n\r\f\v]`.

## 2.7 References

Some good alternative discussions of Python's Unicode support are:

- Processing Text Files in Python 3, by Nick Coghlan.
- Pragmatic Unicode, a PyCon 2012 presentation by Ned Batchelder.

The `str` type is described in the Python library reference at textseq.

The documentation for the `unicodedata` module.

The documentation for the `codecs` module.

Marc-André Lemburg gave a presentation titled "Python and Unicode" (PDF slides) at EuroPython 2002. The slides are an excellent overview of the design of Python 2's Unicode features (where the Unicode string type is called `unicode` and literals start with `u`).