Converting strings

Before Python 3.0, string characters were stored by their ASCII numeric code values in the range 0-127, representing only un-accented Latin characters. For example, the lowercase letter 'a' is assigned 97 as its code value. Each byte of computer memory can, in fact, store values in the range 0-255 but this is still too limited to represent all accented characters and non-Latin characters. For example, accented characters used in Western Europe and the Cyrillic alphabet used for Russian cannot be represented in the range 128-255 because there are more than 127 such characters. Recent versions of Python overcome this limitation by storing string characters as their Unicode code point value to represent all characters and alphabets in the numeric range 0-1,114,111. Characters that are above the ASCII range may require two bytes for their code point value, such as hexadecimal **0xC3 0xB6** for 'ö'.

Hot tip

The term "ASCII" is an acronym for American Standard Code for Information Interchange.

The **str** object's **encode()** method can be used to convert from the default Unicode encoding and its **decode()** method can be used to convert back to the Unicode default encoding.

Python's "unicodedata" module, usefully, provides a **name()** method that reveals the Unicode name of each character. Accented and non-Latin characters can be referenced by their Unicode name or by decoding their Unicode hexadecimal code point value.

unicode.py

1. Start a new Python script by initializing a variable with a string containing a non-ASCII character then display its value, data type, and string length

```
s = 'Röd'
print( '\nRed String:' , s )
print( 'Type:' , type( s ) , '\tLength:' , len( s ) )
```

2. Next, encode the string and again display its value, data type, and string length

```
s = s.encode( 'utf-8' )
print( '\nEncoded String:' , s )
print( 'Type:' , type( s ) , '\tLength:' , len( s ) )
```

3. Now, decode the string and once more display its value, data type, and string length – to reveal the hexadecimal code point of the non-ASCII character

```
s = s.decode( 'utf-8' )
print( '\nDecoded String:' , s )
print( 'Type:' , type( s ) , '\tLength:' , len( s ) )
```

Hot tip

You can use the Character Map app in Windows Accessories to select non-ASCII characters.