Working with characters in Python

Python programs can work with **ASCII** and **Unicode** character sets.

ASCII comes in two versions. The standard version uses **7 bits**, and so can represent **128** different characters potentially. This includes all the numbers, the English alphabet, and various common symbols like + - / * etc.

The **first 32** characters in ASCII (0-31) are **non-printing** symbols, that is they represent a character that you can't see on the screen. The rest of the characters (32-127) represent symbols we can see (including space) which we can usually type on a keyboard. Some of the non printing characters are very important, for example the line feed character (ASCII 10) forces printing to continue on a new line. The tab character (ASCII 9) moves the print position across horizontally to the right for a certain number of characters (how many is dependent on the system settings)

ASCII can only represent the **English** language fully. Symbols for other Western European languages are missing. For example the French characters é and ç or the Spanish ñ and German ö. Even the pound symbol £ is not present in 7 bit ASCII. For this reason, **extended ASCII** which uses **8 bits**, was developed.

Unfortunately, there is no single standard for 8 bit ASCII so it can vary among different computer systems. However it does allow many languages from **Western Europe** to be fully represented.

The most recent character set to be developed is **Unicode**. Because there are only 256 characters available in an 8 bit system, Unicode uses up to **32 bits**. Written in hexadecimal, this would be the numbers 0x0 to 0xFFFFFFFF – a massive range! This can represent all current and many past languages. ASCII is contained inside Unicode and is described as a **subset** of Unicode. You could also describe Unicode as a **superset** of ASCII.

Some example ranges within the full set of Unicode characters are:

0x600 to 0x6ff	(Arabic characters)
0x400 to 0x4ff	(Russian and Ukrainian characters)
0xB80 to 0xBFF	(Tamil characters)
0x4E00 to 0x9FFF	(Chinese, Japanese and Korean characters, known as CJK).

Below are some code techniques you will commonly need to use in Python programs to perform various tasks with characters.

<table>
<tr><th></th><th>Purpose</th><th>Example</th></tr>
<tr><td>chr()</td><td>Print a single character onto the screen. Supply an integer inside the brackets and the equivalent ASCII or Unicode character will be printed.

In the first example, the copyright symbol (©) will be printed. In the second example a pound (£) sign is printed before the number stored in myNum.</td><td>print chr(169)

i=163
print(chr(i), myNum)</td></tr>
<tr><td>ord()</td><td>Does the reverse of the chr() function. If you give it a character inside the brackets, it will return the ASCII or Unicode value of the character. In the first example, 65 will be printed on the screen, in the second, 33 will be printed on the screen.</td><td>print (ord("A"))

myChar="!"
print (ord(myChar))</td></tr>
<tr><td>end=""</td><td>When used as the last item inside a print command, will continue print on the same line, instead of automatically forcing new output onto the next line as would normally happen.</td><td>print(chr(i),end="")</td></tr>
<tr><td><table><tr><td>\\</td><td>Backslash (\)</td></tr><tr><td>\'</td><td>Single quote (')</td></tr><tr><td>\"</td><td>Double quote (")</td></tr><tr><td>\n</td><td>Linefeed (LF)</td></tr><tr><td>\t</td><td>Horizontal Tab (TAB)</td></tr></table></td><td>These escape sequences allow you to put characters inside a string that would not normally be permitted. For example if you want to print speech marks(double quotes) inside speech marks, Python will give a syntax error unless you put a backward slash in before the speech mark.

The first example will print "Quickly, Harry!", said the wizard to the screen because the escape sequence for quotes has been used. In the second example C:\new folder will be printed on the screen because the escape sequence for backslash has been used.</td><td>print("\"Quickly, Harry!\", said the wizard")

print("C:\\new folder")</td></tr>
<tr><td>0x41</td><td>Putting 0x in front of a number is a way of specifying an integer as a hexadecimal value in Python programs. In the first example it will print the copyright symbol, because the value of 169 in hexadecimal is A9. In the second example it will print a Chinese character that represents man/mankind.</td><td>print(chr(0xA9))

print(chr(0x4EBA))</td></tr>
</table>