1.

capitalize()	Converts the first character of the string to a capital (uppercase) letter
casefold()	Implements caseless string matching
center()	Pad the string with the specified character.
count()	Returns the number of occurrences of a substring in the string.
encode()	Encodes strings with the specified encoded scheme
endswith()	Returns “True” if a string ends with the given suffix
expandtabs()	Specifies the amount of space to be substituted with the “\t” symbol in the string
find()	Returns the lowest index of the substring if it is found
format()	Formats the string for printing it to console
format_map()	Formats specified values in a string using a dictionary
index()	Returns the position of the first occurrence of a substring in a string
isalnum()	Checks whether all the characters in a given string is alphanumeric or not
isalpha()	Returns “True” if all characters in the string are alphabets
isdecimal()	Returns true if all characters in a string are decimal
isdigit()	Returns “True” if all characters in the string are digits
isidentifier()	Check whether a string is a valid identifier or not
islower()	Checks if all characters in the string are lowercase
isnumeric()	Returns “True” if all characters in the string are numeric characters
isprintable()	Returns “True” if all characters in the string are printable or the string is empty
isspace()	Returns “True” if all characters in the string are whitespace characters
istitle()	Returns “True” if the string is a title cased string
isupper()	Checks if all characters in the string are uppercase
join()	Returns a concatenated String
ljust()	Left aligns the string according to the width specified
lower()	Converts all uppercase characters in a string into lowercase
lstrip()	Returns the string with leading characters removed
maketrans()	Returns a translation table
partition()	Splits the string at the first occurrence of the separator
replace()	Replaces all occurrences of a substring with another substring
rfind()	Returns the highest index of the substring
rindex()	Returns the highest index of the substring inside the string
rjust()	Right aligns the string according to the width specified
rpartition()	Split the given string into three parts
rsplit()	Split the string from the right by the specified separator
rstrip()	Removes trailing characters
splitlines()	Split the lines at line boundaries
startswith()	Returns “True” if a string starts with the given prefix
strip()	Returns the string with both leading and trailing characters
swapcase()	Converts all uppercase characters to lowercase and vice versa
title()	Convert string to title case
translate()	Modify string according to given translation mappings
upper()	Converts all lowercase characters in a string into uppercase
zfill()	Returns a copy of the string with ‘0’ characters padded to the left side of the string

2.

To create a string, put the sequence of characters inside either single quotes, double quotes, or triple quotes and then assign it to a variable. You can look into how variables work in Python in the Python variables tutorial. For example, you can assign a character 'a' to a variable single_quote_character .

3.

In order to use non-ASCII characters, Python requires explicit encoding and decoding of strings into Unicode. In IBM® SPSS® Modeler, Python scripts are assumed to be encoded in UTF-8, which is a standard Unicode encoding that supports non-ASCII characters.

4.

A text file stores data in the form of alphabets, digits and other special symbols by storing their ASCII values and are in a human readable format. ... Whereas, a binary file contains a sequence or a collection of bytes which are not in a human readable format.

5.

How to write unicode text to a text file in Python

```
unicode_text = u'ƶʒʓʔʕʗʘʙʚʛʜʝʞ'
encoded_unicode = unicode_text. encode("utf8")
a_file = open("textfile.txt", "wb")
a_file. write(encoded_unicode)
a_file = open("textfile.txt", "r") r reads contents of a file.
contents = a_file. read()
print(contents)
```

6.

Click the File > "Save As" menu. The "Save As" dialog box comes up. 3. Enter notepad_utf-16le as the new file name and select "Unicode" option in the Encoding field.

7.

Unicode is a superset of ASCII, and the numbers 0–127 have the same meaning in ASCII as they have in Unicode. Unicode is the universal character encoding used to process, store and facilitate the interchange of text data in any language while ASCII is used for the representation of text such as symbols, letters, digits, etc.

8.

Python 3.0 uses the concepts of text and (binary) data instead of Unicode strings and 8-bit strings. All text is Unicode; however encoded Unicode is represented as binary data. The type used to hold text is str , the type used to hold data is bytes . The biggest difference with the 2. In python 2. x, “print” is treated as a statement and python 3. x explicitly treats “print” as a function. This means we need to pass the items inside your print to the function parentheses in the standard way otherwise you will get a syntax error.