1. Python has a set of built-in methods that you can use on strings.

Note: All string methods returns new values. They do not change the original string.

Method ---------------Description

capitalize() -----------Converts the first character to upper case

casefold() -----------Converts string into lower case

center() ------------Returns a centered string

count() --------------Returns the number of times a specified value occurs in a string

encode() -----------Returns an encoded version of the string

endswith() -------------Returns true if the string ends with the specified value

expandtabs() ----------Sets the tab size of the string

find() ----------Searches the string for a specified value and returns the position of where it was found

format() ----------Formats specified values in a string

format\_map() ---------Formats specified values in a string

index() ----------Searches the string for a specified value and returns the position of where it was found

isalnum() -----------Returns True if all characters in the string are alphanumeric

isalpha() --------Returns True if all characters in the string are in the alphabet

isascii() -------Returns True if all characters in the string are ascii characters

isdecimal() -------Returns True if all characters in the string are decimals

isdigit() ----------Returns True if all characters in the string are digits

isidentifier() --------Returns True if the string is an identifier

islower() --------Returns True if all characters in the string are lower case

isnumeric() -------Returns True if all characters in the string are numeric

isprintable() ------Returns True if all characters in the string are printable

isspace() -----Returns True if all characters in the string are whitespaces

istitle() --------Returns True if the string follows the rules of a title

isupper() ------Returns True if all characters in the string are upper case

join() --------Converts the elements of an iterable into a string

ljust() ----------Returns a left justified version of the string

2. Python supports ASCII as a subset of Unicode. The default encoding of characters in Python is UTF-8 (Unicode Transformation Format - 8-bit).ASCII is really a 7-bit character set; it is mapped to 8-bit bytes by setting the high bit to zero. Thus, ASCII characters are unchanged in UTF-8. A non-zero high bit indicates that a character composed of multiple bytes.

u = 'akash'

print(ord(u[-1]))

3. s = 'Akash'

s += chr(128)+chr(169)

print(s)

4. Files opened in binary mode (appending 'b' to the mode argument) return contents as bytes objects without any decoding. In text mode (the default, or when 't' is appended to the mode argument), the contents of the file are returned as strings, the bytes having been first decoded using a platform-dependent encoding or using the specified encoding if given.

5. While reading file, we can specify the encoding such as(utf16, utf32), so the file will be decoded according to that encoding and results are returned to the user.

text = "this is sample text file for understanding encoding"

f = open('sample.txt','w', encoding='utf16')

f.write(text)

f.close()

f = open('sample.txt','r', encoding='utf16')

print(f.read())

this is sample text file for understanding encoding

6. Writing unicode to a text file adds a line or multiple lines of unicode text to the file. UTF-8 is the most common unicode character encoding.

Call str.encode(encoding) with encoding set to "utf8" to encode str. Call open(file, mode) to open a file with mode set to "wb" . "wb" writes to files in binary mode and preserves UTF-8 format. Call file.write(data) to write data to the file.

unicode\_text = u'ʑʒʓʔʕʗʘʙʚʛʜʝʞ'

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)

7. The first 128 Unicode code points represent the equivalent ASCII characters. They have the same meaning in both ASCII and unicode. Since UTF-8 encodes each of these characters with a single byte, any ASCII text is also a UTF-8 text. Unicode is a superset of ASCII.

8. In Python 3.X and later, there is only one string type that is unicode. So no effect on the code.