Q1. In Python 3.X, what are the names and functions of string object types?

* Ans : Python String Methods.
* string capitalize() in Python.
* Python String casefold() Method.
* Python String center() Method.
* Python String count() Method.
* Python Strings encode() method.
* Python String endswith() Method.
* expandtabs() method in Python.

Q2. How do the string forms in Python 3.X vary in terms of operations?

Ans :

|  |  |
| --- | --- |
| Method | Description |
| rstrip() | Returns a right trim version of the string |
| split() | Splits the string at the specified separator, and returns a list |
| splitlines() | Splits the string at line breaks and returns a list |
| startswith() | Returns true if the string starts with the specified value |

Q3. In 3.X, how do you put non-ASCII Unicode characters in a string?

Ans : 13

I'm programming in [Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29) and I'm obtaining information from a web page through the [urllib2](http://docs.python.org/library/urllib2.html) library. The problem is that that page can provide me with non-ASCII characters, like 'ñ', 'á', etc. In the very moment urllib2 gets this character, it provokes an exception, like this:

I need to handle those characters. I mean, I don't want to handle the exception but to continue the program. Is there any way to, for example (I don't know if this is something stupid), use another [codec](http://en.wikipedia.org/wiki/Codec) rather than the ASCII? Because I have to work with those characters, insert them in a database, etc.

Q4. In Python 3.X, what are the key differences between text-mode and binary-mode files?

Ans : The two file types may look the same on the surface, but they encode data differently. While both binary and text files contain data stored as a series of bits (binary values of 1s and 0s), the bits in text files represent characters, while the bits in binary files represent custom data.

Q5. How can you interpret a Unicode text file containing text encoded in a different encoding than your platform's default?

Ans : Handling character encodings in Python or any other language can at times seem painful. Places such as Stack Overflow have thousands of questions stemming from confusion over exceptions like UnicodeDecodeError and UnicodeEncodeError. This tutorial is designed to clear the Exception fog and illustrate that working with text and binary data in Python 3 can be a smooth experience. Python’s Unicode support is strong and robust, but it takes some time to master.

This tutorial is different because it’s not language-agnostic but instead deliberately Python-centric. You’ll still get a language-agnostic primer, but you’ll then dive into illustrations in Python, with text-heavy paragraphs kept to a minimum. You’ll see how to use concepts of character encodings in live Python code.

Q6. What is the best way to make a Unicode text file in a particular encoding format?

1. Ans : unicode\_text = u'ʑʒʓʔʕʗʘʙʚʛʜʝʞ'
2. encoded\_unicode = unicode\_text. encode("utf8")
3. a\_file = open("textfile.txt", "wb")
4. a\_file. write(encoded\_unicode)
5. a\_file = open("textfile.txt", "r") r reads contents of a file.
6. contents = a\_file. ...
7. print(contents)

Q7. What qualifies ASCII text as a form of Unicode text?

Ans : ASCII == UNICODE? For backward compatibility, the first 128 Unicode characters point to ASCII characters. And since UTF-8 encodes each of those characters using 1-byte. ASCII is essentially just UTF-8, or we can say that ASCII is a subset of Unicode

Q8. How much of an effect does the change in string types in Python 3.X have on your code?

Ans : Python 3.6 introduced a new way to format strings: f-Strings. It is faster than other string formatting methods in Python, and they allow us to evaluate Python expressions inside a string