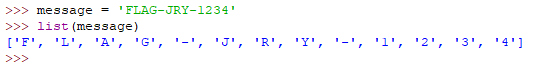
Python Hints for NCL Problems

In Python strings are immutable, meaning that you cannot change them at all. Before you complain, know that lists have many powerful methods and Python makes it easy to converts stings to lists, and back.

# Converting Strings to Lists

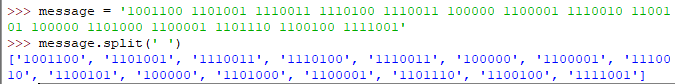
## Break it into characters

If you want to work on the characters one at a time, just convert it to a list.



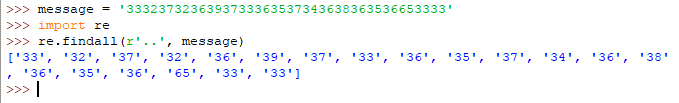
## When there is a separator

The split method works great for this.



## When you need to grab more than one char

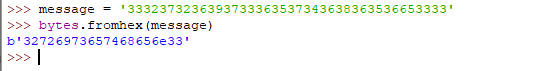
If you need to grab small blocks of characters, a regex works great. If you need two characters, the regex is ‘..’, one dot for each character. If you need three characters, you would use ‘…’ for the regex.



Note: The ‘r’ in r’..’ means raw. This tells Python to take the input exactly as it is so that you don’t have to worry about escaping things.

## If you know the message string is hex

If you know the string is in hex, you do not have to use the regular expression like we did above. Note that this method also converts the hex into integers instead of breaking it into two-character strings.



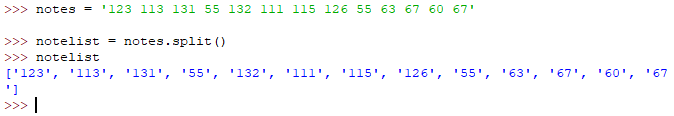
The message is the same as the re example above. Note that the beginning 33 was converted to its ASCII character, 3. The next two characters, 32, were converted into the ASCII character 2. The next two, 37, were converted into 7, and so on.

# For lists—list comprehension

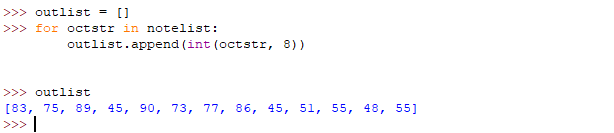
You iterate through lists so often that Python gives you a method to do it inside a list statement.

This message is a bunch of characters that are in base 8, which then can be looked up in an ASCII table. We are guessing that they are octal because no number is greater than 7, and it came from a musical scale which has 8 notes.

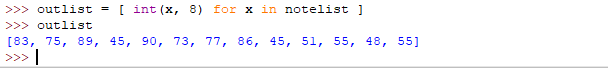
First, we convert the string data to a list. If you do not specify the delimiter, Python defaults to white space.



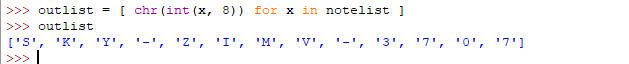
We want to change those strings (123, 113, etc.) into integers. We also want them in base 8. The regular for loop way to do this is:



In a list comprehension, it looks like this.



We want to convert the numbers into ASCII characters, so we do this.



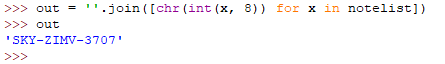
Note that chr() converts an integer into ASCII, and ord() converts ASCII characters into integers.

# Changing Lists back to strings

In the last example, the answer is a list, ['S', 'K', 'Y', '-', 'Z', 'I', 'M', 'V', '-', '3', '7', '0', '7']. It would be nice to have that as a string so we can paste it into the answer. The syntax for joining lists to make strings is a little strange (to me, anyway.)

‘separator’.join(list-to-join)

In our case, we don’t want a separator, so we start with ‘’.



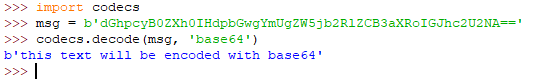
If, for some unknown reason, we wanted all the characters to be separated by commas, we could do this



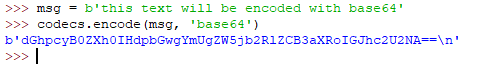
# Decoding base64

The base64 symbol set is a-z, A-Z, 0-9, +, and /. (The last two may differ sometimes.) If you see a message that contains those symbols, base64 is a good thing to try. A dead giveaway for base64 is if it ends in “=” or “==”. Base64 uses 6 bits per character while ASCII uses 8, so they often do not overlap exactly. To account for this base64 adds padding in the form of “=” at the end so that they do overlap. There are several ways to encode and decode base64 in Python. The method we will use requires the input to be in bytes, not a string.

dGhpcyB0ZXh0IHdpbGwgYmUgZW5jb2RlZCB3aXRoIGJhc2U2NA==



Going the other way,



Note: Python codecs defaults to strict error handling. I tried, using errors=’ignore’, to get it to ignore errors in the input data but failed. However, Linux base64 is laxer, and this usually works.

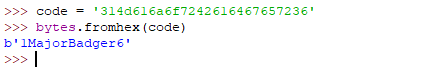
echo ‘some base64 string’ | base64 -d

# Examples

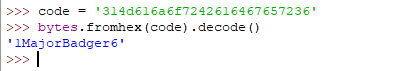
## Another case—base 16

If we know the base is 16, it’s easy. The hex is:

314d616a6f7242616467657236



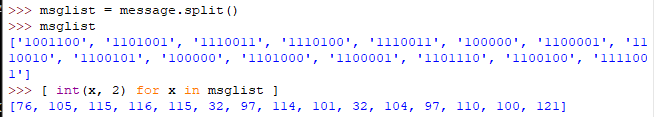
The little “b” at the beginning says that Python 3 has stored the data as bytes. If you want to make that go away, use the decode() method to change it to a pure string.

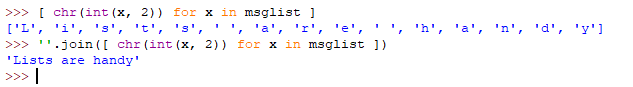


Note: Python3 stores strings in the UTF-8 format, which allows for characters that don’t fit into ASCII (languages other than English.) The type called bytes just stores the raw bytes.

## Here’s a simple one

The message is  
1001100 1101001 1110011 1110100 1110011 100000 1100001 1110010 1100101 100000 1101000 1100001 1101110 1100100 1111001





## Here’s one where you can use the Regular Expression, base 17

The message is:

362f6a635g5e5g625c6a6a72322f

It looks like hex, but if you try bytes.fromhex() it will throw errors. Note that there are “g”s in the message. Well, hex is base 16 and goes up to “f”, let’s try base 17 since the message includes characters up to “g”.

