Unit 3 Assignment

**ANLY:520-51 (Fall 2016)**

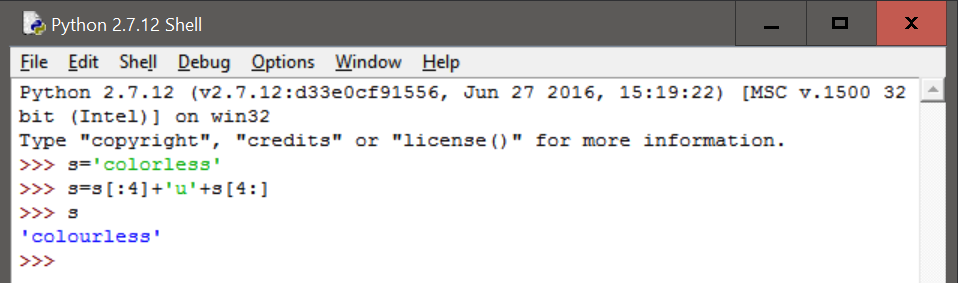
Dean D’souza

H.U. ID: 168424

# **Solutions:**

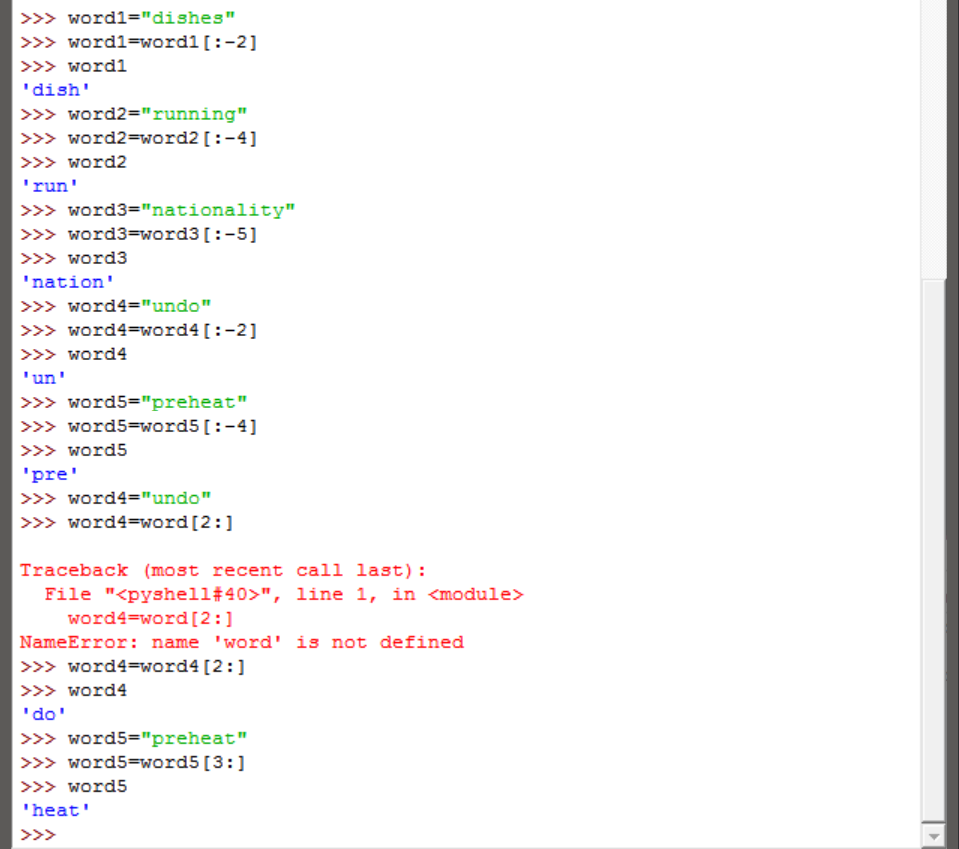
1. **Define a string s = 'colorless'. Write a Python statement that changes this to "colourless" using only the slice and concatenation operations.**

We perform the required operation through the following commands:

****

1. **We can use the slice notation to remove morphological endings on words. For example, 'dogs'[:-1] removes the last character of dogs, leaving dog. Use slice notation to remove the affixes from these words (we've inserted a hyphen to indicate the affix boundary, but omit this from your strings): dish-es, run-ning, nation-ality, un-do, pre-heat.**

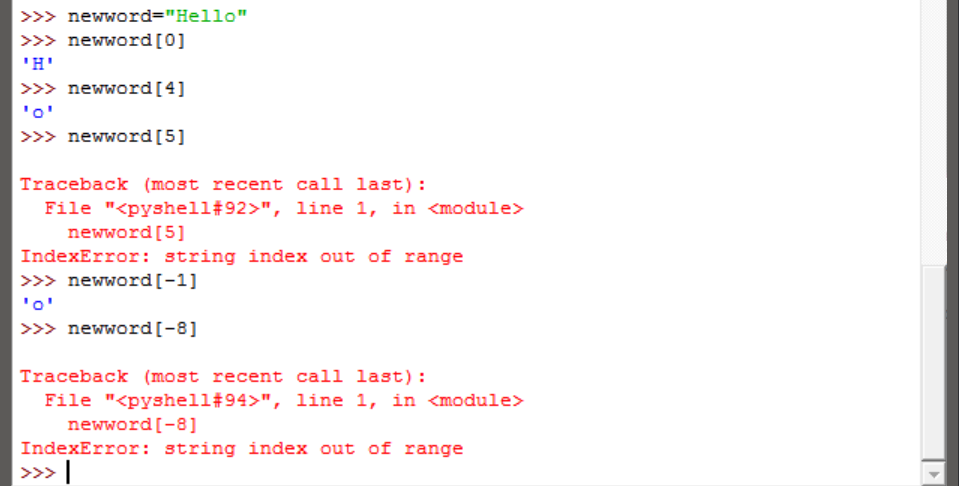
We remove the affixes as follows:

****

It should be noted that for words 4 and 5 we perform the operations using the same notation but as that would not make sense for the words we later correct the way the affix should be removed.

1. **We saw how we can generate an IndexError by indexing beyond the end of a string. Is it possible to construct an index that goes too far to the left, before the start of the string?**

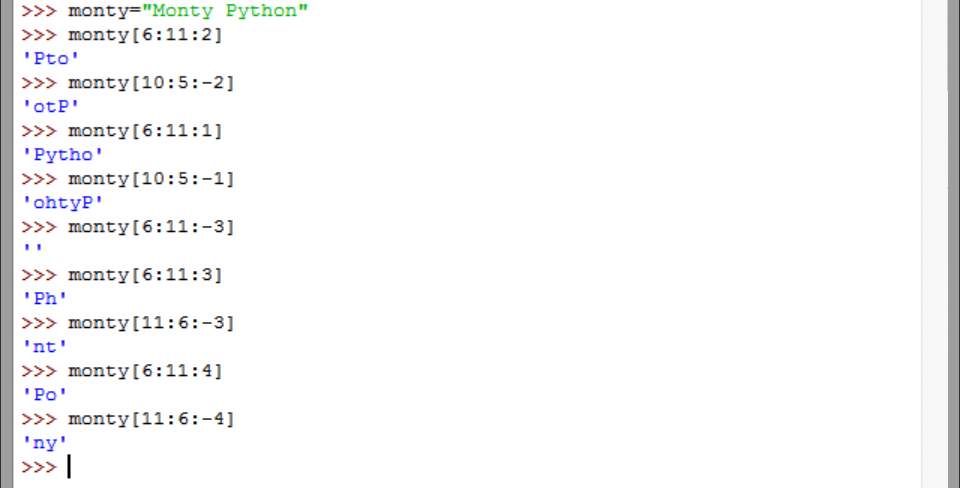
We perform the following commands to show how the indexing would work:

****

As we can see, logically, using a ‘-’ should go refer to an index more to the left, however, in python it just refers to the index position from the end of the string. Hence, as such we cannot generate an IndexError to the far left unless the indexing is wrong in a slice operation.

1. **We can specify a "step" size for the slice. The following returns every second character within the slice: monty[6:11:2]. It also works in the reverse direction: monty[10:5:-2] Try these for yourself, then experiment with different step values.**

We perform the commands as follows:

****

1. **What happens if you ask the interpreter to evaluate monty[::-1]? Explain why this is a reasonable result.**

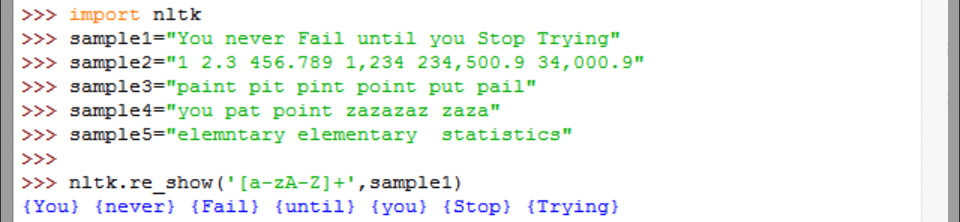
We perform the command as follows:

****

As we can see, the command causes the string to be displayed in reverse. As python uses the negative index number to refer to the index position from the end of the string this is a reasonable result and makes it easier to do string reversal operations as well.

1. **Describe the class of strings matched by the following regular expressions.** **Test your answers using nltk.re\_show().**
2. **[a-zA-Z]+**

The above regular expression should be able to match any combination of uppercase and lowercase letters provided there are one or more of such letters. We can see this as follows:

****

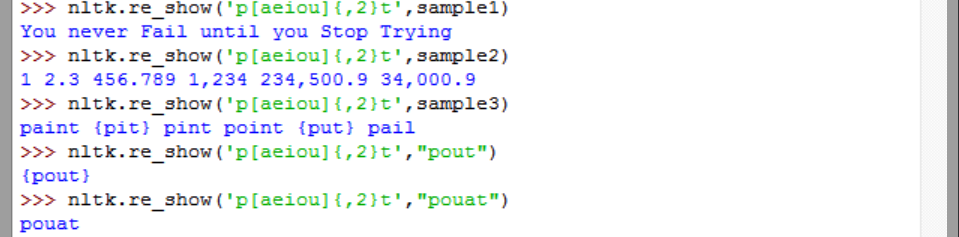
1. **[A-Z][a-z]\***

The above regular expression should match with an uppercase letter followed by zero or more lowercase letters. We can see this as follows:

****

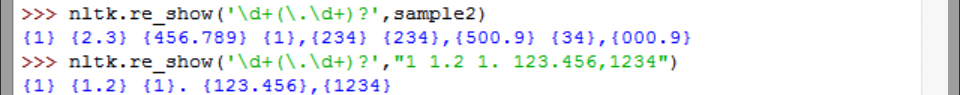
1. **p[aeiou]{,2}t**

The above regular expression matches with patterns which begin with a lowercase p followed by any lowercase vowel (aeiou) such that there are upto 2 of such vowels followed by a lowercase t. We perform the following commands to demonstrate:

****

1. **\d+(\.\d+)?**

The above regular expression matches patterns with one or more digits, which can optionally be followed by a period followed by digits. We perform the following commands to demonstrate:

****

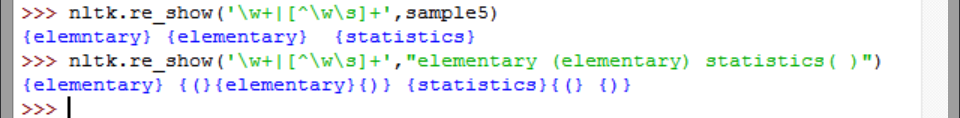
1. **([^aeiou][aeiou][^aeiou])\***

The above regular expression matches any pattern which starts with a lowercase non-vowel (not aeiou) followed by a lowercase vowel (aeiou) followed by a lowercase non-vowel (not aeiou) and such combinations can occur zero or more times. Thus this regular expression can match

****

1. **\w+|[^\w\s]+**

The above regular expression matches any pattern which has word characters or not word characters which are followed by a space. It should be noted however that the nltk.re\_show() functions seems to remove whitespaces and hence we have the commands as follows:

****