ENGL 516X:
Methods of Formal Linguistic Analysis

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

8 Feb 2018

Class outline

- Strings - continuation
- Regular Expressions (RegEx) - Basics
- Practice exercises

Summary of last class

- Looping through strings (while/for loop by index, for loop by character)
- Indexing forwards and backwards
- String slicing
- some builtin string methods

Built-in methods for strings

Some of them

- example="Some Example String"
- print(example.upper())
- print(example.lower())
- print(example.startswith("S"))
- print(example.endswith("S"))
- print(example.isdigit())
- print(example.find("e"))
- print(example.find("e",5))
- print(example.find("tri")

More on String methods

- Each string object can make use of a few built-in functions that are useful to do some string manipulations. Such functions that work on objects/variables are called methods.
- a built-in "function" called "dir()" shows all the available "methods" for a given object like a string, integer etc.

"in" operator and String comparisons

- "The word **in** is a boolean operator that takes two strings and returns True if the first appears as a substring in the second".
- e.g., 'seed' in 'banana' returns FALSE. "a" in "banana" returns TRUE.

"in" operator and String comparisons

- "The word **in** is a boolean operator that takes two strings and returns True if the first appears as a substring in the second".
- e.g., 'seed' in 'banana' returns FALSE. "a" in "banana" returns TRUE.
- We can also compare two strings using $==$, $<$ and $>$ as with numbers (Keep in mind - lower case and upper case characters are treated differently!)

Parsing Strings: Exercise

Write a function that takes in a string, and returns a string with punctuation stripped from original string.

Parsing Strings: Exercise

Write a function that takes in a string, and returns a string with punctuation stripped from original string.

```
import string
def remove_punctuation(s):
    s_without_punct = ""
    for letter in s:s
        if letter not in string.punctuation:
            s_without_punct += letter
    return s_without_punct

print(remove_punctuation('"Well, I never did!", said Alice.'))
print(remove_punctuation("Are you very, very, sure?"))
```

Parsing Strings: Exercise

- From the textbook, understand how .find() method for strings works.
- Now, write a program with a function **findit**, that takes three arguments, two strings and a number, and returns the remaining part of the string after the occurrence of the substring after the given location (real life descriptions can be messy like this!)

Parsing Strings: Exercise

- From the textbook, understand how .find() method for strings works.
- Now, write a program with a function **findit**, that takes three arguments, two strings and a number, and returns the remaining part of the string after the occurrence of the substring after the given location (real life descriptions can be messy like this!)
- Okay, here is how your program's interaction should look like:

```
enter a string: "mississippi"
enter a substring: "ss"
enter a number: 4
output is: ssippi
```

Parsing Strings: Exercise

- From the textbook, understand how .find() method for strings works.
- Now, write a program with a function **findit**, that takes three arguments, two strings and a number, and returns the remaining part of the string after the occurrence of the substring after the given location (real life descriptions can be messy like this!)
- Okay, here is how your program's interaction should look like:

```
enter a string: "mississippi"
enter a substring: "ss"
enter a number: 4
output is: ssippi
```

- ```
  def findit(str,substr,num):
      ind = str.find(substr,num)
      return str[ind:]

  print(findit("Mississippi","ss",4))
  #Do the interactive input part yourself!
  ```
  ```

Another similar question

Exercise 5 of Chapter 6

The question: Take the following Python code that stores a string:
str = 'X-DSPAM-Confidence: 0.8475'
Use find and string slicing to extract the portion of the string after the colon character and then use the float function to convert the extracted string into a floating point number.

Another similar question

Exercise 5 of Chapter 6

The question: Take the following Python code that stores a string:
str = 'X-DSPAM-Confidence: 0.8475'
Use find and string slicing to extract the portion of the string after the colon character and then use the float function to convert the extracted string into a floating point number.

One solution:

```
str = 'X-DSPAM-Confidence:  0.8475'
index = str.find(":")
required = str[index+3:]
#because there are two spaces before the number started.
print(float(required))
```

(Note: in a real program, you need to watch out for possible things that can go wrong and have exception handling)

strip() method for Strings

Taking the previous problem again, I can get what I want in a different way.

```
str = 'X-DSPAM-Confidence:  0.8475'
index = str.find(":")
required = str[index+1:].strip()
print(float(required))
```

strip() method strips off the white spaces, tabs etc at the beginning or end of a string. There are lstrip() and rstrip() methods as well.

replace() method for Strings

replace() is used to replace part of a string with some other value. See this example.

```
str = 'X-DSPAM-Confidence:  0.8475'
newstr = str.replace("X","Y")
print(newstr)
Y-DSPAM-Confidence:  0.8475
```

String methods - practice1

What is this code doing?

```
def secretFunction(str1,str2):
  str1_lower = str1.lower()
  str2_lower = str2.lower()
  if str1_lower == str2_lower:
     return True
  else:
     return False
print(secretFunction("LaTex","latex"))
print(secretFunction("Nature","Nurture"))
```

String methods - practice 2

What is this code doing?

```
def secretFunction2(str1,str2):
  result = ""
  result = str2[0:2] + str1[2:] + " " + str1[0:2] + str2[2:]
  return result
print(secretFunction2("suntan","sinner"))
print(secretFunction2("whats","that"))
```

Regular Expressions

Regular Expressions

- Regular expressions are used to do do pattern based information extraction from data.
- They have their own syntax for doing pattern matching in different ways.
- They are very useful to process text and manipulate it.
- Regular expressions in python are in a module called "re" and you can use them in your code once you add a "import re" statement in your program/console.
- They can simplify a lot of your tasks, but they themselves can be very complicated.
- pythex.org - is what I will use today to explain the syntax. We will use import re in our code next week.

RegEx syntax

1. ^ matches the beginning of a line. For example,
   - a pattern ^Th matches all lines in a text file that start with Th
2. $ matches the end of a line. For example,
   - a pattern Th$ matches all lines in a text file that end with Th
3. \s matches a white space character
4. \S matches a non-white space character.

RegEx syntax

1. . matches any character
2. * -applies to the immediately preceding character and indicates to match zero or more of the preceding character(s).
    - for example, te* matches all locations where there is a t, te, tt, tete etc.
3. + - applies to the immediately preceding character and indicates to match one or more of the preceding character(s).
    - for example, te+ matches all locations where there is a te, tete, tetete etc.

RegEx syntax - continued

The power of square brackets

1. [aeiou]- matches a single character as long as the character is in this set.
2. You can also specify ranges in square brackets. For example, [a-z0-9] matches all characters in lower case or a single digit.
3. When the first character after the square brackets is a caret (^), it works like a "not" keyword. So, [^a-z0-9] matches all characters that are not lower cased letters, and not numbers.

Escape Character

What do you do if you want to match a ? or a . which also carry a meaning in regex?

Escape Character

What do you do if you want to match a ? or a . which also carry a meaning in regex?
We "escape" them to tell regex module that these are real characters and not regex syntax. This is done using a \ character.

So, st\. is a pattern that searches for all occurences of "st." in a string.

Regex practice on http://pythex.org

Go to APLING program homepage (apling.engl.iastate.edu) and copy the welcome message there into pythex test string area. Now, try to write regex patterns to get the following:

1. All occurences of the word "is" (Not th**is**, lingu**is**tics, etc. Only "is")
2. All occurences of the letter e, irrespective of the case.
3. All occurences of "es" where it occurs in the middle of the word (i.e., es should not be followed by a space, comma, fullstop etc)

Next Class

- Topics: re module in python
- Readings: Chapter 11 in the text book.
- Optional exercises for the week: Uploaded on Canvas