KJC4 4CREDITS

NARRATIVE FOR HOMEWORK 13

We start by importing python’s regular expression module. We then open the Dracula file for reading. Next we read it in using the read method and save it in a variable ”drac”

Problem 1:

First we compile regular expression pattern for the dates into a regular expression object using re.compile and save it in a variable “dates”. We then use re.findall to find all occurrences of that pattern in our Dracula text “drac” and save it in a variable “found\_dates”. We then print the length of found\_dates and find that there are 165 dates.

Problem 2:

From the collections module, we import the Counter object. We use the Counter object to find out how many times each date tuple occurred in “found\_dates” and saved the output in a variable “ counted”. We create an empty list “repeateddates”. We use a for loop to loop through each date,count pair in counted.items(). We then use an if statement within the for loop with the condition that if count is >1, print the date and the count, next append the count to our list “repeateddates”. Finally we print the length of “repeateddates”. This gives us how many dates are repeated.

Problem 3:

We create an empty list “mostcommon”. We then use a for loop to loop through each element(count) in counted.most\_common(). We use an if statement within the for loop such that if the second element in each count pair (count[1]) is greater than 1, we append the count pair to our list “mostcommon”. Finally we print the first element of “ mostcommon” which is the the most common date.

Problem 4:

First we compile the regular expression pattern for the lines that start and end with underscores( each new line has the \n character) into a regular expression object using re.compile and save it in a variable “underscores”. We then use re.findall to find all the occurrences of that pattern in our Dracula text “drac” and save it in a variable ”all\_unders”. We print the length of all\_unders. We then loop through each element in “all\_unders” to print each element out. This gives us the lines that begin and end with underscores.

Problem 5:

First we compile the regular expression pattern for the dates into a regular expression object using re.compile and save it in a variable “dates\_2”. Next we use re.findall to find all the occurrences of that pattern in our Dracula text “drac” and save it in a variable ”found\_dates\_2”. We create an empty dictionary “counts”. We then use a for loop to loop through each element in found\_dates\_2 and an if statement within the for loop with the condition that if the month part of the date which starts at the 4th character (date[3:] ) isn’t already in “counts” , assign 1 as it’s count value. Else, if it is already in counts, add 1 to its current count value. This helps us create a dictionary of how many dates within each month were mentioned in the Dracula text. Lastly, we print out “counts”

Problem 6:

First we compile the regular expression pattern into a regular expression object using re.compile and save it in a variable “names”. Next we use re.findall to find all the occurrences of that pattern in our Dracula text “drac” and save it in a variable ”found\_names”. We use the Counter object to find out how many times each name occurred in “found\_names” and store the output in “counted\_2”. We then use list() to convert counted\_2 into a list and store it in a variable ”name\_list”. We open a new file for writing. We then loop through each element in name\_list, and write each element to the new file on individual lines by using the new line character (\n) to join each element.

Using this pattern to match names works because most of the characters are referred to by both their first and last names or by their titles followed by their last names(Dr. Seward) in the Dracula text both of which usually have 1st characters capitalized. This pattern however , follows the form in which many paragraphs and sentences are started in the Dracula text so it will pick up many strings that are not character names too. The pattern used will also pick up strings at the end of a sentence and at the beginning of another sentence such as “Church. Suddenly” which isn’t a character name as well as some strings like “Mile End” , “Black Sea”