IDS 703
HW Assignment I
Submitted by Jiahui (Heather) Qiu
September 14, 2022

Question 1 - Write a regular expression that matches people's full names. Here are some example names:

Match	Not Match
Quan Hongchan	Cher
Philip Seymour Hoffman	not a name
Dr. Nicki Washington	happy feet
Joseph Gordon-Levitt	The end
Ken Griffey, Jr.	
John von Neumann	

Describe in what situations your solution will fail.

pattern = r"([A-Z][a-z]+)(.*)([A-Z][a-z]+)(\.)?"

For readability, I broke the regex into four capturing groups. The first and the third capturing groups ensure that the full name contains two sets of one capitalized letter followed by one or more letters, each at the start or end of text. The second capturing group matches on any single character zero or more times between the pair of letter-cased expressions noted in the first and the third group. The fourth group factors in the period at the end of name suffixes like 'Jr.' To do so, I made the period optional at the end of my pattern.

With these rules, my solution captures all the names in the match column and does not match the four scenarios in question. However, my pattern will also capture false-positive patterns that contain at least two words in letter case, each at the start or end of the text. Below are examples where my answer will return a match, but they are not valid names:

- We have a student in class whose name is Heather.
- The Umbrella Academy
- Not a Name

Additionally, my solution fails to identify names in all lower cases. For example, in Chinese pinyin spelling, my name would be represented as "jiahui qiu" with no capitalization in the first or the last names. Though this is a valid full name, my pattern will not capture my name because it expects the spelling to be in letter case. It will also fail to identify full names outside of the English language.

Question 2 - Develop a regular expression that matches all gerunds within a text. For example, from the text, “harry loves to sing while showering.” Your expression should match only showering.

You may assume the following or state any other assumptions that you would like to make:

- the input contains only lowercase letters, punctuation, and whitespace
- any non-letter character is not part of a word
- there will be no contractions, hyphenated words, etc.

Generate your own test cases and write a script testing your expression, in the style of test_name_matching.py. Describe in what situations your solution will fail.

pattern = r"\w\w+ing" (Note: I adjusted the test_gerund_matching function so that it searches the pattern in the test string rather than looking for a full match.)

According to the Grammarly article[1], “A gerund is a verb that is acting as a noun.” In other words, gerunds are in the format of verb plus -ing. In addition, I assumed that a gerund includes at least two characters before the suffix -ing. Therefore, my regex pattern matches words with two or more letters before ending in -ing. It correctly identified gerunds in examples 1 through 4 below:

index	Test Cases	What are the true gerunds my pattern catches?	Will my pattern return false positives?
1	writing in english is difficult	writing	No
2	seeing is believing	seeing, believing	No
3	ling loves to sing	N/A	No
4	Ana likes having a cup of coffee every morning while reading	having, reading	Yes, ‘morning’ is a noun here.
5	that is an interesting idea	N/A	Yes, ‘interesting’ is an adjective here.
6	she bought a painting from the art fair	N/A	Yes, ‘painting’ is noun here.
7	someone is swimming in the pool	N/A	Yes, ‘swimming’ is a verb here.
8	_0ing is nonsense	N/A	Yes, ‘_0ing’ is not a word.

However, the pattern comes with many false positives, shown in the last column for examples 4 through 8 above. It captured non-gerund words such as morning, interesting, and painting. In addition, my expression fails to identify just gerunds in a sentence. A case in point is example number 4, where the solution captured all three words ending in -ing when only “having” and “reading” are gerunds. Last, my regex expression also captures non-words, such as “_0ing” in example 8, because it matches all alphanumeric characters, including an underscore.

[1] https://www.grammarly.com/blog/gerund/