Assignment 1: Match

Overview

In this assignment, we begin to construct a natural language query system.[1] In the end, we will create a program that will participate in dialogs like the following.

```
Your query? What movies were made in 1974?
Amarcord
Chinatown

Your query? Who acted in The Dresser?
Albert Finney
Tom Courtenay
Edward Fox
Zena Walker

Your query? What movies were directed by Federico Fellini?
Amarcord

Your query? Bye

So long!
```

In the example dialogue above, **orange text** is user input and **blue text** is stated by the system.

There are several parts to the program, and it is convenient (essential, even!) to consider them separately. In this assignment, we consider the textual patterns and how we can match them to real user input. In future assignments, we will read data from sources, look at ways to create functions to attach to the patterns that we recognize, and finally tie it all together into a complete system. While the examples used in this assignment refer to questions about movies, your final system will utilize data from the CIA World Factbook, enabling questions like "Which country is ranked number 10 for population?".

[1] The original version of this assignment is taken from the textbook Concrete Abstractions: An Introduction to Computer Science Using Scheme, by Max Hailperin, Barbara Kaiser, and Karl Knight, Copyright (c) 1998 by the authors. Full text is available for free here. The assignment evolved at Pomona College through several instructors' offerings, with changes by Nathan Shelly and Sara Sood, Northwestern University.

More Details

Consider the questions below. There is a fixed pattern except for the year.

What movies were made in 1974?
What movies were made in 1946?
What movies were made in 2001?

Or consider the questions

Who acted in the movie Jaws?
Who directed the movie Citizen Kane?

These patterns are a little more complicated. The differences are that there are two parts in each question that can differ, "**acted in**" and "**directed**" as well as "**Jaws**" and "**Citizen Kane**." Also, the thing that differs *may* be more than one word.

Let us call a question like the ones above a **source**. A **pattern** is a string like one of these:

what movies were made in _
who % the movie %

The idea is that we match words in the **source** with words in the **pattern**. The symbol '_' (an underscore) can match any single word, and the symbol '%' (a percent sign) can match a sequence of zero or more words.

Your job

The goal of this assignment is to write a function called *match*, that takes a **pattern** and a **source** as arguments (in that order, each as lists of strings) and returns either:

- None (the NoneType) if the **source** does not match the **pattern** OR
- a list of substitutions (strings from the source that correspond to each _ and % in the **pattern**) if the **source** does match the **pattern**.

Note: The sequence of words that match a % in a pattern must all be condensed into one element of the returned list. There should be one space between each word in the sequence, but not at the beginning or end of the sequence.

For example,

```
match(['what', 'movies', 'were', 'made', 'in', '_'],
        ['what', 'movies', 'were', 'made', 'in', '1974'])
should return ['1974']
```

```
match(['who', '%', 'the', 'movie', '%'],
        ['who', 'acted', 'in', 'the', 'movie', 'jaws'])
should return ['acted in', 'jaws']

match(['who', '%', 'the', 'movie', '%'],
        ['what', 'movies', 'were', 'made', 'in', '1974'])
should return None.
```

For simplicity, we do not allow our pattern to contain the sequence ['%', '%'] or ['%', '_'] because that would make the *match* function much more complicated.

You may assume that both the **pattern** and **source** lists of words are all lowercase.

Your job is to come up with your own design for the match function and implement it. We strongly recommend writing your logic out in English first before moving to python.

The assert statements provided in match.py show examples of correct results for the match function. If implemented correctly, your code should pass ALL of these asserts.

Write your match function in match.py, and upload it to canvas.