Homework 6: Regular Expressions

Due: 17 October 2025 at 11:59 pm ET

The length of this homework is inversely proportional to your knowledge in writing regular expressions, both for finding matches and for doing substitutions.

A good resource to use is https://regex101.com/ for checking your expressions.

Background

Please refresh your memory of regular expressions using the class notes. You may also find the Python documentation on regular expressions useful.

A few helpful reminders:

Testing for Patterns

When you use `re.search` to find a regular expression match, it returns a `Match` object if the pattern exists in the string (we will see more about objects later in the semester). If *there is no match*, then `re.search` (and `re.match` and `re.findall`) will return None, which you can test for as shown below:

```
p = re.compile('pattern')
if (p.search(s)):
   # This branch will execute if the pattern is found
else:
   # This branch will execute if the pattern is *not* found
```

`re.match` and `re.search` are different and there is documentation on this too.

Substituting with functions

A common use of `re.sub` is to substitute one string for another (remember that you can use the *groups* that you match in a pattern as part of your string substitution):

```
s = "loooool"
p = re.compile('(l)o+(l)')
p.sub(r'\1o\2', s) # replace "loooool" with "lol"
# note that the digit after the '\' corresponds to the group number
```

You can also call a method instead of providing a replacement string. This method will be called with the `Match object` corresponding to the matched string, and should return a string:

```
def replFun(m) :
   return m.group(2).upper()
s = "loooool"
p = re.compile('(l)(o+)(l)')
m = p.search(s)
p.sub(r'\1'+replFun(m) +r'\3', s) #replace "loooool" with "lOOOOOl"
```

The `re.MatchObject.group()` method returns the complete matched subgroup by default or a tuple of matched subgroups depending on the number of arguments. Remember that `(`, `)`, `-` and `.` are special characters for regular expressions. To search for those characters, you need to precede them with a backslash: `\(` `\)`, `\-`, `\.`.

Instructions

0) Download the folder from Brightspace

The folder should contain three files:

1. `hw5.py`, the file in which you will fill in the functions for the problems. This also contains test code you can use to test your solutions.
2. This `HW6_handout.pdf` and `README.md` files.

Problem 1: Regular expression matches

Congratulations! You've been hired by Vought International to help clean up a messy database of supe-related emails. Some of these were also synced from Godolkin University and government systems, and the Federal Bureau of Superhuman Affairs needs the bad ones filtered out. Your job: decide which emails are valid.

Fill in the function problem1. This function should return the string valid if the input string is a valid email address and invalid if not. A valid email in this universe is defined as follows:

The email must begin with a name composed of upper or lower case letters, containing at least 1, but no more than 10 letters, followed by a . (period).

After the . the email must have the ID number of the supe/student/agent, from 100 up to 799.

The email may have any number of letters only immediately following the ID number, but anything else (digits, symbols, underscores, etc.) between the ID number and the @ symbol is invalid.

The email must have the @ symbol followed by one of these domains: vought.com, godolkin.edu, or fbsa.gov.

There must be nothing following the final .com, .edu, or .gov.

Correct Examples:

```
starlight.123@vought.com
hughie.250supe@fbsa.gov
marie.100@godolkin.edu
queenmaeve.765@vought.com
```

Incorrect Examples:

```
a-train.123@vought.com (breaks rule 1: name has a hyphen)
homelander.800@vought.com (breaks rule 2: number out of range)
blacknoir.567*abc@godolkin.edu (breaks rule 3: non-letters after ID)
noir.123@vought.co (breaks rule 4: wrong domain)
starlight.144@vought.comasdf (breaks rule 5: trailing junk)
```

ANY other format should not count as a valid email. Spaces before or after an otherwise valid email is considered invalid.

Because we are looking for the entire string to be an email, you can either use ^ and $ to force a match to be at the beginning and end of a string, or you can use `fullmatch` instead of `match` or `search`.

Problem 2: Groups

You want to explore other jobs and now you've been hired as a junior librarian (it doesn't pay well, but hey, you get to sit around books all day). The library has a unique problem: all the books are shelved by sentences describing the author and the book. Unfortunately, they're not in any particular order. Your task is to extract the author's name and the title of the book from these sentences, so the books can finally be put back in the right place.

A sentence would have the format *Author wrote Book Name* or *Author wrote books* and will have the following conditions:

1. An author's name can be one word, or two words. The word or words in a name must start with a capital letter. The author names contain letters only, no special characters or numbers.
2. The verb is "wrote".
3. If the book has a name, it can be one, two or three words/numbers. Each word starts with a capital letter or number. If the book does not have a name, "wrote" will be followed by

"books". The book names contain letters and/or numbers only and no other special characters.

4. If the above conditions are both not met, there is no match. You will then return a tuple of strings ("noauthor", "noname")

Example:

```
JK Rowling wrote the Harry Potter Series
```

Although there is the phrase “JK Rowling wrote” it has "the" following the word "wrote" and its first letter is not capital. Therefore, this will not meet the conditions.

Fill in the function `problem2`. This function should search an input string for the author and book then return a tuple of them.

```
George Orwell wrote 1984
```

you should return:

```
('George Orwell','1984')
```

If you pass in:

```
In the 1930s, a Mystery writer wrote Mary Westmacotts. Later it was found that Agatha Christie wrote The Westmacott Novels
```

you should return:

```
('Agatha Christie', 'The Westmacott Novels')
```

This is because "Mystery writer" did not have a capitalized first letter in the word ("writer") preceding "wrote"

If you pass in:

```
Roxette wrote books
```

you should return:

```
('Roxette', 'books')
```

book does not have a name - returned value is just one word and books

If you pass in:

`John Cena wrote wrestling books` or `John cena wrote Wrestling books` or `John cena wrote wrestling books`

you should return:

```
('noauthor', 'noname')
```

This is because all the conditions need to be met for it to be valid.

Be careful not to return (leading or trailing) extra spaces in the return value. You may need to do a little bit of extra processing of the string captured by your group to ensure this. You will receive partial credit for having spaces. Please remove extra spaces for full credit.

Problem 3: Substitution

You ditched library job and now process Godolkin University campus radio transcripts. A bunch of lines mislabel supes with “Boy/Girl” or “boy/girl” when they should read “Man/Woman” or “man/woman.” Your task is to fix the label when the word immediately before it looks like a proper name (starts with a capital letter).

Rules

Replace Boy/Girl → Man/Woman and boy/girl → man/woman only when the preceding token is a name (starts with a capital letter). The replacement should match the case of the first letter of the original word (i.e., preserve capitalization).

If no match is found, return "nomatch".

Here are some examples:

```
Crimson Girl, report to Brink Hall!
```

Should return

```
Crimson Woman, report to Brink Hall!
```

```
There is a girl trapped in Dorm C Tech Boy
```

Should return

```
There is a girl trapped in Dorm C Tech Man
```

First "girl" stays because it is not preceded by a capitalized name; "Tech Boy" becomes "Tech Man".

Be careful not to return extra spaces in the final output. You may need to do a little bit of extra processing of the string captured by your group to ensure this. You will receive partial credit for having unwanted spaces. Please remove extra spaces for full credit.

Testing Your Code

To test your code, run the `hw6.py` file. These tests are not exhaustive, passing them does not guarantee full credit on the homework.

What to Submit

Please submit `hw6.py` with all the functions filled in.

Submitting your code

Please add, commit and push the latest version of your code, as you did in the previous Homeworks. Do not make any modifications after submission to avoid a late penalty.