CPS 211 Python Worksheet 8
(Regular Expression)

1. For a given string verify that it is a floating point number. In this task, a valid float number must satisfy all of the following requirements: Number can start with +, - or . symbol. For example: “+4.50”, “-1.0”, “.5”, “-.7”, “+.4” are valid but and “-+4.5” is wrong.
2. CSS colors are defined using a hexadecimal (HEX) notation for the combination of Red, Green, and Blue color values (RGB). Create a function isValidColor using RE. For specifications of HEX Color Code

   ■ It must start with a '#' symbol.

   ■ It can have 3 or 6 digits.

   ■ Each digit is in the range of 0 to F . (1,2,3,4,5,6,7,8,9,0,A,B,C,D,E and F).

   ■ A-F letters can be lower case. (a,b,c,d,e and f are also valid digits).

3. You are given a string, and you have to validate whether it's a valid Roman numeral. If it is valid, print True. Otherwise, print False. Try to create a regular expression for a valid Roman numeral. As Input, A single line of input containing a string of Roman characters In Output t a single line containing True or False according to the instructions above.
4. A valid email address meets the following criteria: It's composed of a username, domain name, and extension assembled in this format: username@domain.extension, The username starts with an English alphabetical character, and any subsequent characters consist of one or more of the following: alphanumeric characters, ‘-’ ‘.’ and ‘_’. The domain and extension contain only English alphabetical characters. The extension is 1,2 , or 3 characters in length. For given n pairs of names and email addresses as input, print each name and email address pair having a valid email address on a new line.
5. The check_web_address function checks if the text passed qualifies as a top-level web address, meaning that it contains alphanumeric characters (which includes letters, numbers, and underscores), as well as periods, dashes, and a plus sign, followed by a period and a character-only top-level domain such as ".com", ".info", ".edu", etc. Demonstrate the regular expression to do that, using escape characters, wildcards, repetition qualifiers, beginning and end-of-line characters, and character classes. Use Regular Expression only.

   As

- `print(check_web_address("gmail.com")) # True`
- `print(check_web_address("www@google")) # False`
- `print(check_web_address("www.Coursera.org")) # True`
- `print(check_web_address("web-address.com/homepage")) # False`
- `print(check_web_address("My_Favorite-Blog.US")) # True`

6. The check_time function checks for the time format of a 12-hour clock, as follows: the hour is between 1 and 12, with no leading zero, followed by a colon, then minutes between 00 and 59, then an optional space, and then AM or PM, in upper or lower case. Create the regular expression to do that. Use Regular Expression only. As
   - `print(check_time("12:45pm")) # True`
   - `print(check_time("9:59 AM")) # True`
   - `print(check_time("6:60am")) # False`
   - `print(check_time("five o'clock")) # False`
7. The contains_acronym function checks the text for the presence of 2 or more characters or digits surrounded by parentheses, with at least the first character in uppercase (if it's a letter), returning True if the condition is met, or False otherwise. For example, "Instant messaging (IM) is a set of communication technologies used for text-based communication" should return True since (IM) satisfies the match conditions." Use Regular Expression only. As

- `print(contains_acronym("Instant messaging (IM) is a set of communication technologies used for text-based communication")) # True`
- `print(contains_acronym("American Standard Code for Information Interchange (ASCII) is a character encoding standard for electronic communication")) # True`
- `print(contains_acronym("Please do NOT enter without permission!")) # False`
- `print(contains_acronym("PostScript is a fourth-generation programming language (4GL)")) # True`
- `print(contains_acronym("Have fun using a self-contained underwater breathing apparatus (Scuba)!")) # True`

8. Create a check_zip_code `function` to check if the text passed includes a possible U.S. zip code, formatted as follows: exactly 5 digits, and sometimes, but not always, followed by a dash with 4 more digits. The zip code needs to be preceded by at least one space, and cannot be at the start of the text. Use Regular Expression only.
   - `print(check_zip_code("The zip codes for New York are 10001 thru 11104.")) # True`
   - `print(check_zip_code("90210 is a TV show")) # False`

- `print(check_zip_code("Their address is: 123 Main Street, Anytown, AZ 85258-0001.")) # True`
- `print(check_zip_code("The Parliament of Canada is at 111 Wellington St, Ottawa, ON K1A0A9.")) # False`

9. We're working with a CSV file, which contains employee information. Each record has a name field, followed by a phone number field, and a role field. The phone number field contains U.S. phone numbers, and needs to be modified to the international format, with "+1-" in front of the phone number. Create regular expression, using groups, to use the transform_record function to do that.
    - `print(transform_record("Sabrina Green,802-867-5309,System Administrator"))# Sabrina Green,+1-802-867-5309,System Administrator`
    - `print(transform_record("Eli Jones,684-3481127,IT specialist"))`
    - `# Eli Jones,+1-684-3481127,IT specialist`
    - `print(transform_record("Melody Daniels,846-687-7436,Programmer")) # Melody Daniels,+1-846-687-7436,Programmer`
    - `print(transform_record("Charlie Rivera,698-746-3357,Web Developer")) # Charlie Rivera,+1-698-746-3357,Web Developer`
10. The multi_vowel_words function returns all words with 3 or more consecutive vowels (a, e, i, o, u). Find regular expression to do that.
    - `print(multi_vowel_words("Life is beautiful"))# ['beautiful']`
    - `print(multi_vowel_words("Obviously, the queen is courageous and gracious.")) # ['Obviously', 'queen', 'courageous', 'gracious']`
    - `print(multi_vowel_words("The rambunctious children had to sit quietly and await their delicious dinner."))`
    - `# ['rambunctious', 'quietly', 'delicious']`
    - `print(multi_vowel_words("The order of a data queue is First In First Out (FIFO)")) # ['queue']`
    - `print(multi_vowel_words("Hello world!"))# []`
11. The transform_comments function converts comments in a Python script into those usable by a C compiler. This means looking for text that begins with a hash mark (#) and replacing it with double slashes (//), which is the C single-line comment indicator. For the purpose of this exercise, we'll ignore the possibility of a hash mark embedded inside of a Python command, and assume that it's only used to indicate a comment. We also want to treat repetitive hash marks (##), (###), etc., as a single comment indicator, to be replaced with just (//) and not (#//) or (//#). Creat this function using RE:
    - `print(transform_comments("### Start of program"))`
    - `# Should be "// Start of program"`

- `print(transform_comments("  number = 0   ## Initialize the variable"))# Should be "  number = 0   // Initialize the variable"`
- `print(transform_comments("  number += 1   # Increment the variable"))# Should be "  number += 1   // Increment the variable"`
- `print(transform_comments("  return(number)"))`
- `# Should be "  return(number)"`

12. The convert_phone_number function checks for a U.S. phone number format: XXX-XXX-XXXX (3 digits followed by a dash, 3 more digits followed by a dash, and 4 digits), and converts it to a more formal format that looks like this: (XXX) XXX-XXXX. Fill in the regular expression to complete this function.
    - `print(convert_phone_number("My number is 212-345-9999.")) # My number is (212) 345-9999.`
    - `print(convert_phone_number("Please call 888-555-1234")) # Please call (888) 555-1234`
    - `print(convert_phone_number("123-123-12345")) # 123-123-12345`
    - `print(convert_phone_number("Phone number of Buckingham Palace is +44 303 123 7300")) # Phone number of Buckingham Palace is +44 303 123 7300`