Practice Quiz: Advanced Regular Expressions

TOTAL POINTS 5

1. 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. Fill in the regular expression, using groups, to use the transform_record function to do that.

1 / 1 point

```
import re
def transform_record(record):
  new_record = re.sub(r"([\w\s]*),([\d-]*),([\w\s]*)",r"\1,+1-\2,\3",record)
  return new_record

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
```

Run

Reset

2. The multi_vowel_words function returns all words with 3 or more consecutive vowels (a, e, i, o, u). Fill in the regular expression to do that.

1 / 1 point

```
import re
def multi_vowel_words(text):
  pattern = "[\w?]*[aeiouAEIOU]{3,}[\w?]*"
  result = re.findall(pattern, text)
  return result

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 thei
# ['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!"))
# []
```

Run

Reset

```
['beautiful']
['Obviously', 'queen', 'courageous', 'gracious']
['rambunctious', 'quietly', 'delicious']
['queue']
```

3. When capturing regex groups, what datatype does the groups method return? 1 / 1 point

- A string
- A tuple
- A list
- A float

✓ **Correct**

Nice job! Because a tupleis returned, we can access each index individually.

4. 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 (//#). Fill in the parameters of the substitution method to complete this function: 1 / 1 point

```
1  import re
2  def transform_comments(line_of_code):
3      result = re.sub(r"#+","//",line_of_code)
```

4. 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 (//#). Fill in the parameters of the substitution method to complete this function:

1 / 1 point

```
import re
def transform_comments(line_of_code):
  result = re.sub(r"#+","//",line_of_code)
  return result

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)"
```

Run

Reset

```
// Start of program
  number = 0   // Initialize the variable
  number += 1   // Increment the variable
  return(number)
```

5. 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.

1 / 1 point

```
import re
def convert_phone_number(phone):
  result = re.sub(r'(\d{3})-(\d{3})-(\d{4})\b', r'(\1) \2-\3', phone)
  return result

print(convert_phone_number("My number is 212-345-9999.")) # My number is (212) 345-9
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"))
```

Run

Reset

```
My number is (212) 345-9999.
Please call (888) 555-1234
123-123-12345
Phone number of Buckingham Palace is +44 303 123 7300
```

Correct

Well done! You've captured the right groups to identify what we're looking for, and nothing else!