Tutorial 3 Solution

TODO #1: we saw "None" in result variable. How can we replace None with "error"?

```
result.map(lambda x: (x[0], ('error' if x[1][0] is None else x[1][0],
'error' if x[1][1] is None else x[1][1]))).collect()
```

TODO 2: Write a module to read the "wordCountEx.txt" file, split the word by space and do a word count and sort the result descending.

```
textRDD = spark.textFile('wordCountEx.txt')
textRDD.flatMap(lambda x: x.split()) \
       .map(lambda x: (x.lower(),1)) \
       .reduceByKey(add) \
       .sortBy(lambda x: x[1], ascending=False) \
       .collect()
```

TODO 3: Given a small DNA file named dna_small.txt, create a module that count continuous patterns of size k in a DNA string. For instance, if k = 5 and the DNA string: dna_str = 'gcctaagccta', the continuous patters are: 'gccta', 'cctaa', 'ctaag', 'taagc', 'aagcc', 'agcct', 'gccta'.

```
dnaRDD = spark.textFile('dna_small.txt')

dnaRDD.getNumPartitions()

target_partition_index = 1

k = 5

def generatePattern(line, length):
  sequence = [(line[i:i+length], 1) for i in range(len(line) - length
+ 1)]
  return sequence

seqRDD = dnaRDD.flatMap(lambda line: generatePattern(line,
k)).reduceByKey(add).sortBy(lambda x: x[1],
   ascending=False)

seqRDD.collect()
```

Tutorial 3 Solution

TODO 4: Improve the solution from TODO 3 to filter for palindrome patterns that occur more than once. Palindrome patterns are sequences that read the same forwards and backwards.

```
def palindromesPattern(item):
  l = ''
  r = ''
  key = item[0]

  for i in range(len(key)):
    l += key[i]

  for i in range(-1, -1*len(key)-1, -1):
    r += key[i]

  if l == r:
    palindromesPattern = (key, True)
  else:
    palindromesPattern = (key, False)

  return palindromesPattern

words = seqRDD.filter(lambda x: x[1] > 1) \
      .map(lambda x: palindromesPattern(x)) \
      .filter(lambda x: x[1] == True) \
      .collect()

for (word, isPalindromes) in words:
    print("{}".format(word))
```