PageRank Challenge

textLines = sc.textFile('wasb:///tmp/web-Google.txt')

links = textLines.map(lambda line: line.split('	')).groupByKey(128).cache()
ranks = links.mapValues(lambda x: 1.0)

prev_top = [(0, 0)] * 10

should_stop = False
n_iter = 1

while not should_stop:
    # Calculates URL contributions to the rank of other URLs.
    contribs = links.join(ranks) \
        .flatMap(lambda (source_node, (dest_node_list, cur_rank)): map(lambda dest_node: 
            (dest_node, cur_rank / len(dest_node_list)), dest_node_list))

    # Recalculates URL ranks based on neighbor contributions.
    ranks = contribs.reduceByKey(lambda x, y: (x + y)) \
        .mapValues(lambda rank: rank * 0.85 + 0.15).cache()
    cur_top = ranks.takeOrdered(10, key = lambda x: -x[1])

    should_stop = True
    for i in range(0, 10):
        if (cur_top[i][0] != prev_top[i][0]):
            should_stop = False

    prev_top = cur_top
    print("End of iteration " + str(n_iter))
    print(curtop10)
    n_iter += 1
Common mistakes - #1

• Integer division

```python
ranks = links.mapValues(lambda x : 1)
```

*instead of*

```python
ranks = links.mapValues(lambda x : 1.0)
```
Common mistakes - #2

• Sorting (slow and not necessary)

```python
ranks.sortBy(lambda r: r[1],False).take(10)
```

instead of

```python
ranks.takeOrdered(10, key=lambda x: -x[1])
```

• Even worse, usingsortByKey (inefficient swap required)

```python
rank.map(lambda x:(x[1],x[0])).sortByKey()
```
Common mistakes - #3

- No partitioning

Performance can be significantly increased using partitioning, ie.

```python
groupByKey(128).cache()
```
Common mistakes - #4

• Round (???)

In some solutions, a `round()` operator has been used. It is not clear the reason for using it.
Common mistakes - #5

• Links vs. Ranks

```
links = textLines.map(lambda line: line.split('	')).cache()
ranks = textLines.map(lambda line: line.split('	')).mapValues(lambda x : (x,1.0))

instead of

links = textLines.map(lambda line: line.split('	')).groupByKey(128).cache()
ranks = links.mapValues(lambda x : 1.0)
```
Common mistakes - #6

- Collect contributions on the local machine

```python
while (not should_stop):
    contribs = links.join(ranks).flatMap(...).collect()
    # recalculated contributions locally
    sc.parallelize(page_ranks)
```